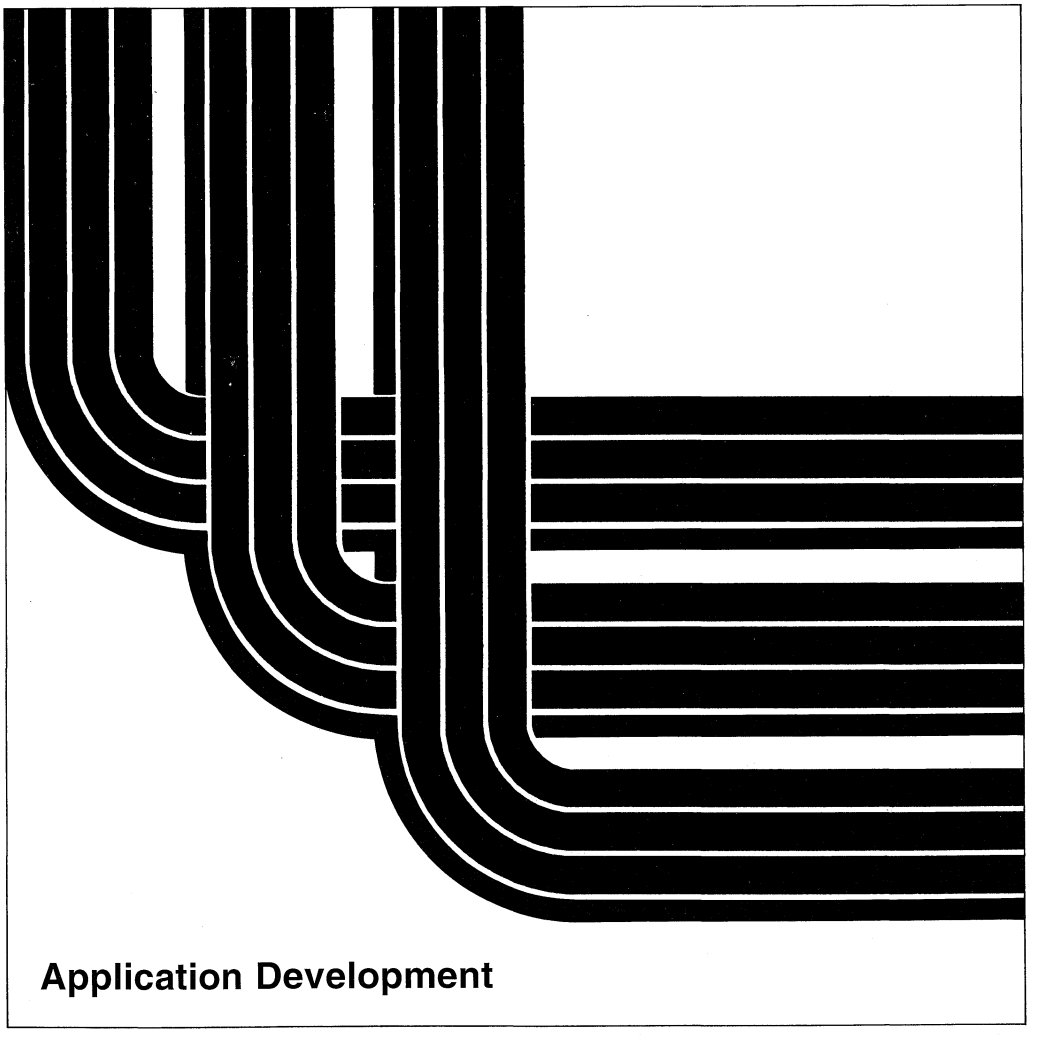
Application System/400

SC41-0025-01

Distributed Relational Database Guide

Version 2



Application Development

Application System/400

SC41-0025-01

Distributed Relational Database Guide

Version 2

Take Note!

Before using this information and the product it supports, be sure to read the general information under "Notices" on page xi.

Second Edition (November 1993)

This edition applies to the licensed program IBM Operating System/400, (Program 5738-SS1), Version 2 Release 3 Modification 0, and to all subsequent releases and modifications until otherwise indicated in new editions. This major revision makes obsolete SC41-0025-0 and Technical Newsletter SC41-0018-0. Make sure you are using the proper edition for the level of the product.

Order publications through your IBM representative or the IBM branch serving your locality. Publications are not stocked at the address given below.

A Customer Satisfaction Feedback form for readers' comments is provided at the back of this publication. If the form has been removed, you can mail your comments to:

Attn Department 245
IBM Corporation
3605 Highway 52 N
Rochester, MN 55901-7899 USA

or you can fax your comments to:

United States and Canada: 800+937-3430
Other countries: (+1)+507+253-5192

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you or restricting your use of it.

Contents

Figures

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of the intellectual property rights of IBM may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Commercial Relations, IBM Corporation, Purchase, NY 10577, U.S.A.

This publication could contain technical inaccuracies or typographical errors.

This publication may refer to products that are announced but not currently available in your country. This publication may also refer to products that have not been announced in your country. IBM makes no commitment to make available any unannounced products referred to herein. The final decision to announce any product is based on IBM's business and technical judgment.

Changes or additions to the text are indicated by a vertical line (|) to the left of the change or addition.

This publication contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

This publication contains small programs that are furnished by IBM as simple examples to provide an illustration. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. All programs contained herein are provided to you "AS IS". THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE EXPRESSLY DISCLAIMED.

Trademarks and Service Marks

The following terms, denoted by an asterisk (*) in this publication, are trademarks of the IBM Corporation in the United States or other countries or both:

AIX/6000
Application System/400
AS/400
C/400
COBOL/400
DATABASE 2
DB2
DB2/2
DB2/6000
Distributed Relational Database Architecture
DRDA
FORTRAN/400
IBM
MVS/ESA
NetView
Operating System/2
Operating System/400
OS/2
OS/400
RPG/400
SAA
SQL/DS
SQL/400
System Application Architecture
System/390
S/390
3090
400

About This Guide

This guide describes the distributed relational database management portion of the Operating System/400 (OS/400) licensed program. Distributed relational database management provides applications with access to data that is external to the application and typically located across a network of computers.

You may need to refer to other IBM manuals for more specific information about a particular topic. The *Publications Guide*, GC41-9678, provides information on all the manuals in the AS/400 library.

For a list of related publications, see the "Bibliography" on page H-1.

Who Should Use This Guide

This guide is intended primarily for system programmers responsible for the development, administration, and support of a distributed relational database on one or more AS/400 systems. System programmers who are not familiar with the AS/400 database can also get a view of the total range of database support provided by the OS/400 operating system. Application programmers may use this guide to see the system context in which distributed relational database applications run.

Before using this guide, you should be familiar with general programming concepts and terminology, and have a general understanding of the AS/400 system and OS/400 operating system.

Chapter 1. Distributed Relational Database and the AS/400 System

Distributed relational database architecture support on the AS/400* system is an implementation of IBM* Distributed Relational Database Architecture* (DRDA*) support. The Operating System/400* (OS/400*) and the Structured Query Language/400 (SQL/400*) licensed programs combine to provide support and functions for this architecture, which provides access to relational data across four IBM Systems Application Architecture* (SAA*) environments.

This chapter describes distributed relational database and how it is used on the AS/400 system. It defines some general concepts of distributed relational database, outlines the IBM DRDA implementation, and provides an overview of the current DRDA implementation on the AS/400 system. It defines some terms and directs you to other parts of this manual for more detail. Finally, an example corporation named *Spiffy* is described. This fictional company uses AS/400 systems in a distributed relational database application program. This sample of the Spiffy Corporation forms the background for all examples used in this manual.

You can find information about distributed relational database in the IBM SAA environment in the *SAA Introduction to Distributed Data*.

Distributed Relational Database Processing

A **relational database** is a set of data stored in one or more tables in a computer. A **table** is a two-dimensional arrangement of data consisting of horizontal rows and vertical columns as shown in Figure 1-1. Each **row** contains a sequence of values, one for each column of the table. A **column** has a name and contains a particular data type such as character, decimal, integer, and so on.

Prices

Item	Name	Price
78476	Baseball	12.95
78477	Football	29.50
78478	Basketball	22.75

Inventory

Item	Name	Supplier	Qty.
78476	Baseball	ACME	650
78477	Football	IMPERIAL	228
78478	Basketball	ACME	105

RV2W726-0

Figure 1-1. Typical Relational Tables

Tables can be defined and accessed in several ways on the AS/400 system. One way to describe and access tables on the system is to use a language like **Structured Query Language (SQL)**. SQL is the SAA database language and provides

the necessary consistency to enable distributed data processing across different system operating environments. Another way to describe and access tables on the AS/400 system is to describe physical and logical files using data description specifications (DDS).

SQL uses different terminology from that used on the AS/400 system. For most SQL objects there is a corresponding system object on the AS/400 system. Figure 1-2 shows the relationship between SQL relational database terms and AS/400 system terms.

Figure 1-2. Relationship of SQL Terms to System Terms

SQL Term	System Term
Collection. Consists of a library, a journal, a journal receiver, a data dictionary, and an SQL catalog. A collection groups related objects and allows you to find the objects by name.	**Library**. Groups related objects and allows you to find the objects by name.
Table. A set of columns and rows.	**Physical file**. A set of records.
Row. The horizontal part of a table containing a serial set of columns.	**Record**. A set of fields.
Column. The vertical part of a table of one data type.	**Field**. One or more bytes of related information of one data type.
View. A subset of columns and rows of one or more tables.	**Logical file**. A subset of fields and records of one or more physical files.
Index. A collection of data in the columns of a table, logically arranged in ascending or descending order.	A type of logical file
Package. An object that contains control structures for SQL statements to be used by an application server.	**SQL Package**. Has the same meaning as the SQL term.
Catalog. A set of tables and views that describe the tables, views and indexes in a collection, including column definitions.	No similar object. However, the Display File Description (DSPFD) and Display File Field Description (DSPFFD) commands provide some of the same information that querying an SQL catalog provides.

A **distributed relational database** exists when the application programs that use the data and the data itself are located on different systems. The simplest form of a distributed relational database is shown in Figure 1-3 on page 1-3 where the application program runs on one system, and the data is located on another system.

When using a distributed relational database, the system on which the application program is run is called the **application requester (AR)**, and the system on which the remote data resides is called the **application server (AS)**.

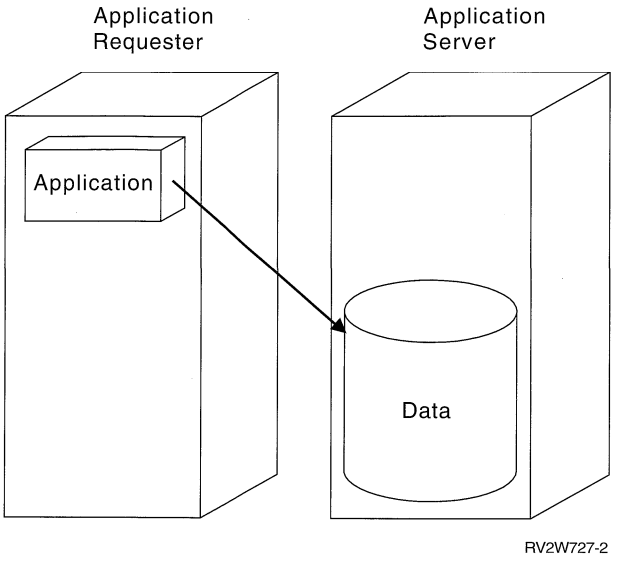


Figure 1-3. A Distributed Relational Database

A **unit of work** is one or more database requests and the associated processing that make up a completed piece of work as shown in Figure 1-4. A simple example is taking a part from stock in an inventory control application program. An inventory program can tentatively remove an item from a shop inventory account table and then add that item to a parts reorder table at the same location. The term transaction is another expression used to describe the unit of work concept.

In the above example, the unit of work is not complete until the part is both removed from the shop inventory account table and added to a reorder table. When the requests are complete, the application program can **commit** the unit of work. This means that any database changes associated with the unit of work are made permanent.

With unit of work support, the application program can also **roll back** changes to a unit of work. If a unit of work is rolled back, the changes made since the last commit or rollback operation are not applied. Thus, the application program treats the set of requests to a database as a unit.

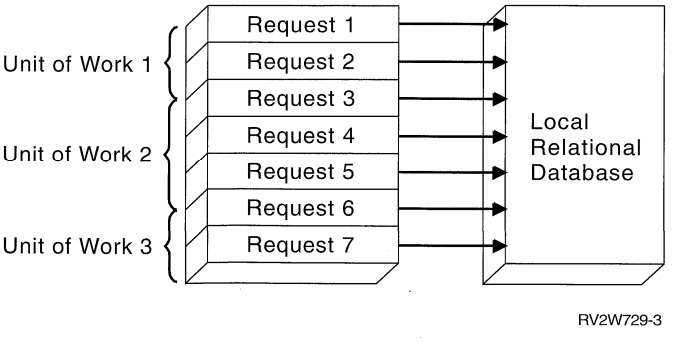


Figure 1-4. Unit of Work in a Local Relational Database

Remote Unit of Work

Remote unit of work (RUOW) is a form of distributed relational database processing in which an application program can access data on a remote database within a unit of work. A remote unit of work can include more than one relational database request, but all requests must be made to the same remote database. All requests to a relational database must be completed (either committed or rolled back) before requests can be sent to another relational database. This is shown in Figure 1-5.

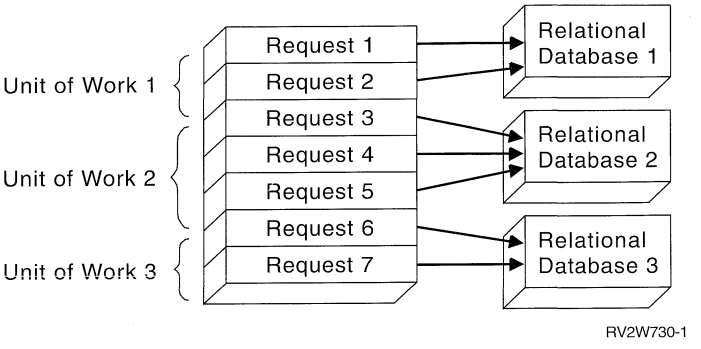


Figure 1-5. Remote Unit of Work in a Distributed Relational Database

Remote unit of work is application-directed distribution because the application program must connect to the correct relational database system before issuing the requests. However, the application program only needs to know the name of the remote database to make the correct connection.

Remote unit of work support enables an application program to read or update data at more than one location. However, all the data that the program accesses within a unit of work must be managed by the same relational database management system. For example, the shop inventory application program must commit its inventory and accounts receivable unit of work before it can read or update tables that are in another location.

In remote unit of work processing, each computer has an associated relational database management system and an associated application requester program that help process distributed relational data requests. This allows you or your application program to request remote relational data in much the same way as you request local relational data.

Other Distributed Relational Database Terms and Concepts

The following discussion provides an overview of additional distributed relational database concepts. Other SAA database managers may support some of these concepts, but the AS/400 system does not. The OS/400 database manager only supports remote unit of work. However, you can create your own version of these concepts as part of AS/400 application programs.

Distributed unit of work enables a user or application program to read or update data at multiple locations within a unit of work. The program issues relational database requests as though all the data is local. Although a unit of work may contain SQL requests to multiple systems, each SQL statement must refer to data at a single location.

A degree of processing sophistication beyond the distributed unit of work is a **distributed request**. This type of distributed relational database access enables a user or application program to issue a single SQL statement that can read or update data at multiple locations.

Tables in a distributed relational database do not have to differ from one another. Some tables can be exact or partial copies of one another. Extracts, snapshots, and replication are terms that describe types of copies using distributed processing. On the AS/400 system, these techniques can be simulated with user-written applications.

Extracts are user-requested copies of tables. The copies are extracted from one database and loaded into another specified by the user. The unloading and loading process may be repeated periodically to obtain updated data. Extracts are most useful for one-time or infrequent occurrences, such as read-only copies of data that rarely changes.

Snapshots are read-only copies of tables that are automatically made by a system. The system refreshes these copies from the source table on a periodic basis specified by the user—perhaps daily, weekly, or monthly. Snapshots are most useful for locations that seek an automatic process for receiving updated information on a periodic basis.

Data **replication** means the system automatically updates copies of a table. It is similar to snapshots because copies of a table are stored at multiple locations. Data replication is most effective for situations that require high reliability and quick data retrieval with few updates.

Tables can also be split across computer systems in the network. Such a table is called a **distributed table**. Distributed tables are split either horizontally by rows or vertically by columns to provide easier local reference and storage. The columns of a vertically distributed table reside at various locations, as do the rows of a horizontally distributed table. At any location, the user still sees the table as if it were kept in a single location. Distributing tables is most effective when the request to access and update certain portions of the table come from the same location as those portions of the table.

Distributed Relational Database Architecture Support

DRDA support for distributed relational database processing is used by

IBM relational database products. DRDA support defines protocols for communication between an application program and a remote relational database.

DRDA support provides distributed relational database management in the SAA environments. In the SAA environments, relational data is managed with the following programs:

- DATABASE2* (DB2*) on MVS
- Structured Query Language/Data System (SQL/DS*) on VM
- IBM Extended Services for OS/2*
- DATABASE2 OS/2 (DB2/2*) with Distributed Database Connection Services/2 (DDCS/2) for OS/2

- DATABASE 2 AIX/6000* (DB2/6000*) with Distributed Database Connection Services/6000 for AIX/6000
- The database management function of the OS/400 licensed program on the AS/400 system

Note: In this book, the term Distributed Database Connection Services (DDCS) refers to both DDCS/2 and DDCS/6000.

DRDA support provides the structure for access to database information for relational database managers operating in like and unlike environments. For example, access to relational data between two or more AS/400 systems is distribution in a **like environment**, and access to relational data between an AS/400 system and systems using the DB2 database manager is distribution in an **unlike environment**.

SQL is the SAA database language. It provides the necessary consistency to enable distributed data processing across like and unlike operating environments. Within DRDA support, SQL allows users to define, retrieve, and manipulate data across the SAA environments.

DRDA and CDRA Support

One of the interesting possibilities in a distributed relational database is that the database may not only span different types of computers, but those computers may be in different countries or regions. Different types of systems encode data differently. The same systems, such as AS/400 systems, can encode data differently depending on the language used on the system. For instance, a System/390 system, an AS/400 system, and a PS/2 system encode numeric data in their own unique formats. In addition, a System/390 system and an AS/400 system use the EBCDIC encoding scheme to encode character data, while a PS/2 system uses an ASCII encoding scheme.

For numeric data, these differences do not matter. Unlike systems that provide DRDA support automatically convert any differences between the way a number is represented in one computer system to the way it is represented in another. For example, if an AS/400 application program reads numeric data from a DB2 database, DB2 sends the numeric data in System/390 format and the OS/400 database management system converts it to the AS/400 numeric format.

However, the handling of character data is more complex, but this too can be handled within a distributed relational database.

Character Conversion

Not only can there be differences in encoding schemes (such as EBCDIC versus ASCII), but there can also be differences related to language. For instance, systems configured for different languages can assign different characters to the same code, or different codes to the same character. For example, a system configured for U.S. English can assign the same code to the character } that a system configured for the Danish language assigns to å. But those two systems can assign different codes to the same character such as $.

If data is to be shared across different systems, character data needs to be seen by users and applications the same way. In other words, a PS/2 user in New York and an AS/400 user in Copenhagen both need to see a $ as a $, even though $

may be encoded differently in each system. Furthermore, the user in Copenhagen needs to see a }, if that is the character that was stored at New York, even though the code may be the same as a Danish å. In order for this to happen, the $ must be converted to the proper character encoding for a PS/2 system (that is, U.S. English character set, ASCII), and converted back to Danish encoding when it goes from New York to Copenhagen (that is, Danish character set, EBCDIC). This sort of character conversion is provided for by the AS/400 system as well as the other SAA distributed relational database managers. This conversion is done in a coherent way in accordance with the **Character Data Representation Architecture (CDRA)**.

CDRA specifies the way to identify the attributes of character data so that the data can be understood across systems, even if the systems use different character sets and encoding schemes. For conversion to happen across systems, each system must understand the attributes of the character data it is receiving from the other system. CDRA specifies that these attributes be identified through a **coded character set identifier (CCSID)**. All character data in DB2, SQL/DS, and the OS/400 database management systems have a CCSID, which indicates a specific combination of encoding scheme, character set, and code page. All character data in an Extended Services environment has a code page only (but the other database managers treat that code page identification as a CCSID). A **code page** is a specific set of assignments between characters and internal codes.

For example, CCSID 37 means encoding scheme 4352 (EBCDIC), character set 697 (Latin, single-byte characters), and code page 37 (USA/Canada country extended code page). CCSID 5026 means encoding scheme 4865 (extended EBCDIC), character set 1172 with code page 290 (single-byte character set for Katakana/ Kanji), and character set 370 with code page 300 (double-byte character set for Katakana/Kanji).

DB2, SQL/DS, the OS/400 system, and DDCS/2 include mechanisms to convert character data between a wide range of CCSID-to-CCSID pairs and CCSID-to-code page pairs. Character conversion for many CCSIDs and code pages is already built into these products. For a complete list and description of all CCSIDs registered in CDRA, see *Character Data Representation Architecture - Level 1 Registry*. For a description of the use of CCSIDs on the AS/400 system, see "Coded Character Set Identifier (CCSID)" on page 10-9.

Distributed Relational Database on the AS/400 System

All data on the AS/400 system is stored in a single relational database. The OS/400 program provides all the database management functions for this AS/400 relational database. Distributed relational database support on the system is an integral part of the OS/400 program, just as is support for communications, work management, security functions and other functions.

The AS/400 system can be part of a distributed relational database network with other systems that support a DRDA implementation. The AS/400 system can be an AR or an AS in either like or unlike environments. Distributed relational database implementation on the AS/400 system supports remote unit of work. An AS/400 system can send multiple SQL requests to a database on another system and have the requests be treated as a single unit of work. Support for remote unit of work requires that all work in that unit of work be in the same database. This means you cannot get data from a remote database and put the data into the local

database within a unit of work. However, you could get data from a remote database with one unit of work and update a local database in another unit of work.

On the AS/400 system, the conceptual terms snapshots and replication, introduced in "Other Distributed Relational Database Terms and Concepts" on page 1-4, are not automatically performed by the system. These functions can be simulated through user application programs. Extracts can also be handled by user-written applications. Tables can be distributed to meet the best distributed relational database planning and design interests of the organization. More information about how you can organize these functions in a distributed relational database is discussed in Chapter 7, "Data Availability and Protection."

The OS/400 program includes run-time support for SQL. You do not need the SQL/400 licensed program installed on an AS/400 application requester or application server to process distributed relational database requests. However, you do need the SQL/400 program to precompile programs with SQL statements, run interactive SQL, or run SQL/400 Query Manager.

Managing an AS/400 Distributed Relational Database

Managing a distributed relational database on the AS/400 system requires broad knowledge of the resources and tools within the OS/400 licensed program. This guide provides an overview of the various functions available with the operating system that can help you administer a distributed relational database on AS/400 systems. This guide explains distributed relational database functions and tasks in a network of AS/400 systems (a *like* environment). Differences between AS/400 distributed relational database functions in a like and unlike environment are presented only in a general discussion in this guide. Considerations for different distributed relational database platforms working with AS/400 distributed relational database are discussed in Appendix B, "Cross-Platform Distributed Relational Database Notes" on page B-1. If you want more information about another IBM system that supports distributed relational database, see the information provided with that system or the manuals listed in "Distributed Relational Database Library" and "Other IBM Distributed Relational Database Platform Libraries" in the "Bibliography."

Planning and Design

The first requirement for the successful operation of a distributed relational database is thorough planning. The needs and goals of your enterprise must be considered when making the decision to use a distributed relational database. How you code an application program, where it resides in relation to the data, and the network design that connects application programs to data are all important design considerations.

Database design in a distributed relational database is more critical than when dealing with just one AS/400 relational database. With more than one AS/400 system to consider, you must develop a consistent management strategy across the network. Operations that require particular attention when forming your strategy are: general operations, system security, accounting, problem analysis, and backup and recovery processes. Chapter 2, "Planning and Design for Distributed Relational Database" discusses some things to consider when planning for and designing a distributed database.

Communications

The communications support for AS/400 distributed relational database is the same used by AS/400 Distributed Data Management (DDM) Architecture. This support includes the IBM Systems Network Architecture (SNA) through advanced program-to-program communications (APPC), with or without advanced peer-to-peer networking (APPN). Ethernet, integrated services digital network (ISDN), synchronous data link control (SDLC), token ring network, or X.25 data link protocols are supported. Functions such as display station pass-through, SNA distribution services (SNADS), alert support, and DDM support are helpful in managing a distributed relational database network. See Chapter 3, "Communications for an AS/400 Distributed Relational Database" for more information about these functions and configuration samples.

Security

The AS/400 system has security functions built into the operating system to limit access to the data resources of an application server. Security options range from simple physical security to full password security coupled with authorization to commands and data objects. Users must be properly authorized to have access to the database whether it is local or remote. They must also have proper authorization to collections, tables, and other relational database objects necessary to run their application programs. This typically means that distributed database users must have valid user profiles for the databases they use throughout the network. Security planning must consider user and application program needs across the network. Chapter 4, "Security for an AS/400 Distributed Relational Database" provides information on the security considerations for an AS/400 distributed relational database.

Set Up

The run-time support for an AS/400 distributed relational database is provided by the OS/400 program. Therefore, when the operating system is installed, distributed relational database support is installed. However, some setup work is required to make the application requesters and application servers ready to send and receive work. One or more subsystems can be used to control interactive, batch, spooled, and communications jobs. All the systems in the network must also have their relational database directory set up with connection information. Finally, you may wish to put data into the tables of the application servers throughout the network.

The **relational database directory** contains database names and values that are translated into communications network parameters. You add an entry for each database in the network, including the local database. Each directory entry consists of a unique relational database name and corresponding communications path information.

There are a number of ways to enter data into a database. You can use an SQL application program, some other high-level language application program, or one of these methods:

- Interactive SQL
- OS/400 query management
- Data file utility (DFU)
- Copy File (CPYF) command

For more information on ways to enter data into a distributed database, along with a discussion of subsystems and relational database directories on the AS/400 system, see Chapter 5, "Setting Up an AS/400 Distributed Relational Database."

Administration

As a database administrator in a distributed relational database network, you may need to locate and monitor work being done on any one of several systems. Work management functions on the AS/400 system provide effective ways to track this work by allowing you to do the following:

- Work with jobs in the network
- Work with the communications networks, controllers, devices, modes, and sessions
- Enable DDM conversations
- Grant and remove authority for users
- Stop work on a remote system

You can read more about how to do these tasks in Chapter 6, "Distributed Relational Database Administration and Operation Tasks."

Data Protection and Availability

The AS/400 system provides a wide array of functions to ensure that data on systems in a distributed relational database network is available for use. These include save/restore functions of the system, journal management and access path journaling, commitment control, auxiliary storage pools, checksum protection, mirrored protection and the uninterruptible power supply. While the system operator for each AS/400 system is typically responsible for backup and recovery of that system's data, you should address provisions for network redundancy and data redundancy to provide optimum data availability. Chapter 7, "Data Availability and Protection" discusses these topics in more detail.

Performance

No matter what kind of application programs you are running on a system, performance can always be a concern. For a distributed relational database, network, system, and application performance are all crucial. Factors that affect network performance include line speed, connection type, frame size, request unit (RU) sizing, and pacing. System performance can be affected by the size and organization of main and auxiliary storage. There can also be performance gains if you know the strengths and weaknesses of SQL programs. Chapter 9, "Handling Distributed Relational Database Problems" offers guidance in these areas.

Problems

When a problem occurs within a distributed relational database, it is necessary to first identify where the problem originates. The problem may be on the application server or application requester. When the database problem is located properly in the network, you may further isolate the problem as a user problem, a problem with an application program, an AS/400 system problem, or communications problem in order to correct the error. Chapter 9, "Handling Distributed Relational Database Problems" describes the methods you can use to isolate and solve distributed database problems.

Application Programming

Programmers can write high-level language programs that use SQL/400 statements for AS/400 distributed application programs. The main differences from programs written for local processing only are the ability to connect to remote databases and to create SQL packages. The CONNECT SQL statement can be used to explicitly connect an application requester to an application server, or the name of the relational database can be specified when the program is created.

An **SQL package** is an AS/400 object used only for distributed relational databases. It can be created as a result of the precompile process of SQL or can be created from a compiled program object. An SQL package resides on the application server. It contains SQL statements, host variable attributes, and access plans which the application server uses to process an application requester's request.

Because application programs can connect to many different systems, programmers may need to pay more attention to data conversion between systems. The AS/400 system provides for conversion of various types of data, including coded character set identifier (CCSID) support for the management of character information.

See Chapter 10, "Writing Distributed Relational Database Applications" for an overview of distributed relational database topics for the application programmer.

Spiffy Corporation Example

The Spiffy Corporation is used in several IBM manuals to describe distributed relational database support. In this manual, this fictional company has been changed somewhat to illustrate AS/400 support for DRDA support in an AS/400 network. Examples used throughout this manual illustrate particular functions, connections, and processes. These may not correspond exactly to the examples used in other distributed relational database publications but an attempt has been made to make them look familiar.

Though the Spiffy Corporation is a fictional enterprise, the business practices described here are modeled after those in use in several companies of similar construction. However, this example does not attempt to describe all that can be done using a distributed relational database, even by this example company.

Spiffy Organization and System Profile

Spiffy Corporation is a national product distributor that sells and services automobiles, among other products, to retail customers through a network of regional offices and local dealerships. Given the high competitiveness of today's automobile industry, the success of an operation like the Spiffy Corporation depends on high-quality servicing and timely delivery of spare parts to the customer. To meet this competition, Spiffy has established a vast service network incorporated within its dealership organization.

The dealership organization is headed by a central vehicle distributor that is located in Chicago, Illinois. There are several regional distribution centers across North America. Two of these are located in Minneapolis, Minnesota and Kansas City, Missouri. These centers minimize the distribution costs of vehicles and spare parts by setting up regional inventories. The Minneapolis regional center serves approxi-

mately 15 dealerships while the Kansas City center serves as many as 30 dealerships.

Figure 1-6 illustrates a system organization chart for Spiffy Corporation.

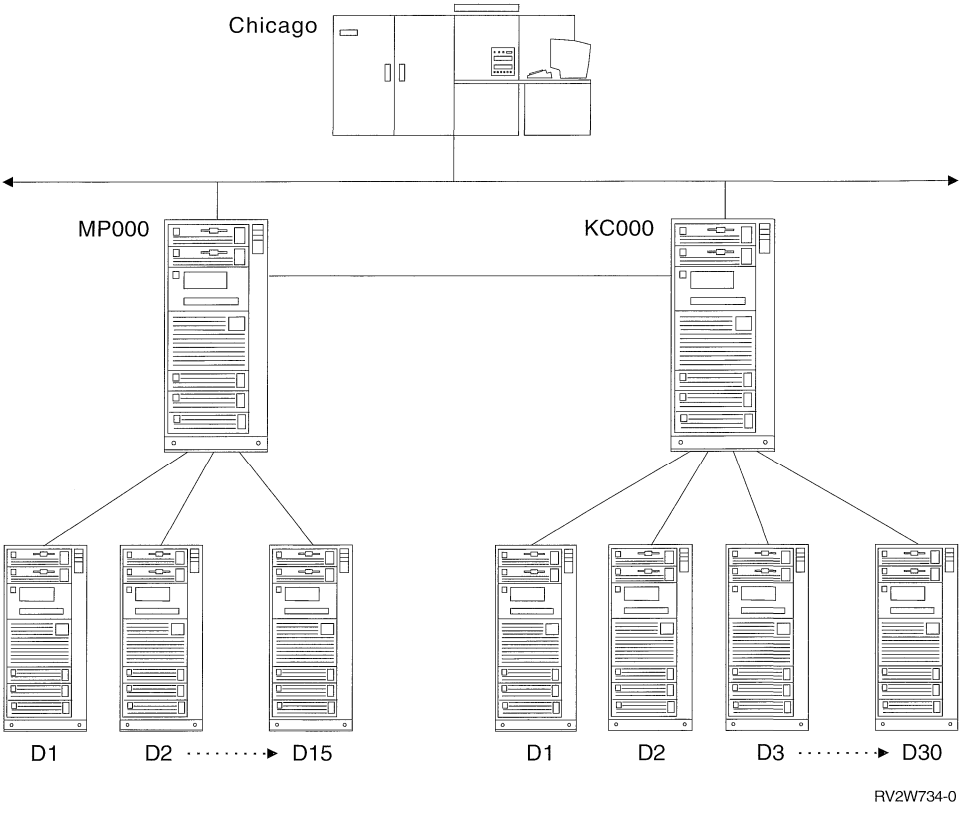


Figure 1-6. The Spiffy Corporation System Organization

Spiffy is in the process of building up a nationwide integrated telecommunications network. For the automobile division they are setting up a network of AS/400 systems for the regional distributions centers and the dealerships. These are connected to an IBM 3090* at the central vehicle distributor. This network is considered a vital business asset for maintaining the competitive edge.

The central distributor runs MVS/ESA* on its IBM 3090 system with DB2 and relevant decision support software. This system is used because of the large amounts of data that must be handled at any one time in a variety of application programs. The central vehicle distributor system is not dedicated to automobile division data processing. It must handle work and processes for the corporation that do not yet operate in a distributed database environment. The regional centers are running AS/400 Model D60 systems. They use APPC/APPN with SNADS and 5250 Display Station Pass-through using an SDLC protocol.

All of the dealerships use AS/400 systems but they may range in size from a D06 in the smaller enterprises up to a D60 in the largest. These systems are connected to the regional office using SDLC protocol. The largest dealerships have a part time programmer and a system operator to tend to the data processing functioning of the enterprise. Most of the installations do not employ anyone with programming expertise and some of the smaller locations do not employ anyone with more than a very general knowledge of computers.

Business Processes of the Spiffy Corporation Automobile Service

The Spiffy Corporation automobile division has business practices that are automated in this distributed relational database environment. To keep the examples from becoming more complicated than necessary, consider just those functions in the company that pertain to vehicle servicing.

Dealerships can have a list of from 2000 to 20,000 customers. This translates to 5 service orders per day for a small dealership and up to 50 per day for a large dealership. These service orders include scheduled maintenance, warranty repairs, regular repairs, and parts ordering.

The dealers stock only frequently needed spare parts and maintain their own inventory databases. Both regional centers provide parts when requested. Dealer inventories are also stocked on a periodic basis by a forecast-model-controlled batch process.

Distributed Relational Database Administration for the Spiffy Corporation

Each dealership manages its data processing resources and procedures as a stand-alone enterprise. Spiffy Corporation requires that each dealership have one or more AS/400 systems and that those systems must be available to the network at certain times. However, the size of the system and the number of business processes that are automated on it are determined by each dealership's needs and the resources available to it.

The Spiffy Corporation requires all dealerships to be active in the inventory distributed relational database. Since the corporation operates its own dealerships, it has a full complement of dealership software that may or may not access the distributed relational database environment. The Spiffy dealerships use the full set of software tools. Most of the private franchises use them also since they are tailored specifically to the Spiffy Corporation way of doing business.

The regional distribution centers manage the inventory for their region. They also function as the database administrator for all distributed database resources used in the region. The responsibilities involved vary depending on the level of data processing competency at each dealership. The regional center is always the first contact for help for any dealership in the region.

The Minneapolis regional distribution center has a staff of AS/400 programmers with a wide range of experience and knowledge about the systems and the network. The dealership load is about one half that of other regional centers to allow this center to focus on network-wide AS/400 support functions. These functions include application program development, program maintenance, and problem handling.

The following are the database responsibilities for each level of activity in the network:

Dealerships
- Perform basic operation and administration of system
- Enroll local users

Regional distribution centers
- Set up data processing for new dealerships

- Disperse database resources for discontinued dealerships
- Enroll network users in region
- Maintain inventory for region
- Develop service plans for dealerships
- Operate help desk for dealerships

In addition to the regional distribution center activities above, the Minneapolis AS/400 competency center does the following activities:

- Develop applications for AS/400 network
- Operate help desk for regional centers
- Tune database performance
- Alert focal point
- Resolve database problems

Examples used throughout this manual are associated with one or more of these activities. Many examples show the process of obtaining a part from inventory in order to schedule customer service or repairs. Others show distributed relational database administration tasks used to set up, secure, monitor, and resolve problems for systems in the Spiffy Corporation distributed relational database network.

Chapter 2. Planning and Design for Distributed Relational Database

To prepare for a distributed relational database, you must understand both the needs of the business and relational database technology.

Because the planning and design of a distributed relational database are closely linked to each other, this chapter combines these topics when discussing the following related tasks:

- Identifying your needs and expectations
- Designing the application, data, and network
- Putting together a management strategy

Additional information about planning for distributed relational databases can be found in *Planning for Distributed Data*.

Identifying Your Needs and Expectations

When analyzing your needs and expectations of a distributed relational database, consider the following:

1. Data needs. What data is pertinent to your plans, who will need it, for what reason, and how often?
2. Distributed relational database capabilities. Do the requirements lend themselves to a distributed relational database solution?
3. Goals and directions. If a distributed relational database appears to be a viable solution, what short-term and long-term goals can be met?

Data Needs

The first step in your analysis is to determine which factors affect your data and how they affect it. Ask yourself the following questions:

- What locations are involved?
- What kind of transactions do you envision?
- What data is needed for each transaction?
- What dependencies do items of data have on each other, especially referential limitations? For example, will information in one table need to be checked against the information in another table? (If so, both tables must be kept at the same location.)
- Does the data currently exist? If so, where is it located? Who "owns" it (that is, who is responsible for maintaining the accuracy of the data)?
- What priority do you place on the availability of the needed data? Integrity of the data across locations? Protection of the data from unauthorized access?
- What access patterns do you envision for the data? For instance, will the data be read, updated, or both? How frequently? Will a typical access return a lot of data or a little data?
- What level of performance do you expect from each transaction? What response time is acceptable?

Distributed Relational Database Capabilities

The second step in your analysis is to decide whether or not your data needs lend themselves to a distributed relational database solution.

Applications where most database processing is done locally and access to remote data is needed only occasionally are typically *good* candidates for a distributed relational database.

Applications with the following requirements are usually *poor* candidates for a distributed relational database:

- The data is kept at a central site *and* most of the work that a remote user needs to do is at the central site.
- Consistently high performance, especially consistently fast response time, is needed. It takes longer to move data across a network.
- Consistently high availability, especially twenty-four hour, seven-day-a-week availability, is needed. Networks involve more systems and more in-between components, such as communications lines and communications controllers, which increases the chance of breakdowns.
- A distributed relational database function that you need is not currently available or announced.

Goals and Directions

The third step in your analysis is to assess your short-term and long-term goals.

SQL is the SAA database language. If your goals and directions include portability or remote data access on unlike systems, you should use distributed relational database on the AS/400 system.

Also, if the distributed database function of remote unit of work matches your current data needs, an AS/400 distributed database may do well for your business. However, if your distributed database application requires other function such as extracts, snapshots, or replication, other options are available to you on the AS/400 system until the function is made available on the operating system. For example, you may do one of the following:

- Provide the needed function yourself
- Stage your plans for distributed relational database to allow for new function to become available
- Reassess your goals and requirement to see if you can satisfy them with a currently available or announced function. Some alternative solutions are listed in Figure 2-1. These alternatives can be used to supplement or replace available function.

Figure 2-1. Alternative Solutions to Distributed Relational Database

Solution	Description	Advantages	Disadvantages
Distributed Data Management (DDM)	A function of the operating system that allows an application program or user on one system to use database files stored on a remote system. The system must be connected by a communications network, and the remote system must also use DDM.	• For simple read and update accesses, the performance is better than for SQL. • Existing applications do not need to be rewritten. • Can be used to access S/38, S/36, CICS, and PC Support • DDM is an SAA function	• SQL is more efficient for complex functions • May not be able to access other distributed relational database platforms • Does not perform CCSID and numeric data conversions
Intersystem Communications Function/Common Programming Interface (ICF/CPI Communications)	ICF is a function of the operating system that allows a program to communicate interactively with another program or system. CPI Communications is a call-level interface that provides a consistent application interface for applications that use program-to-program communications. These interfaces make use of SNA's logical unit (LU) 6.2 architecture to establish a conversation with a program on a remote system, to send and receive data, to exchange control information, to end a conversation, and to notify a partner program of errors.	• Allows you to customize your application to meet your needs. • Can provide better performance.	Compared to distributed relational database and DDM, a slightly more complicated program is needed to support communications and data conversion requirements.
Display station pass-through	A communications function that allows a user to sign on to one AS/400 system from another AS/400 system and use that system's programs and data.	• Applications and data on remote systems are accessible from local system. • Allows for quick access when data is volatile and a large amount of data on one system is needed by users on several systems.	Response time on screen updates is slower than locally attached devices.

A distributed relational database usually evolves from simple to complex as business needs change and new products are made available. Remember to consider this when analyzing your needs and expectations.

Designing the Application, Network, and Data

Designing a distributed relational database involves making choices about the applications, network, and data.

Application Considerations

Distributed relational database applications have different requirements than applications developed solely for use on a local database. To properly plan for these differences, design your applications with the following in mind:

- Take advantage of remote unit of work (RUOW) processing.
- Plan for types of access that will become available in the future, such as distributed unit of work.
- Code programs using SAA interfaces.
- Consider dividing a complex application into smaller parts and placing each piece of the application in the location best suited to process it.
- Investigate how the initial database applications will be prepared, tested, and used.
- Take advantage, when possible, of SQL set-processing capabilities. This will minimize communication with the application servers. For example, update multiple rows with one SQL statement whenever you can.
- Be aware that database operations within a unit of work must be done at a single site. There are two ways to avoid the RUOW restriction:
  - Integrated Language Environment (ILE) multiple activation group support makes it possible for an application to have active connections to multiple application servers concurrently within an application. The application must assume some responsibility for error recovery when you use this technique of extending the RUOW capability.
  - Files you open without commitment control are free from the RUOW restriction. For example, to access a local table within the same unit of work that updates a DB2 table, open the local table under native file support while the DB2 table is accessed by SQL. This makes use of the following OS/400 features:
    - Native database files can be accessed with or without commitment control protection.
    - Native file support can be used to access tables within an SQL collection.

    You can extend this approach by replacing the local native file with a DDM file. This allows access to data at a second remote system within the same unit of work. When you use either of these approaches, the application program is responsible for coordinating any recovery that may be necessary. Changes to files that are not under commitment control cannot be rolled back.

Network Considerations

The design of a network directly affects the performance of a distributed relational database. To properly design a distributed relational database that works well with a particular network, do the following:

- Because the line speed can be very important to application performance, provide sufficient capacity at the appropriate places in the network to achieve efficient performance to the main distributed relational database applications.
- Evaluate the available communication hardware and software and, if necessary, your ability to upgrade.
- Consider the session limits and conversation limits specified when the network is defined.
- Identify the hardware, software, and communication equipment needed (for both test and production environments), and the best configuration of the equipment for a distributed relational database network.
- Take into consideration the initial service level agreements with end user groups (such as what response time to expect for a given distributed relational database application), and strategies for monitoring and tuning the actual service provided.
- Develop a naming strategy for database objects in the distributed relational database and for each location in the distributed relational database system. A **location** is a specific relational database management system in an interconnected network of relational database management systems that participate in distributed relational database. Consider the following when developing this strategy:
  - The fully qualified name of an object in a distributed database system has three (rather than two) parts, and the highest-level qualifier identifies the location of the object.
  - Each location in a distributed relational database system should be given a unique identification; each object in the system should also have a unique identification. Duplicate identifications can cause serious problems. For example, duplicate locations and object names may cause an application to connect to an unintended remote database, and once connected, access an unintended object. Pay particular attention to naming when networks are coupled.

Data Considerations

The placement of data in respect to the applications that need it is an important consideration when designing a distributed relational database. When making such placement decisions, consider the following:

- The level of performance needed from the applications
- Requirements for the security, currency, consistency, and availability of the data across locations
- The amount of data needed and the predicted patterns of data access
- If the distributed relational database functions needed are available
- The skills needed to support the system and the skills that are actually available

- Who "owns" the data (that is, who is responsible for maintaining the accuracy of the data)
- Management strategy for cross-system security, accounting, monitoring and tuning, problem handling, data backup and recovery, and change control
- Distributed database design decisions, such as where to locate data in the network and whether to maintain single or multiple copies of the data

Developing a Management Strategy

The management strategy for a distributed relational database includes the following:

- General operations
- Security
- Accounting
- Problem analysis
- Backup and recovery

General Operations

To plan for the general operation of a distributed relational database, consider both performance and availability. The following design considerations can help you improve both the performance and availability of a distributed relational database:

- If an application involves transactions that run frequently or that send or receive a lot of data, you should try to keep it in the same location as the data.
- For data that needs to be shared by applications in different locations, put the data in the location with the most activity.
- If the applications in one location need the data as much as the applications in another location, consider keeping copies of the data at both locations. When keeping copies at multiple locations, ask yourself the following questions about your management strategy:

  Will users be allowed to make updates to the copies?

  How and when will the copies be refreshed with current data?

  Will all copies have to be backed up or will backing up one copy be sufficient?

  How will general administration activities be performed consistently for all copies?

  When is it permissible to delete one of the copies?
- Consider whether the distributed databases will be administered from a central location or from each database location.

Performance may also be improved by doing the following:

- If data and applications must be kept at different locations, do the following to keep the performance within acceptable limits:
  - Keep data traffic across the network as low as possible by only retrieving the data columns that will be used by the application; that is, avoid using * in place of a list of column names as part of a SELECT statement.

  - Discourage programmers from coding statements that send large amounts of data to or receive large amounts of data from a remote location; that is, encourage the use of the WHERE clause of the SELECT statement to limit the number of rows of data.
  - Use read-only queries where appropriate by specifying the FOR FETCH ONLY clause.
  - Keep the number of accesses to remote data low by using local data in place of remote data whenever possible.
  - Use SQL SET operations to process multiple rows at the application requester with a single SQL request.
  - Try to avoid the dropping of connections by using DDMCNV(*KEEP) in general attempt to minimize the number of connection changes.
- Provide sufficient network capacity by doing the following:
  - Increase the capacity of the network by installing high-speed, high-bandwidth lines or by adding lines at appropriate points in the network.
  - Reduce the contention or improve the contention balance on certain processors. For example, move existing applications from a host system to a departmental system or group some distributed relational database work into batch.
- Encourage good table design. At the distributed relational database locations, encourage appropriate use of primary keys, table indexes, and normalization techniques.

Availability may also be improved by doing the following:

- In general, try to limit the amount of data traffic across the network.
- If data and applications must be kept at different locations, do the following to keep the availability within acceptable limits:
  - Establish alternate network routes.
  - Consider the effect of time zone differences on availability:
    - Will qualified people be available to bring up the system?
    - Will off-hours batch work interfere with processing?
  - Ensure good backup and recovery features.
  - Ensure people are skilled in backup and recovery.

Security

Part of planning for a distributed relational database involves the decisions you must make about securing distributed data. These decisions include:

- What systems should be made accessible to users in other locations and which users in other locations should have access to those systems.
- How tightly controlled access to those systems should be. For example, should a user password be required when a conversation is started by a remote user?
- What data should be made accessible to users in other locations and which users in other locations should have access to that data.
- What actions those users should be allowed to take on the data.

- Whether authorization to data should be centrally controlled or locally controlled.
- If special precautions should be taken because multiple systems are being linked. For example, should name translation be used?

When making the previous decisions, consider the following when choosing locations:

- Physical protection. For example, a location may offer a room with restricted access.
- Level of system security. The level of system security often differs between locations. The security level of the distributed database is no greater than the lowest level of security used in the network.

  All systems can do the following:

  - Verify that when one system receives a request to communicate with another system in the network, the requesting system is actually "who it says it is" and that it is authorized to communicate with the receiving system.
  - Pass a user's identification from the local system to the remote system for verification before any remote data access is allowed.
  - Grant and revoke privileges to access and manipulate SQL objects such as tables and views.

  The AS/400 system includes security audit functions that allow you to track unauthorized attempts to access data, as well track other events pertinent to security. The system also provides a function that can prevent all distributed database access from remote systems.
- Security-related costs. When considering the cost of security, consider both the cost of buying security-related products and the price of your information staff's time to perform the following activities:
  - Maintain system identification of remote-data-accessing users at both local and remote systems.
  - Coordinate auditing functions between sites.

Accounting

You need to be able to account and charge for the use of distributed data. Consider the following:

- Accounting for the use of distributed data involves the use of resources in one or more remote systems, the use of resources on the local system, and the use of network resources that connect the systems.
- Accounting information is accumulated by each system independently. Network accounting information is accumulated independent of the data accumulated by the systems.
- The time zones of various systems may have to be taken into account when trying to correlate accounting information. Each system clock may not be synchronized with the remote system clock.

- Differences may exist between each system's permitted accounting codes (numbers). For example, the AS/400 system restricts accounting codes to a maximum of 15 characters.

The following functions are available to account for the use of distributed data:

- AS/400 job accounting journal. The AS/400 system writes job accounting information into the job accounting journal for each distributed relational database application. The Display Journal (DSPJRN) command can be used to write the accumulated journal entries into a database file. Then, either a user-written program or query functions can be used to analyze the accounting data. For more information, see "Job Accounting" on page 6-20.
- NetView* accounting data. The NetView licensed program can be used to record accounting data about the use of network resources.

Problem Analysis

Problem analysis needs to be managed in a distributed database environment. Problem analysis involves both identifying and resolving problems for applications that are processed across a network of systems. Consider the following:

- Distributed database processing problems manifest themselves in various ways. For example, an error return code may be passed to a distributed database application by the system that detects the problem. In addition, responses may be slow, wrong, or nonexistent.
- Tools are available to diagnose distributed database processing problems. For example, each distributed relational database product provides trace functions that can assist in diagnosing distributed data processing problems.
- When system failures are detected by an AS/400 system, the system does the following:
  - Logs information about program status immediately after the failure is detected.
  - Produces an alert message. All the alerts can be directed to a single control point in the network. This can be either the NetView licensed program or an AS/400 system.
  - If a correction to an IBM program is required and if you have a System/390* with Network Distribution Manager (NDM) installed in the network, you can use the NDM and the Distributed System Node Executive products to receive and transmit updates and replacements to appropriate systems in the network.

Backup and Recovery

In a single-system environment, backup and recovery takes place locally. But in a distributed database, backup and recovery also affects remote locations.

The AS/400 system allows individual tables, collections, or groups of collections to be backed up and recovered. Although backup and recovery can only be done locally, you may want to plan to have less critical data on a system that does not have adequate backup support. Backup and recovery procedures must be consistent with data that may exist on more than one application server. Because you have more than one system in the network, you may want to save such data to a second system so that it is always available to the network in some form. Strate-

gies such as these need to be planned and laid out specifically before a database is distributed across the network.

Chapter 3. Communications for an AS/400 Distributed Relational Database

This chapter describes which communications functions to use when you are setting up a network or changing an existing network to work with a distributed relational database. This guide does not contain all the information you need. It is intended to help you ask the right questions and determine your own answers, that ensures maximum use of your resources based on the needs of your business.

This chapter discusses distributed relational database supported communications functions, including communications types and lines, and AS/400 functions, such as alert support for problem notification and display station pass-through for managing activity on a remote system. An overview of network considerations gives you a better understanding of the various options available when making a decision about your network hardware and design. Steps for configuring a network, configuring alerts, and configuring systems for display station pass-through are provided with an accompanying configuration example.

Additional information about how to connect unlike systems in a network for distributed relational database work can be found in the *Distributed Relational Database Connectivity Guide*.

Communications Tools

Communications support for distributed relational database on the AS/400 system is provided under the IBM Systems Network Architecture (SNA) through the advanced program-to-program communications (APPC) communications type, with or without advanced peer-to-peer networking (APPN). Ethernet, integrated service digital network (ISDN), synchronous data link control (SDLC), token-ring network, or X.25 data link protocols can be used.

Systems Network Architecture

SNA is an architecture made up of several logical unit (LU) types. These logical units are architectural definitions of how to communicate with systems, controllers, and terminals that also support the same LU types. All of the SNA support necessary for distributed relational database on the AS/400 system is part of the OS/400 licensed program.

APPC/APPN

APPC is the AS/400 system implementation of SNA LU 6.2 and physical unit (PU) T2.1 architectures. It allows applications that reside on different processors to communicate and exchange data in a peer relationship with one another.

APPN support is an enhancement to the PU T2.1 architecture that provides networking functions such as:

- Dynamically locating LUs in the network by searching distributed directories
- Dynamically selecting routes to LUs based on selection characteristics when an application requests a session

- Intermediate routing of LU 6.2 session traffic through the node for sessions between other LU 6.2 partners
- Routing session data based on transmission priorities
- Dynamically creating and starting remote location partner definitions

APPC and APPN also support these IBM-supplied functions:

- SNA distribution services (SNADS)
- Display station pass-through to the AS/400 system
- Alert support to help you manage problems from a central location

Using DDM and Distributed Relational Database

Distributed relational database implementation on the AS/400 system uses Distributed Data Management (DDM) architecture commands to communicate with other systems. However, distributed relational database and DDM support handle some functions differently.

Using distributed relational database processing, the application connects to a remote system using a relational database directory on the local system. The relational database directory provides the necessary links between a relational database name and the communications path to that database. An application running under distributed relational database only has to identify the database name and run the SQL statements needed for processing.

Using DDM support, the remote file is identified and the communications path is provided by means of a DDM file on the local system.

You can use DDM to support distributed relational database processing for administrative tasks such as submitting remote commands, copying files, and moving data from one system to another. To use DDM support, a DDM file must be created. This is discussed in "Setting Up DDM Files" on page 5-10. Using a DDM file with the AS/400 copy file commands is discussed in "Using Copy File Commands" on page 5-16. Using DDM files to submit a remote command is discussed in "Submit Remote Command (SBMRMTCMD) Command" on page 6-8.

Systems Network Architecture Distribution Services

Systems Network Architecture distribution services (SNADS) provides the capability to distribute objects such as SQL objects, job streams, and messages to other systems that are directly or indirectly attached to your system. A SNADS configuration must exist on all systems that use the distribution services. With SNADS, distribution can be to:

- One local user
- One remote user
- A list of local or remote users identified on a SNADS distribution list

Objects are distributed using the Send Network File (SNDNETF) and Receive Network File (RCVNETF) commands. The SNDNETF command should be run directly on the system that contains the data being sent. The RCVNETF command must be run on the system where the data is sent. The Work With Network File (WRKNETF) command provides a display with options to receive and see network files waiting to be received. To run the appropriate command on the remote system, you can use either display station pass-through or the Submit Remote Command (SBMRMTCMD) command.

When you configure your network for distributed relational database processing, all the equipment and functions are in place to support SNADS. However, you must configure SNADS using the Configure Distribution Services (CFGDSTSRV) command. You may also want to review the number of sessions and conversations described for the communications mode description as you review the potential increase in line use associated with a distributed relational database environment. For a sample SNADS configuration see "Configuring SNADS Support" on page 3-24. For more information on SNADS, see the *Distribution Services Network Guide*.

Alert Support

Alert support on the AS/400 system allows you to manage problems from a central location. Alert support is useful for managing systems that do not have an operator, managing systems where the operator is not skilled in problem management, and maintaining control of system resources and expenses.

On the AS/400 system, **alerts** are created based on messages that are sent to the local system operator. These messages are used to inform the operator of problems with hardware resources, such as local devices or controllers, communication lines, or remote controllers or devices. These messages can also report software errors detected by the system or application programs.

Any message with the alert option field (located in the message description) set to a value other than *NO can generate an alert. Alerts are generated from several types of messages:

- OS/400 messages defined as alerts.

  OS/400 support sends alerts for problems related to distributed relational database functions. For more information about distributed relational database related alerts, see "Alerts" on page 9-19.

- IBM-supplied messages where the value in the alert option field is specified as *YES by the Change Message Description (CHGMSGD) command. In this way, you can select the messages for which you want alerts sent to the distributed relational database administrator.

- Messages that you create and define as alerts, or that you create with the QALGENA application program interface (API).

In a distributed relational database, a system is part of a communications network, and local system messages cause alerts to be created and sent through the network to a central problem management site called a focal point. An alert **focal point** is a system in a network that receives and processes (logs, displays, and optionally sends) alerts. This allows you to centralize management of the network at the focal point.

A focal point's **sphere of control** is a collection of network node control points or systems within an APPN network from which the focal point system receives alerts. The focal point maintains connectivity with other network nodes in the sphere of control, accepts alerts received from systems in the sphere of control, and forwards alerts to a higher level focal point, if one exists.

An AS/400 system can be defined to be a primary focal point or a default focal point. As a **primary focal point**, the system receives alerts from all systems

explicitly defined in its sphere of control. As a **default focal point**, the system receives alerts from all systems that do not already have a primary focal point.

The AS/400 system also provides the capability to nest focal points. You can define a high level focal point, which accepts all of the alerts collected by lower level focal points.

See "Configuring Alert Support" on page 3-26 for an example configuration for alert handling.

5250 Display Station Pass-Through

Display station pass-through is a communications function that allows a user at a display station signed on to one AS/400 system to sign on and use a remote system to perform database administration tasks for the distributed database. Display station pass-through and distributed relational database support use the same APPC/APPN network configuration and no special considerations are necessary to configure display station pass-through with a distributed relational database. However, at the target system you must either configure virtual controllers and devices that are used, or set the QAUTOVRT system value to automatic configuration. See "Configuring for Pass-Through" on page 3-29 for more information.

Display station pass-through allows a user signed on to the source system to sign on to a target system. The **source system** is the system that issues the request to establish communications with another system and a **target system** is the system that receives the user's request to establish communications with the source system.

Figure 3-1 shows a typical pass-through environment. For example, the AS/400 system user working at the display station uses the display station pass-through function to access system MP110 *through* system MP000.

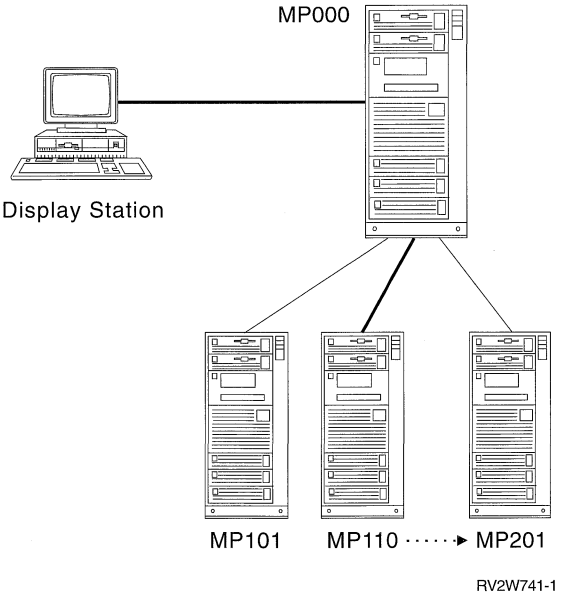


Figure 3-1. Pass-Through Environment

The system to which your display station is attached, either locally or remotely, is the source system (MP000). The system to which you are passing through is the target system (for example, MP110).

Both the target system and the source system contain information describing a display station. However, the target system has a virtual display that represents an actual display on the source system.

The network joining the source system and the target system can consist of many possible objects and configurations. Users can tailor their network to use the objects and configurations that best suit their application, and also use the system to help them set up their network.

For more information on display station pass-through and the remote workstation, see the *Remote Work Station Guide*.

Communications Line Support

Link level connectivity ties the communications protocols to the physical layer or hardware. Selection of the appropriate line requires advance planning and represents a significant long-term expense. Also, it is usually the most significant part of response time in a communications environment. Ethernet, ISDN, SDLC, token-ring, and X.25 line types are supported for distributed relational database on the AS/400 system.

Data link protocols serve two distinct network designs. Local area networks (LANs), like an Ethernet network or a token-ring network, offer very high data transmission rates, but are limited in the distance they can serve (usually within the same premises). Data link protocols such as ISDN, SDLC, and X.25 do not have geographic or distance restrictions like LANs, but they transmit data at slower rates.

In addition to these two general considerations, the data link protocol that is chosen for a particular environment can have an effect on the performance of applications in that environment. The following sections discuss some performance-related differences between various data link protocols supported on the AS/400 system for distributed relational database.

Ethernet

The Institute of Electrical and Electronics Engineers (IEEE) uses the Ethernet network as the model for the IEEE 802.3 standard. Ethernet (IEEE 802.3) is a local area network with a bus topology that uses the Carrier Sense Multiple Access with Collision Detection (CSMA/CD) access method. Most Ethernet networks have a data rate of 10 million bits per second (bps), a maximum length of 500 meters per segment, and a maximum network length of 2500 meters when segments are connected by repeater sets.

Bus configuration can affect the performance of Ethernet networks. Distributed relational database processing supports both the Ethernet Version 2 and IEEE 802.3 Ethernet standards, and the AS/400 system can operate on both frame types simultaneously. You should configure your Ethernet line for the best performance based on your specific needs. For example, if your job requires a Transmission Control Protocol Internet Protocol (TCP/IP) configuration for work not related to distributed relational database, you should be aware that TCP/IP uses only the IEEE 802.3 Ethernet standard. If the line is configured to operate on both standards,

then all incoming Ethernet Version 2 frames are also processed. This unnecessary processing can cause performance problems.

Large volumes of data or a large number of active stations on an Ethernet network can cause performance problems. Ethernet is more sensitive to large volumes of data than other LANs. Therefore if you anticipate your network configuration will be heavily used, another LAN, such as a token-ring network, may be a better choice.

The maximum frame size on an Ethernet line is 1496 bytes. An Ethernet line description can be configured for a smaller frame size by specifying a service access point (SAP) MAXFRAME value. The maximum frame size that can be used on Ethernet can be dependent on the frame size capability of other networks or bridges that the data must pass through.

When you specify *CALC for the MAXLENRU parameter of the Create Mode Description (CRTMODD) command, AS/400 support calculates the best correlation between SNA request/response unit (RU) size and frame size.

For more information about Ethernet networks, see the *Local Area Network Guide*.

Token-Ring Network

A token-ring network is a local area network that conforms to IEEE 802.5, which allows all members on the local area network to exchange data using a token-passing scheme. Data is sent in one direction throughout a specified number of locations. It uses the symbol of authority for control of the transmission line, called a token, to allow any sending station in the network to send data when the token arrives at that station. Token-ring networks can transmit data at very high rates, up to 16 million bps over limited distances (usually less than 1 mile). In comparison to an Ethernet LAN, a token-ring network is faster and less sensitive to heavy traffic.

Ring configuration, or the actual wiring of your network can have some bearing on performance. Two or more local area networks can be linked together by bridges that connect networks by storing and forwarding information in packets. Crossing these bridges can have a performance affect, depending on the amount of traffic on the bridge.

The AS/400 token-ring network supports a range of frame sizes up to 8156 bytes or up to 16 393 bytes on a 16 million bps line, depending on the token-ring adapter installed. Likewise, a token-ring supports a range of frame sizes up to 1994 bytes or up to 4060 bytes on a 4 million bps line, depending on the token-ring adapter. The frame size is specified with the MAXFRAME parameter in the line and controller descriptions. Larger frame sizes generally supply better performance, and the improved performance caused by larger frame sizes tends to be more noticeable with the token-ring network because of the faster media speed.

When you specify *CALC for the MAXLENRU parameter of the Create Mode Description (CRTMODD) command, AS/400 support calculates the best correlation between SNA request/response unit (RU) size and frame size.

For more information about the token-ring network, see the *Local Area Network Guide*.

SDLC

Synchronous data link control (SDLC) is a polling protocol that conforms to subsets of American National Standards Institute (ANSI) Advanced Data Communications (ADCCPI) and International Organization for Standardization (ISO) High-Level Data Link Control (HDLC). It is used for sending and receiving synchronous, code-transparent, serial-by-bit data. These transmissions can be made over duplex or half-duplex, switched or nonswitched lines, and the configuration can be point-to-point or multipoint. When connected to duplex lines, SDLC does not take advantage of the duplex nature of the line to receive and transmit simultaneously.

SDLC can be less expensive over a switched line, since it uses standard telephone lines and modems to transmit. It can also be a less expensive alternative with nonswitched lines than other protocols, depending on your network configuration. For small amounts of data and infrequent connections, a switched line is a good choice. When more data is exchanged and connections are more frequent, a non-switched line becomes increasingly attractive.

Performance considerations for SDLC typically include line speed, frame size, frequency of polling, and overhead. In any environment, line speed is the primary consideration; SDLC transmits at up to 64 000 bps.

The OS/400 support for SDLC can use a range of frame sizes up to 2057 bytes. Usually, the larger the frame size used, the better the performance. However, large frame sizes may not be as efficient in an environment where the line has a high probability for errors. If the frames need to be transmitted again, larger frames take longer.

When you specify *CALC for the MAXLENRU parameter of the Create Mode Description (CRTMODD) command, AS/400 support calculates the best correlation between SNA request/response unit (RU) size and frame size.

For more information about the SDLC network see the *OS/400* Communications Configuration Reference*.

X.25

The AS/400 system supports SNA data communications over packet-switched networks conforming to the International Telegraph and Telephone Consultative Committee (CCITT) Recommendation X.25. While X.25 supports both SNA and asynchronous communications, support for distributed relational database only works in the SNA environment.

X.25 is an alternative protocol to SDLC because it is a duplex protocol. A duplex protocol is capable of sending and receiving data at the same time. While a single APPC/APPN session does not take advantage of X.25 duplex support, the fact that you can have multiple APPC/APPN sessions on the same line can provide a significant performance advantage for those applications that can make use of this feature. Usually X.25 requires approximately 10% more system overhead than SDLC for similar application programs, but this increase could be offset by the duplex capability of the lines.

Packet size and window size can affect performance. The AS/400 system operates with a range of packet sizes of up to 1024 bytes for X.25. As with SDLC and local area networks, larger packet sizes provide better performance. In many cases,

however, the packet size is specified by the particular network to which the system is attached.

When you specify *CALC for the MAXLENRU parameter of the Create Mode Description (CRTMODD) command, AS/400 support calculates the best correlation between SNA request/response unit (RU) size and packet size.

The X.25 window size is similar to the maximum number of outstanding frames parameter used by SDLC and the token-ring networks. As with large packet sizes, a large window size may not work well for error-prone lines or networks.

For more information about the X.25 network, see the *X.25 Network Guide*.

ISDN Data Link Control

Integrated services digital network (ISDN) data link control (IDLC) is a duplex protocol used by the AS/400 system when the integrated ISDN support is used. (SDLC is used if an ISDN connection is through the 7820 ISDN Terminal Adapter.) ISDN transfers information end to end in a digital format that usually reduces errors on the line.

The connection can be either switched or nonswitched (permanent). When using IDLC, either system can transmit data without being polled. ISDN supports both data channels (B-channels) and a signalling channel (D-channel) on a line. The signalling channel (D-channel) is used to call a remote system and also to send information that the called system can use to identify the caller. Line speed on the data channels (B-channels) can be up to 64 000 bps and line speed on the signalling channel is 16 000 bps.

The switched or permanent connection refers to the connection between the two systems wanting to communicate. The AS/400 system is physically connected to the ISDN permanently, as with any common carrier line. For switched connections, connection list functions provide information on which incoming calls to accept and what is to be sent with outgoing calls.

The factors to be considered when deciding between a switched or permanent connection are usually the same for any protocol. With ISDN, the functions provided by the connection list and the calling party identification may make a switched connection more desirable.

The AS/400 support for IDLC allows frame sizes of 256 to 8196 bytes. The default frame size is 2048 bytes. The best data throughput is usually achieved with the largest frame size possible. When you specify *CALC for the MAXLENRU parameter of the Create Mode Description (CRTMODD) command, AS/400 support calculates the best correlation between SNA request/response unit (RU) size and frame size.

For more information about the ISDN network see the *ISDN Guide*.

Distributed Relational Database Network Considerations

Communications usage increases in a distributed relational database and there are some things you should consider for your communications network when you depend on it for database processing.

Because of the increased usage that comes with distributed relational database processing, you may want to increase the maximum number of sessions parameter (MAXSSN) and the maximum number of conversations parameter (MAXCNV) for the MODE description created for both the local and remote location.

In addition to increasing capacity through the MODE descriptions, you may want to consider increasing the line speed for various lines within the network or selecting a better quality line to improve performance of the network for your distributed relational database processing.

Another consideration for your distributed relational database network is the question of data accessibility and availability. The more critical a certain database is to daily or special enterprise operations, the more you need to consider how users can access that database. This means examining paths and alternative paths through the network to provide availability of the data as it is needed. More about this topic is discussed in Chapter 7, "Data Availability and Protection."

Line speed and how you configure your communications line can significantly affect network performance. However, it is important to ask a few questions about the nature of the information being transferred in relation to both line speed and type of use. For example:

- How much information must be moved?
- What is a typical transaction and unit of work for batch applications?
- What is a typical transaction and unit of work for an interactive application and how much data is sent and received for each transaction?
- How many application programs or users will be using the line at the same time?

The following sections highlight other network considerations, from a general communications standpoint and as they relate to distributed relational database support. For more information about network planning and performance considerations, see the *Communications Management Guide*.

Interactive and Batch Applications

A good practice for interactive applications is to make sure that the average line use does not exceed 50%. This ensures consistent response time for all users of the line. Line use for batch applications can usually approach 100% with no adverse performance effects. It is not a good practice to mix both interactive and batch application programs on the same line; careful planning is required to ensure that both interactive and batch transactions are given sufficient opportunity to use the line if you do choose to combine them.

Frame Sizes

The frame size determines the maximum amount of data transmitted over the line or network in one operation. All of the protocols discussed in this chapter provide the capability to use more than one frame size. Larger frame sizes are usually more efficient for both the line or network and the AS/400 system. Larger frame sizes result in less protocol cost and fewer changes on the line or network between transmitting and receiving.

Larger frame sizes may have little or no effect on certain types of interactive application programs. These programs have many small transactions that cannot wait to be assembled into a larger frame and cannot use a line or network as efficiently as application programs that can use larger frames. This difference in efficiency is less noticeable as the speed of the line increases.

Line Speed

For line speeds of 19 200 bits per second (bps) or less, most application programs can fully use the bandwidth (data capacity) of the line. The performance in this environment depends largely on the ability of the line to transfer the data.

For line speeds greater than 19 200 bps, some application programs cannot fully use the bandwidth of the line. The performance in this environment depends much more on the performance of the application itself. For example, a batch program that involves multiple actions on data before requesting the output operation may not deliver data to the line fast enough to take full advantage of the line speed. Multiple jobs such as those running at the same time can take advantage of the greater line speed.

Duplex versus Half-Duplex Networks

The most important factor when considering switched lines versus nonswitched lines is the modem turnaround time. This is the amount of time required for a station on the line to stop receiving data and begin transmitting data. Nonswitched lines generally have little modem turnaround time because they normally have four wires. These four wires can be used so that two wires are always ready to transmit data and the other two wires are always ready to receive. Switched lines can have a significant amount of modem turnaround time because they have only two wires.

Typical times for a switched line range from 0.1 second to 0.5 second, depending on the modem and the quality of the line connection. This is especially important for interactive application programs if a station alternates frequently between sending and receiving data, in some cases, many times for the same transaction. Large transfer application programs also see some degradation, because most line protocols require multiple transitions from sending to receiving during the course of a transfer to ensure data integrity.

SDLC Point-to-Point versus Multipoint

The nonswitched communications lines must be either point-to-point (only two devices are connected on a line) or multipoint (several devices are connected on a line). It is easier to control communications on a point-to-point line. It is usually cheaper, however, to transmit data using multipoint lines if there are several devices. With multipoint lines, the host system only needs one modem to communicate with several devices.

With both point-to-point and multipoint lines, one device is usually designated as the *primary* station that controls the line. The other devices are designated as *secondary* stations and they must wait until they are allowed to transmit by the primary device (host). The amount of time to wait depends on the number of stations on the line and the particular protocol used. Some protocols provide a way to assign higher priority to certain devices on the line.

For a distributed relational database, on an SDLC multipoint line, access to an AS from an AR can only occur from a secondary station AR to its primary station AS, or from a primary station AR to any secondary station acting as an AS.

Nonswitched versus Switched Lines

The communications lines must be either nonswitched (the line is always available) or switched (the connection is established by dialing, and only made available when needed). Switched lines can only be point-to-point (versus multipoint). Because time is required to establish the connection, a switched line should probably be used for short and infrequent transmission of data. Another consideration is cost of the line. Although a switched line is usually less expensive initially than a permanent line, it could cost more to maintain as frequency of use and duration increase.

Maximum Throughput

The maximum throughput on a line can depend on any of the following:

- Number of rows returned and the size of the row (the amount of data)
- Line quality (number of errors on the line)
- Modem considerations

For example, if a request to a database is producing data at 19 000 bps, moving from a 19 200 bps modem to a 64 000 bps modem does not increase the throughput of the line.

Modem Considerations

Aggregate line speeds (of a communications controller), use of the modem on nonswitched versus switched lines, duplex or half-duplex support capability, and diagnostic capability are considerations for choosing a modem. These considerations may also affect your choice of a communications line, so modem selection typically involves matching the needs of the network with the capabilities of the lines.

Performance Considerations for Alerts

If a large number of alerts are sent on a system or received from other systems, there may be a delay in the logging of the alerts.

If the network or the primary focal point system has failed, a primary focal point with a large sphere of control may require significant processing to try to establish sessions again. This is especially true if there is much link activation/deactivation occurring in the network. By using nested focal points, the size of any particular sphere of control can be reduced.

There should be no more than one default focal point in a network. A default focal point serves as a focal point for systems in the network that do not already have a primary focal point. Having more than one default focal point in the network does not provide any additional benefit.

Shared Lines, Controllers, and Devices

In an SNA environment, multiple controllers can be defined to share the same multipoint SDLC, X.25, IDLC, or LAN line. The AS/400 system can be defined to the network as multiple stations (or PUs) in the network by using multiple controller descriptions. If you specify a host controller on your network, the host controller can support different device types. This means you can support other device types in addition to the APPC/APPN device types you create for distributed relational database processing.

Multiple Devices Sharing a Controller: When communicating with a host system, any secondary device can be configured to share the controller. However, an APPC device that uses APPN functions cannot share a controller with an APPC device that is configured to not use the APPN functions. When you configure a host controller, you must specify whether APPN functions are to be used. Any APPC devices attached to that controller description must use the same value for the APPN parameter.

When communicating with a remote system using an APPC controller, all devices attached to that controller must use the same value for the APPN parameter (*YES or *NO) as specified in the controller description.

If you need to communicate on the same line to an APPC device that does not use APPN and one that does, you can configure multiple stations for that line (X.25, token-ring network, Ethernet, IDLC, or SDLC multipoint lines only). You must create two controller descriptions (APPC or host, depending on the remote system).

One controller description and the device descriptions attached to it must specify APPN(*YES); the other controller description must specify APPN(*NO). The device description attached to the controller must specify the same APPN value as the controller description.

Configuring Communications for a Distributed Relational Database

The following section briefly describes how to configure communications for a distributed relational database, how to set up alerts for systems in the distributed relational database network, and how to configure for pass-through functions. Each of these topics is complex; for example, communications can be configured with many variations. The following discussions use basic configuration examples to illustrate the steps needed to configure systems in a network and handle alerts at a central location.

Configuring a Communications Network

Configuring communications for a distributed relational database requires that the local and remote systems are defined in the network. Once the systems in the network are defined, you can use DDM functions or SNADS to distribute information throughout the network, establish your alert handling systems, use display station pass-through to connect to a target system from a workstation on a local system, and setup a relational database directory for systems in the distributed relational database network. A relational database directory associates communications configuration values with the names of relational databases in the distributed relational database network. See Chapter 5, "Setting Up an AS/400 Distributed Relational Database" for information about setting up the relational database directory.

Each AS/400 system in the network must be defined so that each system can identify itself and the remote systems in the network. To define a system in the network you must:

1. Define the network attributes.
2. Create the appropriate line descriptions.
3. Create a controller description.
4. Create a class-of-service description.
5. Create a mode description.
6. Create device descriptions automatically or manually.

Defining Network Attributes

To define the network attributes, use the Change Network Attributes (CHGNETA) command. The network attributes contain the local system name, the default local location name, the default control point name, the local network identifier, the network node type, and, if the machine is an end-node, the names of the network servers used by this AS/400 system.

Defining a Line Description

Create a line description to describe the physical line connection and the data link protocol to be used between the AS/400 system and the network. Use the following commands to create line descriptions:

- Create Line Description (Ethernet) (CRTLINETH)
- Create Line Description (ISDN) (CRTLINISDN)
- Create Line Description (SDLC) (CRTLINSDLC)
- Create Line Description (Token-ring) (CRTLINTRN)
- Create Line Description (X.25) (CRTLINX25)

Defining a Controller Description

A controller description describes the adjacent systems in the network. The use of APPN support is indicated by specifying APPN(*YES) when creating the controller description. Use the following commands to create controller descriptions:

- Create Controller Description (APPC) (CRTCTLAPPC)
- Create Controller Description (SNA HOST) (CRTCTLHOST)

If the AUTOCRTCTL parameter on a token-ring or Ethernet line description is set to *YES, then a controller description is automatically created when the system receives a session start request over the token-ring or Ethernet line.

Defining a Class-of-Service Description

A class-of-service description is used to select the communications routes (transmission groups) and assign transmission priority for sessions using APPN. Five class-of-service descriptions are supplied by the system:

#CONNECT The default class of service.

#BATCH A class of service for batch jobs.

#BATCHSC The same as #BATCH except a data link security of at least a packet switched network is required. In packet switched networks, data does not always follow the same path through the network.

#INTER A class of service tailored for interactive communications.

#INTERSC The same as #INTER except a data link security of at least a packet switched network is required.

Other class-of-service descriptions can be created using the Create Class-of-Service Description (CRTCOSD) command.

Defining a Mode Description

The mode description provides the session characteristics and number of sessions that are used to negotiate the allowed values between the local and remote location. The mode description also points to the class of service to use for the conversation. Five predefined modes are shipped with the system:

BLANK The default mode name specified in the network attributes when the system is shipped.

#BATCH A mode tailored for batch jobs.

#BATCHSC The same as #BATCH except the associated class-of-service description requires a data link security of at least a packet switched network. In packet switched networks, data does not always follow the same path through the network.

#INTER A mode tailored for interactive communications.

#INTERSC The same as #INTER except the associated class-of-service description requires a data link security of at least a packet switched network is required.

Other mode descriptions can be created using the Create Mode Description (CRTMODD) command.

Defining a Device Description

The device description describes the characteristics of the logical connection between the local and remote systems. Whether devices are created manually or automatically is dependent on the APPN support parameter on the Create Controller Description (CRTCTLAPPC) command.

If APPN(*YES) is specified, device descriptions are automatically created and attached to the appropriate controller description when the session is established. If APPN(*NO) is specified, the device description must be manually created using the Create Device Description (CRTDEVAPPC) command. If APPN(*NO) is specified on the controller description, the device description must also specify APPN(*NO) to indicate the device is not used in an APPN network.

The configuration examples in this chapter and other examples in other chapters of this book describe a network with APPN support, in which devices are automatically created by the AS/400 system. For information on how to configure devices using the CRTDEVAPPC command, see to the *APPC Programmer's Guide*.

Other Configuration Considerations

If additional local locations or special characteristics of remote locations for APPN are required, APPN location lists must be created. One local location name is the control point name specified in the network attributes. If additional locations are needed for the AS/400 system, an APPN local location list is required. Special characteristics of remote locations include whether the remote location is in a different network from the local location and security requirements. If special characteristics of remote locations exist, an APPN remote location list is required. APPN location lists can be created using the Create Configuration List (CRTCFGL)

command. See "Session Level and Location Security" on page 4-8 for more information about APPN configuration lists and security requirements.

The communication descriptions can be varied on (activated) by using the Vary Configuration (VRYCFG) command or the Work with Configuration Status (WRKCFGSTS) command. If the nonswitched line descriptions are varied on, the appropriate controllers and devices attached to that line are also varied on. The WRKCFGSTS command also gives the status of each connection. For more information about working with communication configuration status, see Chapter 6, "Distributed Relational Database Administration and Operation Tasks."

Notes:

1. The controller description is equivalent to the IBM Network Control Program and Virtual Telecommunications Access Method (NCP/VTAM) PU macros. The information in a controller description is found in the Extended Services Communication Manager Partner LU profile.
2. The device description is equivalent to the NCP/VTAM logical unit (LU) macro. The information in a device description is found in Extended Services Communications Manager Partner LU and LU profiles.
3. The mode description is equivalent to the NCP/VTAM mode tables. The information in a mode description is found in Extended Services Communications Manager Transmission Service Mode profile and Initial Session Limits profile.

The *OS/400* Communications Configuration Reference* and the *APPN Guide* contain more information about configuring for networking support and working with location lists.

APPN Configuration Example

To help illustrate a basic configuration example, consider the Spiffy Corporation network as illustrated in the following example.

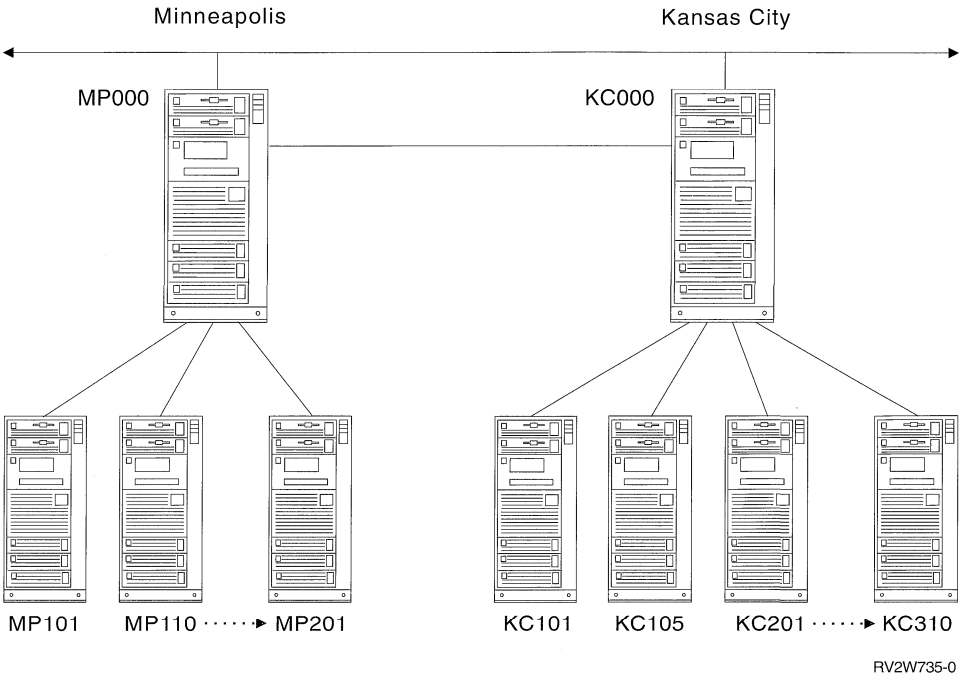


Figure 3-2. The Spiffy Corporation Network Organization

In this network organization, two of Spiffy Corporation's regional offices are the network nodes systems named MP000 and KC000. The MP000 system in Minneapolis and the KC000 system in Kansas City communicate with each other over an SDLC nonswitched line with an SDLC switched line as a backup line. The MP000 AS/400 system serves as a development and problem handling center for the KC000 system and the other regional network nodes.

The following example programs and explanations describe how to configure the Minneapolis and Kansas City AS/400 systems as network nodes in the network, and also shows how Minneapolis configures its network to one of its area dealerships. This example is intended to describe only a portion of the tasks needed to configure the network shown in Figure 3-2, and is not a complete configuration for that network.

Configuring Network Node MP000

The following example program shows the control language (CL) commands used to define the configuration for the system identified as MP000 (network node 1). The example shows the commands as used within a CL program; the configuration can also be performed using the configuration menus.

```
/*********************************************************************/
/*                                                                   */
/*  MODULE:  MP000                  LIBRARY:  PUBSCFGS               */
/*                                                                   */
/*  LANGUAGE:  CL                                                    */
/*                                                                   */
/*  FUNCTION:  CONFIGURES APPN NETWORK:                              */
/*                                                                   */
/*            THIS IS: MP000 TO KC000          (nonswitched)        */
/*                     MP000 TO KC000          (switched)           */
/*                     MP000 TO MP101 - MP299  (nonswitched)        */
/*                                                                   */
/*                                                                   */
/*                                                                   */
/*********************************************************************/
PGM
            /* Change network attributes for MP000 */          1
            CHGNETA    LCLNETID(APPN) LCLCPNAME(MP000) +
                         LCLLOCNAME(MP000) NODETYPE(*NETNODE)
/*********************************************************************/
/*                       MP000 to KC000   (nonswitched)              */
/*********************************************************************/
            /* Create nonswitched line description for MP000 to KC000*/
            CRTLINSDLC LIND(KC000L) RSRCNAME(LIN021)           2
            /* Create controller description for MP000 to KC000     */
            CRTCTLAPPC CTLD(KC000L) LINKTYPE(*SDLC)  +         3
                         LINE(KC000L) RMTNETID(APPN) +
                         RMTCPNAME(KC000) STNADR(01) +
                         NODETYPE(*NETNODE)
/*********************************************************************/
/*                       MP000 TO KC000 (switched)                   */
/*********************************************************************/
            /* Create switched line description for MP000 to KC000  */
            CRTLINSDLC LIND(KC000S) RSRCNAME(LIN022) +         4
                         CNN(*SWTPP) AUTOANS(*NO) STNADR(01)
            /* Create controller description for MP000 to KC000     */
            CRTCTLAPPC CTLD(KC000S) LINKTYPE(*SDLC) +          5
                         SWITCHED(*YES) SWTLINLST(KC000S) +
                         RMTNETID(APPN) RMTCPNAME(KC000) +
                         INLCNN(*DIAL) CNNNBR(8165551111 +
                         STNADR(01) TMSGRPNBR(3) NODETYPE(*NETNODE)

/*********************************************************************/

/*********************************************************************/
/*                       MP000 to MP101   (nonswitched)              */
/*********************************************************************/
            /* Create nonswitched line description for MP000 to KC000*/
            CRTLINSDLC LIND(MP101L) RSRCNAME(LIN031)           6
            /* Create controller description for MP000 to MP101     */
            CRTCTLAPPC CTLD(MP101L) LINKTYPE(*SDLC)  +
                         LINE(MP101L) RMTNETID(APPN) +
                         RMTCPNAME(MP101) STNADR(01) +
                         NODETYPE(*ENDNODE)
/*********************************************************************/
ENDPGM
```

1 Changing the Network Attributes (MP000)

The Change Network Attributes (CHGNETA) command is used to set the attributes for the system within the network. The following attributes are defined for the MP000 regional system, and these attributes apply to all connections in the network for this network node.

LCLNETID(APPN)
The name of the local network is APPN. The remote system (KC000 in the example program) must specify this name as the remote network identifier (RMTNETID) on the CRTCTLAPPC command. In this example, it defaults to the network attribute.

LCLCPNAME(MP000)
The name assigned to the Minneapolis regional system local control point is MP000. The remote systems specify this name as the remote control point name (RMTCPNAME) on the CRTCTLAPPC command.

LCLLOCNAME(MP000)
The default local location name is MP000. This name will be used for the device description that is created by the APPN support.

NODETYPE(*NETNODE)
The local system (MP000) is an APPN network node.

2 Creating the Line Description (MP000 to KC000, Nonswitched)

The line used in this example is an SDLC nonswitched line. The command used to create the line is CRTLINSDLC. The parameters specified are:

LIND(KC000L)
The name assigned to the line description is KC000L.

RSRCNAME(LIN021)
The physical communications port named LIN021 is defined.

3 Creating the Controller Description (MP000 to KC000, Nonswitched)

Because this is an APPN environment (AS/400 system to AS/400 system), the controller is an APPC controller, and the CRTCTLAPPC command is used to define the attributes of the controller. The following attributes are defined by the example command:

CTLD(KC000L)
The name assigned to the controller description is KC000L.

LINKTYPE(*SDLC)
Because this controller is attached through an SDLC communications line, the value specified is *SDLC. This value must correspond to the type of line defined by a Create Line Description command (CRTLINxxx).

LINE(KC000L)
The name of the line description to which this controller is attached is KC000L. This value must match a name specified by the LIND parameter in a line description.

RMTNETID(APPN)
The name of the network in which the remote control point resides is APPN.

RMTCPNAME(KC000)
The remote control-point name is KC000. The name specified here must match the name specified at the remote system for the local control-point name. In the example, the name is specified at the remote system (KC000) by the LCLCPNAME parameter on the Change Network Attributes (CHGNETA) command.

STNADR(01)
The address assigned to the remote controller is hex 01.

NODETYPE(*NETNODE)
The remote system (KC000) is an APPN network node.

4 Creating the Line Description (MP000 to KC000, Switched)

The line used in this example is an SDLC switched line. The command used to create the line is CRTLINSDLC. The parameters specified are:

LIND(KC000S)
The name assigned to the line description is KC000S.

RSRCNAME(LIN022)
The physical communications port named LIN022 is defined.

CNN(*SWTPP)
This is a switched line connection.

AUTOANS(*NO)
This system will not automatically answer an incoming call.

STNADR(01)
The address assigned to the local system is hex 01.

5 Creating the Controller Description (MP000 to KC000, Switched)

Because this is an APPN environment (AS/400 system to AS/400 system), the controller is an APPC controller, and the CRTCTLAPPC command is used to define the attributes of the controller. The following attributes are defined by the example command:

CTLD(KC000S)
The name assigned to the controller description is KC000S.

LINKTYPE(*SDLC)
Because this controller is attached through an SDLC communications line, the value specified is *SDLC. This value must correspond to the type of line defined by a Create Line Description command (CRTLINxxx).

SWITCHED(*YES)
This controller is attached to a switched SDLC line.

SWTLINLST(KC000S)
The name of the line description (for switched lines) to which this controller can be attached is KC000S. In the example, there is only one line (KC000S). This value must match a name specified by the LIND parameter in a line description.

RMTNETID(APPN)
The name of the network in which the remote control point resides is APPN.

RMTCPNAME(KC000)
The remote control-point name is KC000. The name specified here must match the name specified at the remote system for the local control-point name. In the example, the name is specified at the remote system by the LCLCPNAME parameter on the CHGNETA (Change Network Attributes) command.

INLCNN(*DIAL)
The initial connection is made by the AS/400 system either answering an incoming call or placing a call.

CNNNBR(8165551111)
The connection (telephone) number for the remote Kansas City controller is 8165551111.

STNADR(01)
The address assigned to the remote Kansas City controller is hex 01.

TMSGRPNBR(3)
The value (3) is to be used by the APPN support for transmission group negotiation with the remote system.

The remote system must specify the same value for the transmission group.

NODETYPE(*NETNODE)
The remote system (KC000) is an APPN network node.

6 Creating a Line and Controller for MP101

This portion of the example shows a line and controller configuration for MP000 to MP101, a dealership end node. A similar configuration must be made from MP000 to each of its dealership end nodes. Also, to complete the configuration for Minneapolis, each of the dealerships must use configuration commands or a program similar to this one to create lines and controllers for each system that they will communicate with.

Likewise, to complete the network configuration shown in Figure 3-2 on page 3-16, the KC000 system must configure to each of its dealership end nodes, and each end node must configure a line and controller to communicate with the KC000 system.

These connections are not shown in the example.

Configuring Network Node KC000

The following example program shows the CL commands used to define the configuration for the regional system identified as KC000. The example shows these commands as used within a CL program; the configuration can also be performed using the configuration menus.

```
/*****************************************************************/
/*                                                               */
/*  MODULE:  KC000                    LIBRARY:  PUBSCFGS         */
/*                                                               */
/*  LANGUAGE:  CL                                                */
/*                                                               */
/*  FUNCTION:  CONFIGURES APPN NETWORK:                          */
/*                                                               */
/*             THIS IS: KC000 TO MP000     (nonswitched)         */
/*                      KC000 TO MP000     (switched)            */
/*                                                               */
/*                                                               */
/*****************************************************************/
PGM
            /* Change network attributes for KC000               */
            CHGNETA    LCLNETID(APPN) LCLCPNAME(KC000) +      7
                         LCLLOCNAME(KC000) NODETYPE(*NETNODE)
/*****************************************************************/
/*                           KC000 TO MP000  (nonswitched)       */
/*****************************************************************/
            /* Create line description for KC000 to MP000        */
            CRTLINSDLC LIND(MP000L) RSRCNAME(LIN022)          8
            /* Create controller description for KC000 to MP000  */
            CRTCTLAPPC CTLD(MP000) LINKTYPE(*SDLC) +          9
                         LINE(MP000L) RMTNETID(APPN) +
                         RMTCPNAME(MP000) STNADR(01) +
                         NODETYPE(*NETNODE)
/*****************************************************************/
/*                         KC000 TO MP000   (switched)           */
/*****************************************************************/
            /* Create switched line description for KC000 to MP000S */
            CRTLINSDLC LIND(MP000S) RSRCNAME(LIN031) +       10
                         CNN(*SWTPP) AUTOANS(*NO) STNADR(01)
            /* Create controller description for KC000 to MP000  */
            CRTCTLAPPC CTLD(MP000S) LINKTYPE(*SDLC) +        11
                         SWITCHED(*YES) SWTLINLST(MP000S) +
                         RMTNETID(APPN) RMTCPNAME(MP000) +
                         INLCNN(*ANS) CNNNBR(6125551111) +
                         STNADR(01) TMSGRPNBR(3) NODETYPE(*NETNODE)
ENDPGM
```

7 Changing the Network Attributes (KC000)

The Change Network Attributes (CHGNETA) command is used to set the attributes for the system within the network. The following attributes are defined for the regional system named KC000, and these attributes apply to all connections in the network for this network node:

LCLNETID(APPN)
The name of the local network is APPN. The remote systems (the Minneapolis network node in this example) must specify this name as the remote network identifier (RMTNETID) on the CRTCTLAPPC command.

LCLCPNAME(KC000)
The name assigned to the local control point is KC000. The remote system specifies this name as the remote control point name (RMTCPNAME) on the CRTCTLAPPC command.

LCLLOCNAME(KC000)
The default local location name is KC000. This name will be used for the device description that is created by the APPN support.

NODETYPE(*NETNODE)
The local system (KC000) is an APPN network node.

8 Creating the Line Description (KC000 to MP000, Nonswitched)

The line used in this example is an SDLC nonswitched line. The command used to create the line is CRTLINSDLC. The parameters specified are:

LIND(MP000L)
The name assigned to the line description is MP000L.

RSRCNAME(LIN022)
The physical communications port named LIN022 is defined.

9 Creating the Controller Description (KC000 to MP000, Nonswitched)

Because this is an APPN environment (AS/400 system to AS/400 system), the controller is an APPC controller, and the CRTCTLAPPC command is used to define the attributes of the controller. The following attributes are defined by the example command:

CTLD(MP000L)
The name assigned to the controller description is MP000L.

LINKTYPE(*SDLC)
Because this controller is attached through an SDLC communications line, the value specified is *SDLC. This value must correspond to the type of line defined by a Create Line Description command (CRTLINxxx).

LINE(MP000L)
The name of the line description to which this controller is attached is MP000L. This value must match a name specified by the LIND parameter in a line description.

RMTNETID(APPN)
The name of the network in which the remote system resides is APPN.

RMTCPNAME(MP000)
The remote control-point name is MP000. The name specified here must match the name specified at the remote system for the local control-point name. In the example, the name is specified at the Minneapolis region remote system (MP000) by the LCLCPNAME parameter on the Change Network Attributes (CHGNETA) command.

STNADR(01)
The address assigned to the remote controller is hex 01.

NODETYPE(*NETNODE)
The remote system (MP000) is an APPN network node.

10 Creating the Line Description (KC000 to MP000, Switched)

The line used in this example is an SDLC switched line. The command used to create the line is CRTLINSDLC. The parameters specified are:

LIND(MP000S)
The name assigned to the line description is MP000S.

RSRCNAME(LIN031)
The physical communications port named LIN031 is defined.

CNN(*SWTPP)
This is a switched line connection.

AUTOANS(*NO)
This system will not automatically answer an incoming call.

STNADR(01)
The address assigned to the local system is hex 01.

11 Creating the Controller Description (KC000 to MP000, Switched)

Because this is an APPN environment (AS/400 system to AS/400 system), the controller is an APPC controller, and the CRTCTLAPPC command is used to define the attributes of the controller. The following attributes are defined by the example command:

CTLD(MP000S)
The name assigned to the controller description is MP000S.

LINKTYPE(*SDLC)
Because this controller is attached through an SDLC communications line, the value specified is *SDLC. This value must correspond to the type of line defined by a Create Line Description command (CRTLINxxx).

SWITCHED(*YES)
This controller is attached to a switched SDLC line.

SWTLINLST(MP000S)
The name of the line description (for switched lines) to which this controller can be attached is MP000S. In the example, there is only one line (MP000). This value must match a name specified by the LIND parameter in a line description.

RMTNETID(APPN)
The name of the network in which the remote control point resides is APPN.

RMTCPNAME(MP000)
The remote control-point name is MP000. The name specified here must match the name specified at the remote regional system for the local control-point name. In the example, the name is specified at the remote Minneapolis regional system (MP000) by the LCLCPNAME parameter on the Change Network Attributes (CHGNETA) command.

INLCNN(*ANS)
The initial connection is made by the AS/400 system answering an incoming call.

CNNNBR(6125551111)
The connection (telephone) number for the remote Minneapolis controller is 6125551111.

STNADR(01)
The address assigned to the remote Minneapolis controller is hex 01.

TMSGRPNBR(3)
The value (3) to be used by the APPN support for transmission group negotiation with the remote system. The remote system must specify the same value for the transmission group.

NODETYPE(*NETNODE)
The remote system (MP000) is an APPN network node.

Configuring SNADS Support

If you want to use SNA distribution services (SNADS) to move objects from one system to another in your network, you must configure SNADS by setting up distribution queues and a routing table for each system in your network. For more information on configuring distribution services, see the *Distribution Services Network Guide*.

The following steps show the basic operations you need to perform to set up SNADS.

1. Enter the Configure Distribution Services (CFGDSTSRV) command to see the following display:

```
                    Configure Distribution Services

Type choice, press Enter.

  Type of distribution services
    information to configure  . . .    1        1=Distribution queues
                                                2=Routing table
                                                3=Secondary system name table
```

2. Distribution queues must be set up first. Select option 1 (Distribution queues) to show the Configure Distribution Queues display.
3. Select F6 (Add distribution queue) from the Configure Distribution Queues display to show the Add Distribution Queue display.

```
                          Add Distribution Queue          Page 1 of 2

Type choices, press Enter.

  Queue  . . . . . .  MP000                 Name
  Queue type . . . .  *SNADS          *SNADS, *RPDS, *DLS
  Remote location name MP000                Name
  Mode . . . . . .    *NETATR         Name, *NETATR
  Remote net ID  . .  *LOC            Name, *LOC, *NONE
  Local location name *LOC            Name, *LOC
  Normal priority:
    Send time:
      From/To  . . . . . .   __ : __ __ : __   00:00-23:59
      Force  . . . . . .     __ : __          00:00-23:59
    Send depth . . . . . .   1                1-999, blank
  High priority:
    Send time:
      From/To  . . . . . .   __ : __ __ : __   00:00-23:59
      Force  . . . . . .     __ : __          00:00-23:59
    Send depth . . . . . .   1                1-999, blank
```

In this example, a distribution queue is set up for the MP000 system on the MP101 system (named *LOC for the local location name). A distribution queue should be configured for each system to which the MP101 system is directly connected if distribution services to those remote systems are desired. (In the APPN configuration example, the MP101 system is an end node that is directly connected to the MP000 system.)

4. The routing tables must be configured next. Select option 2 (Routing table) on the Configure Distribution Services display to show the Configure Routing Table command display.

5. Select F6 (Add routing table entry) from the Configure Routing Table display to show the Add Routing Table Entry display.

```
                          Add Routing Table Entry

Type choices, press Enter. (At least one queue name is required.)

  System name/Group  . .  MP000    MP000
  Description  . . . . .  Regional Center in Minneapolis
  Service level:
    Fast:
      Queue name . . . .  MP000              Distribution queue name
      Maximum hops . . .  *DFT                        Number of hops, *DFT
    Status:
      Queue name . . . .  MP000
      Maximum hops . . .  *DFT
    Data high:
      Queue name . . . .  MP000
      Maximum hops . . .  *DFT
    Data low:
      Queue name . . . .  MP000
      Maximum hops . . .  *DFT
```

In this example, a routing table is set up for the MP000 system on the MP101 system. The remote location name of MP000, specified when the APPN example network was configured, is used as the system name in the routing table. It is also the queue name established for this system from the previous

step. The routing table system name must match the name specified for the system name (SYSNAME) parameter on the CHGNETA command for the remote system. However, using the same name for network locations, queue names and routing table system names makes identification of systems in large networks easier.

6. When the routing tables are set up, you must add users and systems to your SNADS distribution directory. Use the Work with Directory (WRKDIR) command to show the following display:

```
                              Work with Directory
Type options, press Enter.
  1=Add   2=Change   4=Remove   5=Display details   6=Print details
  7=Assign different ID to description   9=Add another description
Opt   User ID    Address    Description

__   _______    _______    ___________
__    *ANY       MP000      All users and AS/400s on MP000
__    *ANY       MP101      All users unknown to MP101
__    MPDBA      MP101      MP000 Distributed Database Administrator
__    MPSUP      MP101      MP000 Distributed Database Support
__    MPCLRK1    MP101      Service Clerk 1
__    MPCLRK2    MP101      Service Clerk 2
```

In this example, the directory on the MP101 system contains an entry for any users on the MP000 system, an entry for any users that are unknown to the MP101 system, and entries for users on the MP101 system. Providing an entry for unknown users for your system allows you to handle a distribution to a user not in your directory and prevents an error or communications performance problem.

This simple example shows only how the MP101 system could configure to send and receive from the MP000 system. The MP000 system must configure for SNADS distribution to MP101, the other end nodes, and any other network nodes to which it is directly connected.

Configuring Alert Support

Depending on what role your system assumes in the network, several actions help establish alert support for your system.

In an APPN network, an end node sends its alerts either to its serving network node or to another system as specified by the alerts controller name (ALRCTLD) parameter of the Change Network Attributes (CHGNETA) command. When your system is an end node in the network and you turn on the alert status (ALRSTS) parameter of the CHGNETA command, alerts are forwarded to a serving network node.

You can define your system as a default focal point using the alert default focal point (ALRDFTFP) parameter of the CHGNETA command. When your system is defined to be a default focal point, the AS/400 system automatically adds network node control points to the sphere of control using the APPN network topology database. When the AS/400 system detects that a network node system has entered the network, the system sends management services capabilities to the new control point so that the control point sends alerts to your system (if no other focal point is specified for the new network node system). The alert status (ALRSTS) parameter of the CHGNETA command should be turned off so your system does not forward alerts because it is the default focal point.

You can define your system as a primary focal point using the alert primary focal point (ALRPRIFP) parameter of the CHGNETA command. When your system is defined to be a primary focal point, you must explicitly define the control points that are to be in your sphere of control. This set of control points is defined using the Work with Sphere of Control (WRKSOC) command.

The WRKSOC command allows you to add network node control point systems to the sphere of control and to delete existing control points.

```
                    Work with Sphere of Control (SOC)
                                                          System:   MP000
Position to . . . . . .  _______  Control Point
  Network ID . . . . . .  _______

Type options, press Enter.
  1=Add   4=Remove

     Control
Opt  Point     Network ID  Current Status
__   _______    *NETATR
__   CH000      APPN        Delete pending
__   KC000      APPN        Active - in sphere of control
__   SL000      APPN        Add pending - in sphere of control
__   NY000      APPN        Active - in sphere of control
```

Select option 1 (Add) on the Work with Sphere of Control (SOC) display, or use the Add Sphere of Control Entry (ADDSOCE) command to add a system to your sphere of control. To add a system to the sphere of control, type the control point name and network ID of the new system.

Select option 4 (Remove) from the Work with Sphere of Control (SOC) display, or use the Remove Sphere of Control Entry (RMVSOCE) command to delete systems from the alert sphere of control. The systems are specified by network ID and control point name.

Unless a default focal point is established for your network, a control point in the sphere of control should not be removed from the sphere of control until another focal point has started focal point services to that system.

The Display Sphere of Control Status (DSPSOCSTS) command shows the current status of all systems in your sphere of control. This includes both systems that you have defined using the WRKSOC command, if your system is defined to be a primary focal point, and systems that the AS/400 system has added for you, if your system is defined to be a default focal point.

Example Configuration for Alert Support

You can establish alert support for the two network nodes MP000 and KC000 and their associated end nodes as shown in the following figure:

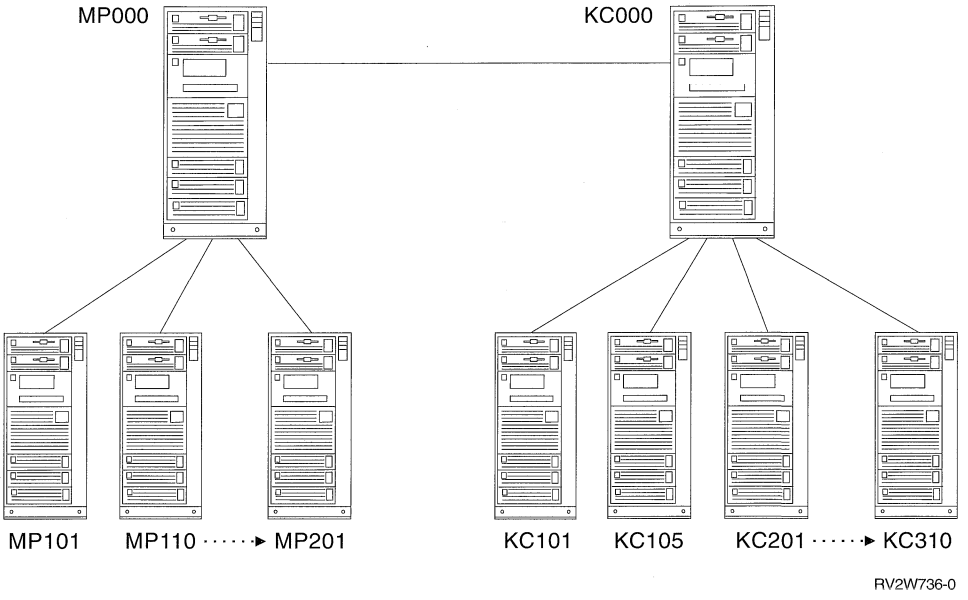


Figure 3-3. Spiffy Corporation Example Network Configuration

This example configuration shows how to:

- Begin creating alerts at a network node
- Set up a network node to forward alerts to a primary focal point
- Begin creating alerts at an end node

The CL command examples that follow are used to establish the system named MP000 as a primary focal point for alerts handling. While this system may serve as the primary focal point for several systems, this example only illustrates how one other network node (KC000) is configured to forward alerts to the MP000 system and how MP000 is set up to be the primary focal point that does not pass the alerts on to another system. To configure alerts for this example, the database administrator would:

1. Create alerts at a network node.
2. Define a network node as the primary focal point.
3. Add network nodes to the primary focal point's sphere of control.
4. Create alerts at end node systems.

Create Alerts at a Network Node

The system configured as KC000 begins to create alerts and forward them to a primary focal point when the database administrator turns on the local alerts parameter (ALRSTS) on the Change Network Attributes (CHGNETA) command as shown in the example below. This system is also set up to log alerts it creates locally.

```
CHGNETA    ALRSTS(*ON) ALRLOGSTS(*LOCAL)
```

Because a primary focal point is not active and has not included KC000 in its sphere of control, any alerts created at the KC000 system are logged at the KC000 system and not forwarded to another system yet.

Define A Primary Focal Point

To define a network node as a primary focal point, the database administrator needs to specify *YES for the alert primary focal point (ALRPRIFP) parameter on the Change Network Attributes (CHGNETA) command for the selected system.

This system creates alerts locally by specifying *ON for the alert status (ALRSTS) parameter. Also, this system logs alerts created locally and alerts received from other systems when *ALL is specified for the alert logging (ALRLOGSTS) parameter on the CHGNETA command.

An example is shown below.

```
CHGNETA    ALRPRIFP(*YES) ALRSTS(*ON) ALRLOGSTS(*ALL)
```

Update the Primary Focal Point's Sphere of Control

The system named MP000 does not receive alerts from other systems until the database administrator adds the names of systems that should forward alerts to the MP000 system's sphere of control.

The Work with Sphere of Control (WRKSOC) command identifies the network nodes from which MP000 receives alerts. In the example below, the KC000 system is included in the MP000 system's sphere of control using the Add Sphere of Control Entry (ADDSOCE) command. The network identifier is specified as *NETATR, and the control point name for KC000 is specified for the entry.

```
ADDSOCE    ENTRY((*NETATR KC000))
```

Create Alerts for End Nodes

End nodes may participate in an APPN network by using the services of an attached network node (the serving network node). In the above figure, KC000 is the serving network node for KC105.

The end node must begin creating alerts by specifying ALRSTS(*ON) for the Change Network Attributes (CHGNETA) command. However, after its network node is set up to forward alerts, the alerts sent by KC105 are forwarded by KC000 to the focal point at MP000 without a database administrator having to specify how KC105 system alerts are handled.

Configuring for Pass-Through

Virtual devices are used by the target system to direct output to devices on the source system. You can allow the system to automatically configure these devices, you can configure them yourself, or you can do both. The following discussion focuses on automatic configuration. For information about manual configuration, see the *Remote Work Station Guide*.

Automatic Configuration

When the source system requests pass-through support to automatically configure a device, the target system examines each of the virtual devices on the virtual controller named QPACTL*nn*. If a device is available, the target system program configures that device to match your physical device. If the target pass-through program cannot find an available virtual device, it checks the QAUTOVRT system value. If creating another device makes the number of devices attached to the QPACTL*nn* virtual controllers greater than the QAUTOVRT system value, another device is not created. Instead, the program ends the pass-through session. If cre-

ating another device keeps the number of devices less than or equal to the QAUTOVRT value, a device is created.

To allow automatic configuration, use the Change System Value (CHGSYSVAL) command to set the QAUTOVRT system value to the maximum number of devices that you will allow automatically configured by the program. Consider the following when using automatic configuration of virtual devices by pass-through:

- Virtual devices that are automatically configured are owned by the user profile running the pass-through programs on the target system. Therefore, when automatic sign-on is used, the virtual devices that are created are owned by the user specified in the RMTUSER parameter of the STRPASTHR command.
- Pass-through does not delete virtual devices, even if the number of these devices attached to automatically configured virtual controllers exceeds the QAUTOVRT limit.

  If you specify a value less than the number of devices attached to QPACTL*nn* controllers and you want the extra devices deleted, you must manually delete any virtual devices that exceed the QAUTOVRT limit.

  If you do not specify a virtual controller or virtual devices when you issue the STRPASTHR command, the pass-through function uses the special virtual controller named QPACTL00 to request automatic configuration support from the target system.
- If the target system does not support automatic configuration, a user requesting automatic configuration can still pass through. However, the target system operator must create a virtual controller named QPACTL00 on the target system, and the QPACTL00 controller must have an appropriate virtual device attached.

Security Considerations for Virtual Controllers and Devices

If you allow automatic configuration of virtual devices, it is easier for users to attempt to break in by using pass-through. Without automatic configuration, a user attempting to break in has a limited number of attempts at each virtual device. This limit is defined by the security officer using the QMAXSIGN system value and by the number of devices available. With automatic configuration active, the actual limit increases because the system limit for sign-on operations that are not valid is multiplied by the number of virtual devices that can be created by the automatic configuration support defined by the QAUTOVRT system value.

Chapter 4. Security for an AS/400 Distributed Relational Database

A distributed relational database administrator is faced with two security issues to resolve:

- System to system protection
- Identification of users at remote sites

When two or more systems are set up to access each other's databases, it is important to make sure that the other side of the communications line is the intended location and not an intruder. The AS/400 system use of advanced program-to-program communications (APPC) and advanced peer-to-peer networking (APPN) communications configuration capabilities provides options for you to do this necessary network level security.

The second concern for the distributed relational database administrator is that data security is maintained by the system storing the data. In a distributed relational database, the user has to be properly authorized to have access to the database (according to the security level of the system) whether the database is local or remote. Distributed relational database network users must be properly identified with a user ID on the application server (AS) for any jobs they run on the AS.

This chapter describes general AS/400 security concepts and discusses security related to communications and distributed relational database requirements. In remaining security discussions, the term *user* also includes remote users starting communications jobs.

AS/400 System Security Concepts

Security on an AS/400 system is controlled by the interaction of the system security level, user profiles, and authorities granted to users for objects on the system.

For more information about AS/400 security topics see the *Security Reference* manual.

System Security Levels

The AS/400 system supports five security levels that determine what authority users are given by default and the level of enforcement performed by the system. The levels of security are:

10 Requires only a user ID to sign on. After signing on, the user has access to all objects on the system.

20 Requires a user ID and password to sign on. After signing on, the user has access to all objects on the system.

30 Requires a user ID and password to sign on. After signing on, the user must have authority to objects before they can be used.

40 Requires a user ID and password to sign on. After signing on, the user must have authority to objects before they can be used. System integrity checking is active. All attempts to use undocumented and unsupported interfaces fail.

50 Requires a user ID and password to sign on. After signing on, the user must have authority to use objects and system resources. System integrity and security checking of the QTEMP and *USRxxx objects are enforced. All attempts to pass unsupported parameter values to supported interfaces fail.

The level of security for the system is selected by changing the system value QSECURITY and then doing an initial program load (IPL) of the system.

User Profiles

A **user profile** is an AS/400 object with a unique name that is used to identify a user to the system and verify the user's authorities on the system. User profiles tell the system who can sign on and what functions the user can perform on the system resources after signing on.

User profiles contain information that tailor the way a user operates on the system by using such things as special authority, initial program, and menu presentation. When identifying a user on the system, you can specify a user class in the user profile. A **user class** classifies a user as one of the following types of system users:

- Security officer (*SECOFR)
- Security administrator (*SECADM)
- Programmer (*PGMR)
- System operator (*SYSOPR)
- User (*USER)

For example, a distributed relational database administrator might need a user class of *SECOFR on systems in the network to allow the administrator to perform administration tasks and correct problems that arise. However, an end user would have a user class of *USER, which permits fewer capabilities than other user classes.

User Profiles in a Distributed Relational Database

In a distributed relational database, the application server (AS) controls access to its objects. If the system security level is set to 20 or above, a user profile should be created on the AS for every application requester (AR) user to allow communications access to the AS. When the security level is 30 or above at the AS, the user profile is also used to specify authorities the AR user needs to objects on the AS.

Pass-through from the source system to a target system also requires that a user ID profile exist on the target system. These user profiles should only have authority appropriate to the functions being performed by each user.

A user profile is set up using the Create User Profile (CRTUSRPRF) command. The following example shows the command used to create a user profile for a distributed relational database administrator.

```
CRTUSRPRF  USRPRF(KCDBA)  PASSWORD(password)
           JOBD(SPIFFY/KCDBA) USRCLS(*SECOFR)
           TEXT('User profile for Database Administrator')
```

Example User Profile

An example of a user profile for a distributed relational database administrator might look like the following:

```
                        Display User Profile - Basic

 User profile . . . . . . . . . . . . . . . . :   KCDBA

 Previous sign-on . . . . . . . . . . . . . . :   03/29/92  07:59:04
 Sign-on attempts not valid . . . . . . . . . :   0
 Status . . . . . . . . . . . . . . . . . . . :   *ENABLED
 Date password last changed . . . . . . . . . :   03/29/92
 Password expiration interval . . . . . . . . :   *SYSVAL
 Set password to expired  . . . . . . . . . . :   *NO
 User class . . . . . . . . . . . . . . . . . :   *SECOFR
 Special authority  . . . . . . . . . . . . . :   *ALLOBJ
                                                  *JOBCTL
                                                  *SAVSYS
                                                  *SECADM
                                                  *SERVICE
                                                  *SPLCTL
 Group profile  . . . . . . . . . . . . . . . :   QSECOFR
 Owner  . . . . . . . . . . . . . . . . . . . :   *USRPRF
 Group authority  . . . . . . . . . . . . . . :   *NONE
 Assistance level . . . . . . . . . . . . . . :   *SYSVAL
 Current library  . . . . . . . . . . . . . . :   SPIFFY
 Initial menu . . . . . . . . . . . . . . . . :   MAIN
   Library  . . . . . . . . . . . . . . . . . :     *LIBL
 Initial program  . . . . . . . . . . . . . . :   INLPGM
   Library  . . . . . . . . . . . . . . . . . :     KCDBA
 Limit capabilities . . . . . . . . . . . . . :   *NO
 Text . . . . . . . . . . . . . . . . . . . . :   KC000 DB Admin
 Display sign-on information  . . . . . . . . :   *SYSVAL
 Limit device sessions  . . . . . . . . . . . :   *SYSVAL
 Keyboard buffering . . . . . . . . . . . . . :   *SYSVAL
 Maximum storage allowed  . . . . . . . . . . :   *NOMAX
   Storage used . . . . . . . . . . . . . . . :     3078
 Highest scheduling priority  . . . . . . . . :   3
 Job description  . . . . . . . . . . . . . . :   QDFTJOBD
   Library  . . . . . . . . . . . . . . . . . :     QGPL
 Accounting code  . . . . . . . . . . . . . . :
 Message queue  . . . . . . . . . . . . . . . :   KCDBA
   Library  . . . . . . . . . . . . . . . . . :     QUSRSYS
 Message queue delivery . . . . . . . . . . . :   *NOTIFY
 Message queue severity . . . . . . . . . . . :   00
 Output queue . . . . . . . . . . . . . . . . :   KCDBA
   Library  . . . . . . . . . . . . . . . . . :     QUSRSYS
 Printer device . . . . . . . . . . . . . . . :   *WRKSTN
 Special environment  . . . . . . . . . . . . :   *SYSVAL
 Attention program  . . . . . . . . . . . . . :   QUSCMDLN
   Library  . . . . . . . . . . . . . . . . . :     *LIBL
 Language ID  . . . . . . . . . . . . . . . . :   ENU
 Country ID . . . . . . . . . . . . . . . . . :   US
 Coded Character Set ID . . . . . . . . . . . :   00037
 User options . . . . . . . . . . . . . . . . :   *NONE
```

Types of Authorities

When the system value for security is set to level 30 or higher, users must have authority to an object to be able to access the object. Depending on user class and special authorities granted, a user may have authority to certain objects without specifically having authority granted. There are two major types of system authority: special authority and specific authority.

Special authority allows a user to perform system control operations and determines what menu options the user can select. Special authority is defined in the user profile by specifying the class of user or granting special authorities to the user.

Specific authority determines how a user can use a specific object. This includes object authority and data authority. Specific authority can be defined two ways: by combining one or more object authorities with one or more data authorities, or by specifying one of the authorities defined by the system that is a combination of object authorities and data authorities. The following sections describe the object and data authorities you can specify.

Object Authority

Object authority allows you to perform operations on an object such as move or rename the object, control the object's existence, and use the data contained in the object. You can specify the following object authorities:

Object Operational (*OBJOPR) authority allows the user to look at the description of an object and use the object as determined by the data authorities that the user has to the object.

Object Management (*OBJMGT) authority allows the user to specify the security for the object, move or rename the object, and add members to database files.

Object Existence (*OBJEXIST) authority allows the user to control the object's existence and ownership. This authority is necessary for users who want to delete the object, free storage of the object, perform save and restore operations for the object, or transfer ownership of an object. (A user who has save system (*SAVSYS) special authority does not need object existence authority to perform save and restore operations.)

Authorization List Management (*AUTLMGT) authority allows the user to add and remove users and their authorities on an authorization list. If you have authorization list management authority you can remove a user profile name from the list only if you have the same authorities as the user profile name being removed. You can add, change, or remove authority only if you have the same authorities being added, changed, or removed.

Data Authority

Data authority allows you to access the contents of an object. For example, you can read, add, update, and delete entries from an object. You can specify the following data authorities:

Read (*READ) authority allows the user to display the contents of an object or run a program.

Add (*ADD) authority allows the user to add entries to an object, for example, adding job entries to a job queue or adding rows to a table.

Update (*UPD) authority allows the user to change the entries in an object.

Delete (*DLT) authority allows the user to remove entries from an object, for example, removing messages from a message queue or rows from a table.

Subset of Authorities Defined by the System

The system has defined four authorities that allow the user to select a subset of object authorities and data authorities for an object. This subset combines one or more object authorities with one or more data authorities (see Figure 4-1). Authorization list management authority can be specified with one of the system-defined authorities only for an authorization list object. The only exception is exclude authority. If *EXCLUDE authority is specified, no other authority can be specified. You can specify these system-defined authorities:

All (*ALL) authority provides all the object authorities and data authorities. The user can control the object's existence, specify the security for the object, change the object, and perform basic operations on the object such as run a program or display the object's description and contents.

Change (*CHANGE) authority provides object operational authority and all the data authorities. The user can add, change, and delete entries in an object, or read the contents of an entry in the object.

Use (*USE) authority provides object operational authority and read authority. The user can run a program or display the object's description or contents. The user is prevented from changing the object.

Exclude (*EXCLUDE) authority prevents the user from accessing the object. If this authority is specified, no other authority can be specified.

Figure 4-1 shows the subsets of object authorities and data authorities.

Figure 4-1. System-Defined Authority

	Object			Data			
Authority	**OPR**	**MGT**	**EXIST**	**READ**	**ADD**	**UPD**	**DLT**
*ALL	X	X	X	X	X	X	X
*CHANGE	X			X	X	X	X
*USE	X			X			
*EXCLUDE	No	authority					

Authorization Methods

The OS/400 program provides the following methods for assigning authority to users.

Private authority is the authority specifically given to a user for an object. A user's private authorities override any other authorities.

Public authority is the authority that applies to all users on the system who do not have authority granted through some other means of authorization. It can be specified for the object when the object is created or it can be defined in the authorization list that secures the object.

An **authorization list** contains a list of users and the authority that each user has to the objects that the list secures. One user's authority may differ from the authority of another user on the list. If a user on the authorization list also has private authority to the object secured by the list, the user's private authority overrides the authority specified on the authorization list.

A **group profile** provides a way of specifying the same set of authorities to multiple users by allowing each member to use the authority assigned to the group profile. Private authority to an object overrides the authority of the group profile.

For example, a group of end users can be assigned to a single group profile. The group profile only allows the users to enter and update entries. This eliminates the need to assign each user profile authority individually. The coordinator for the group of end users may require additional access to the files and can be given private authority, which overrides that allowed by the group profile.

In the following example, a user profile is created for a service scheduler called KCCLR. This individual is also assigned to a group profile called SCHED.

```
CRTUSRPRF  USRPRF(KCCLR)  PASSWORD(unique-password)
           JOBD(SPIFFY/KCCLR) GRPPRF(SCHED)
           TEXT('User profile for service scheduler')
```

With **adopted authority**, a user uses (adopts) the program or package owner's authority, and does not need authority specifically granted for the objects used by the program or package. The authority given to a user by the program adopt function is in addition to any private authorities the user has to objects.

For example, in the Spiffy distributed relational database, the service schedulers have authority to run an inventory program to verify that parts are currently in stock for a job. Through this program, the user has read and update authority to the regional inventory table, and only read authority to the local back ordered parts information. When users exit the program, their authority to the same data is limited to what is obtained through private authorities, group profiles, and public authority.

Object Ownership

Each object is assigned an owner when it is created. The owner is either the user who creates the object or the group profile if the member user profile has specified that the group profile should be the owner of the object. When the object is created, the owner is given all authority to the object. When you give ownership of an object to another user profile, you have the option to keep all authority you had before the change of ownership, or remove all the authority you had for the object.

The owner of an object always has all the authority for the object unless any or all authority is removed. The owner of an object has the authority to give any authority to any other user for the object. If you are the object owner, you can remove some of your specific authority and restore that authority when it is needed. For example, if a file contains critical information you can remove your object exist-

ence authority to prevent accidental deletion of the file and restore your object existence authority when you need it for this object.

Elements of Distributed Relational Database Security

A distributed relational database administrator needs to protect the resources of the ASs in the network without unnecessarily restricting access to data by ARs in the network.

An AR secures its objects and relational database to ensure only authorized users have access to distributed relational database programs. This is done using normal AS/400 object authorization to identify users and specify what each user (or group of users) is allowed to do with an object. Alternatively, authority to tables, views, and SQL packages can be granted or revoked using the SQL GRANT and REVOKE statements. Providing levels of authority to SQL objects on the AR helps ensure that only authorized users have access to an SQL application that accesses data on another system.

The level of system security in effect on the AS determines whether a request from an AR is accepted and whether the remote user is authorized to objects on the AS.

Some aspects of security planning for AS/400 systems in a distributed relational database network include:

- Physical security such as locked doors or secured buildings that surround the systems, modems, communication lines and terminals that can be configured in the line description and used in the route selection process
- Location security that verifies the identity of other systems in the network
- User-related security to verify the identity and rights of users on the local system and remote systems
- Object-related security to control user access to particular resources such as confidential tables, programs, and packages

Location, user-related, and object-related security are only possible if the system security level is set at level 20 or above.

When the system is using level 10 security, an AS/400 system connects to the network as a nonsecure system. The AS/400 system does not validate the identity of a remote system during session establishment and does not require conversation security on incoming program start requests. For level 10, security information configured for the APPC remote location is ignored and is not used during session or conversation establishment. If a user profile does not exist on the AS/400 system, one is created.

When the system is using security level 20 or above, an AS/400 system connects to the network as a secure system. The AS/400 system can then provide both session and conversation level security functions.

Having system security set at the same level across the systems in your network makes the task of security administration easier. An AS controls whether the session and conversation can be established by specifying what is expected from the AR to establish a session. For example, if the security level on the AR is set at 10 and the security level on the AS is above 10, the appropriate information may

not be sent and the session might not be established without changing security elements on one of the systems.

For more information on security levels, see the *Security Reference* manual and security consideration topics in the *APPC Programmer's Guide* or the *APPN Guide*.

Session Level and Location Security

Communications security occurs when a Systems Network Architecture (SNA) bind occurs between two locations, and involves session level and location security.

Session level security verifies the identity of the two systems attempting to establish a communications session. Session level security is established during communications configuration in one of two ways, depending on whether the network uses APPN.

- If you specify APPN(*NO) on the Create Controller Description (CRTCTLAPPC) command, communications devices are created manually. APPC devices are created by using the Create Device Description (CRTDEVAPPC) command. The LOCPWD parameter on the CRTDEVAPPC command specifies if a password is used to verify the remote location.
  - If you specify a password for LOCPWD on a device description the AS/400 system uses that password to validate the identity of the remote system during session establishment. The password must match the password specified on the remote system or the connection is not allowed.
  - If you specify *NONE on the LOCPWD parameter during configuration, the AS/400 system does not validate the identity of the remote system when a session is established.
- If you specify APPN(*YES) on the CRTCTLAPPC command, communications devices are created automatically using APPN. For APPN, a location-password on the remote location list specifies a password the two locations use to verify identities. Use the Create Configuration List (CRTCFGL) command to create a remote location list type (*APPNRMT).
  - If you specify a location password on an APPN remote location list, the AS/400 system uses that password to validate the identity of the local and remote system pair. The location password must match the location password specified on the remote system's remote location list or the connection is not allowed.
  - If you specify *NONE for the location password, the AS/400 system does not validate the identity of the remote system when a session is established. The remote system must also specify *NONE for the location password.

Location security establishes what security information each location requires from the other location for each remotely initiated APPC conversation. Location security is established during communication configuration in one of two ways, depending on whether APPN is used.

- In an APPC network, the SECURELOC parameter on the CRTDEVAPPC command specifies whether the local system allows the remote system to verify security. Specifying *YES for SECURELOC means that the local system allows the remote system to verify user security information. If you specify *NO on the

SECURELOC parameter, the local system verifies security information for the incoming request.

- In an APPN network, the secure-location value on the remote location list verifies security. Specifying *YES for secure-location on an APPN remote configuration list means that the local system allows the remote system to verify user security information. If you specify *NO, the local system verifies security information for the incoming request.

  Note: APPN creates location information based on the first device description that is varied on for the remote network ID, remote location name, and local location name pair. To avoid using security information that cannot be predicted, you must ensure that all of the device descriptions with the same remote network ID, remote location name, and local location name pair contain exactly the same security information.

For more information on session and location security issues, see the considerations chapter in the *APPC Programmer's Guide*.

APPN Configuration Lists

In an APPN network, location passwords are specified for those pairs of locations that are going to have end-to-end sessions between them. Location passwords need not be specified for those locations that are intermediate nodes.

The remote location list is created with the CRTCFGL command, and it contains a list of all remote locations, their location password, and whether the remote location is secure. There is one system-wide remote location configuration list on an AS/400 system. A central site AS/400 system can create location lists for remote AS/400 systems by sending them a control language (CL) program.

Changes can be made to a remote configuration list using the Change Configuration List (CHGCFGL) command, however, they do not take effect until all devices for that location are all in a varied off state.

When the Display Configuration List (DSPCFGL) command is used, there is no indication that a password exists. The CHGCFGL command indicates a password exists by placing *PASSWORD in the field if a password has been entered. There is no way to display the password. If you have problems setting up location security you may have to enter the password again on both systems to be sure the passwords match.

For more information on configuration lists, see the *APPN Guide*.

Conversation Level Security

Systems Network Architecture (SNA) logical unit (LU) 6.2 architecture identifies three conversation security designations that various types of systems in an SNA network can use to provide consistent conversation security across a network of unlike systems. The SNA security levels are:

SECURITY(NONE) No user ID or password is sent to establish communications.

SECURITY(SAME) A user ID is required, but no password is sent for communications.

SECURITY(PGM) Both a user ID and a password are sent for communications.

The AS/400 system supports all three SNA levels of conversation security. The AS controls the SNA conversation levels used for the conversation. The SECURELOC parameter on the APPC device description or the secure location value on the APPN remote location list determines what is accepted from the AR for the conversation.

For the SECURITY(NONE) level, an AS does not expect a user ID or password. The conversation is allowed using a default user profile on the AS. Whether a default user profile can be used for the conversation depends on the value specified on the DFTUSR parameter of the Add Communications Entry (ADDCMNE) command or the Change Communications Entry (CHGCMNE) command for a given subsystem. A value of *NONE for the DFTUSR parameter means the AS does not allow a conversation using a default user profile on the AS. SECURITY (NONE) is sent when no password or user ID is supplied and the AS has SECURELOC(*NO) specified.

For the SECURITY(SAME) level, an AS expects a user ID and an Already Verified indicator from the AR. A value of *YES for SECURELOC or secure location means SECURITY(SAME) conversation level security is used. A value of *NO for SECURELOC means the AS does not allow SNA SECURITY(SAME). This causes SECURITY(NONE) to be used.

For the SECURITY(PGM) level, an AS expects both a user ID and password from the AR for the conversation. To allow a conversation only if both a user ID and password are sent, the AS/400 AS must be set up so the SECURELOC parameter or the secure location value is *NO and no default user profile is specified for the communications subsystem. The password is validated when the conversation is established and is ignored for any following uses of that conversation.

An AS/400 AR sends a password if the USER and USING optional keywords and their associated values are coded on the SQL CONNECT statement. For example:

```
EXEC SQL CONNECT TO :locn USER :userid USING :pw;
```

Using Passwords: Once a conversation is established at the SECURITY(PGM) level, you do not need to enter a password again. If you connect to another AS, your conversation with the first AS may or may not be dropped, depending on the kind of AS you are connected to and your AR job attributes (for the specific rules, see "Controlling DDM Conversations" on page 6-15). If the conversation to the first AS is not dropped, it remains unused while you are connected to the second AS. If you connect again to the first AS and the conversation is unused, the conversation becomes active again without you needing to enter your user ID and password. On this second use of the conversation, your password is also not validated again. If another user ID and password are entered, they are ignored.

The OS/400, DB2, and SQL/DS licensed programs only accept passwords whose alphanumeric characters are in upper case. If you enter lowercase characters in your password when you connect, your connection is rejected.

Connecting to a Secure Distributed Relational Database

A valid user profile must exist on the AS to process distributed relational database work. You can specify a default user profile for a subsystem that handles communications jobs on an AS/400 system. The name of the default user profile is specified on the DFTUSR parameter of the Add Communications Entry (ADDCMNE) command on the AS. The ADDCMNE command adds a communications entry to a subsystem description used for communications jobs.

If a default user profile is specified in a communications subsystem, whether the AS is a secure location or not determines if the default user profile is used for this request. The SECURELOC parameter on the CRTDEVAPPC command, or the secure location designation on an APPN remote location list, specifies whether the AS is a secure location.

- If *YES is specified for SECURELOC or secure location on the AS, the AS considers the AR a secure location. A user ID and an Already Verified indicator is expected from the AR with its request. If a user profile exists on the AS that matches the user ID sent by the requester, the request is allowed. If not, the request is rejected.
- If *NO is specified for the SECURELOC parameter on the AS, the AS does not consider the AR a secure location. Although the AR still sends a user ID, the AS does not use this for the request. Instead, a default user profile on the AS is used for the request, if one is available. If no default user profile exists on the AS, the request is rejected.

Figure 4-2 shows all of the possible combinations of the elements that control SNA SECURITY(PGM) on the AS/400 system. A "Y" in any of the following columns indicates that the element is present or the condition is met. These elements are:

UID User ID

PWD Password

AVI Already Verified Indicator

SEC(Y) SECURELOC(*YES)

DFT Default User ID

VALID Are the user ID and password valid?

The elements in the "Access" column are:

Use UID Connection made with supplied user ID

Use DFT Connection made with default user ID

Reject Connection not made

Figure 4-2. Remote Access to a Distributed Relational Database

UID	PWD	AVI	SEC(Y)	DFT	Valid	Access
Y	Y		Y	Y	Y	Use UID
Y	Y		Y	Y		Reject
Y	Y		Y		Y	Use UID
Y	Y		Y			Reject
Y	Y			Y	Y	Use UID
Y	Y			Y		Reject
Y	Y				Y	Use UID
Y	Y					Reject
Y		Y	Y	Y	Y	Use UID
Y		Y	Y	Y		Reject
Y		Y	Y		Y	Use UID
Y		Y	Y			Reject
Y		Y		Y	Y	Use DFT
Y		Y		Y		Use DFT
Y		Y			Y	Reject
Y		Y				Reject
			Y	Y		Used DFT
			Y			Reject
				Y		Use DFT
						Reject

To avoid using default user profiles, create a user profile on the AS for every AR user that needs access to the distributed relational database objects. If you decide to use a default user profile, care must be taken when defining subsystems (object type *SBSD) to make sure that users are correctly identified on the system. For example, to protect the AS, do not add a communications entry with the following specifications:

```
ADDCMNE   SBSD(SAMPLE) DEV(*ALL) DFTUSER(QUSER)
```

In this example, the default user parameter specified as DFTUSER(QUSER) allows the system to accept job start requests without a user ID or password from a communications request. The communications job is signed on using the QUSER user profile.

Object Related Security

If the AS/400 system is an AS, there are two object-related levels at which security can be enforced to control access to its relational database tables.

The DDMACC parameter is used on the Change Network Attributes (CHGNETA) command to indicate whether the tables on this AS/400 system can be accessed at all by another system and, if so, at which level of security the incoming DDM requests are to be checked.

- If *REJECT is specified on the DDMACC parameter, all distributed relational database requests received by the AS are rejected. However, this system (as

an AR) can still use SQL requests to access tables on other systems that allow it. No remote system can access a database on any AS/400 system that specifies *REJECT.

If *REJECT is specified while an SQL request is already in use, all *new* jobs from any system requesting access to this system's database are rejected and an error message is returned to those jobs; existing jobs are not affected.

- If *OBJAUT is specified on the DDMACC parameter, normal object-level security is used on the AS as discussed in "Types of Authorities" on page 4-4.

  The DDMACC parameter is initially set to *OBJAUT. A value of *OBJAUT allows all remote requests, but they are controlled by the object authorizations on this AS. If the DDMACC value is *OBJAUT, the user profile used for the job must have appropriate object authorizations through private, public, group, or adopted authorities, or the profile must be on an authorization list for objects needed by the AR job. For each SQL object on the system, all users, no users, or only specific users (by user ID) can be authorized to access the object.

  The user ID that must be authorized to objects is the user ID of the AS job. See "Connecting to a Secure Distributed Relational Database" on page 4-11 for a discussion on what user profile the AS job runs under.

  When the value *OBJAUT is specified, it indicates that no further verification (beyond AS/400 object level security) is needed.

- For DDM jobs, if the name of an optional, user-supplied user exit program (or access control program) is specified on the DDMACC parameter, an additional level of security is used. The user exit program can be used to control whether a user of a source system can use a specific command to access a specific file on the target system.

  A qualified-program-name entry is only valid when using DDM files. If a user-written exit program name is specified for this parameter when handling a distributed relational database job, the system treats the entry as though *OBJAUT is specified.

The DDMACC parameter, initially set to *OBJAUT, can be changed to one of the previously described values by using the Change Network Attributes (CHGNETA) command, and its current value can be displayed by the Display Network Attributes (DSPNETA) command. You can also get the value in a CL program by using the Retrieve Network Attributes (RTVNETA) command.

If the DDMACC parameter value is changed, although it takes effect immediately, it affects only *new* distributed relational database jobs started on this system (as the AS). Jobs running on this AS before the change was made continue to use the old value.

For a description of the DDMACC parameter, see description of the Change Network Attributes (CHGNETA) command in the *Communications Management Guide*.

Authority to Distributed Relational Database Objects

You can use either the SQL GRANT and REVOKE statements or the control language (CL) Grant Object Authority (GRTOBJAUT) and Revoke Object Authority (RVKOBJAUT) commands to grant and revoke a user's authority to relational database objects. The SQL GRANT and REVOKE statements only operate on packages, tables, and views. In some cases, it is necessary to use GRTOBJAUT and RVKOBJAUT to authorize users to other objects, such as commands and programs.

The authority checked for SQL statements depends on whether the statement is static, dynamic, or being run interactively.

- For static SQL statements, authority is checked against the profile of the owner of the program containing the SQL statement when running in the *SQL naming mode, or against the profile of the person running the program when running in the *SYS naming mode. For more information on *SQL and *SYS naming conventions, see "Naming Distributed Relational Database Objects" on page 10-2.
- For dynamic SQL statements or for statements issued interactively, authority is checked against the profile of the person running the program or processing the statement.

Users running a distributed relational database application need authority to run the SQL package on the AS. The GRANT EXECUTE ON PACKAGE statement allows the owner of an SQL package, or any user with administrative privileges to it, to grant specified users the privilege to run the statements in an SQL package. You can use this statement to give all users authorized to the AS, or a list of one or more user profiles on the AS, the privilege to run statements in an SQL package.

Normally, users have processing privileges on a package if they are authorized to the distributed application program created using the CRTSQL*xxx* command. If the package is created using the CRTSQLPKG command you may have to grant processing privileges on the package to users. You can issue this statement in an SQL program or using interactive SQL. The following shows a sample statement:

```
GRANT EXECUTE
            ON PACKAGE  SPIFFY.PARTS1
            TO PUBLIC
```

The REVOKE EXECUTE ON PACKAGE statement allows the owner of an SQL package, or any user with administrative privileges to it, to remove the privilege to run statements in an SQL package from specified users. You can remove the EXECUTE privilege to all users authorized to the AS or to a list of one or more user profiles on the AS.

If you granted the same privilege to the same user more than once, revoking that privilege from that user nullifies all those grants. If you revoke an EXECUTE privilege on an SQL package you previously granted to a user, it nullifies any grant of the EXECUTE privilege on that SQL package, regardless of who granted it. The following shows a sample statement:

```
REVOKE EXECUTE
             ON PACKAGE  SPIFFY.PARTS1
             FROM PUBLIC
```

You can also grant authority to an SQL package using the GRTOBJAUT command or revoke authority to an SQL package using the RVKOBJAUT command.

Programs That Run Under Adopted Authority

A distributed relational database program can run under adopted authority, which means the user adopts the program owner's authority to objects used by the program while running the program. When a program is created using the *SQL precompiler option for naming, the program runs under the program owner's user profile.

An SQL package from an unlike system always adopts the package owner's authority for all static SQL statements in the package. An SQL package created on an AS/400 system using the CRTSQL*xxx* command with OPTION(*SQL) specified, also adopts the package owner's authority for all static SQL statements in the package.

A distributed relational database administrator can check security exposure on ASs by using the Display Program Adopt (DSPPGMADP) command. The DSPPGMADP command displays the programs and SQL packages that use a specified user profile, as shown below. You may also send the results of the command to a printer or to an output file.

```
                        Display Programs That Adopt

User profile . . . . . :   MPSUP

Object      Library   Type        Attribute   Text
INVENT      SPIFFY    *PGM                    Adopting program
CLIENT1     SPIFFY    *PGM                    Adopting program
TESTINV     TEST      *PGM        CLP         Test inventory pgm
INVENT1     SPIFFY    *SQLPKG                 SQL package
CLIENT1     SPIFFY    *SQLPKG                 SQL package
TESTINV     SPIFFY    *SQLPKG                 SQL package

                                                                  Bottom
Press Enter to continue

F3=Exit   F12=Cancel   F17=Top   F18=Bottom
(C) COPYRIGHT IBM CORP. 1980, 1991.
```

Protection Strategies in a Distributed Relational Database

Network security in an AS/400 distributed relational database must be planned to protect critical data on any AS from unauthorized access. But because of the distributed nature of the relational database, security planning must ensure that availability of data in the network is not unnecessarily restricted.

One of the decisions that a distributed relational database administrator needs to make is the system security level in place for each system in the network. A system security level of 10 provides no security for ASs other than physical security at the system site. A system security level of 20 provides some protection to ASs

because network security checking is done to ensure the local and remote system are correctly identified. However, this level does not provide the object authorization necessary to protect critical database elements from unauthorized access. An AS/400 system security level of 30 and above is the recommended choice for systems in a network that want to protect specific system objects.

The distributed relational database administrator must also consider how communications are established between ARs on the network and the ASs. Some questions that need to be resolved might include:

- Should a default user profile exist on an AS?

  Maintaining many user profiles throughout a network can be difficult. However, creating a default user profile in a communications subsystem entry opens the AS to incoming communications requests if the AS is not a secure location. In some cases this might be an acceptable situation, in other cases a default user profile might reduce the system protection capabilities too far to satisfy security requirements.

  For example, systems that serve many ARs need a high level of security. If their databases were lost or damaged, the entire network could be affected. Since it is possible to create user profiles or group profiles on an AS that identifies all potential users needing access, it is unnecessary for the database administrator to consider creating a default user profile for the communications subsystem or subsystems managing distributed relational database work.

  In contrast, an AS/400 system that rarely acts as an AS to other systems in the network and does not contain sensitive or critical data might use a default user profile for the communications subsystem managing distributed relational database work. This might prove particularly effective if the same application is used by all the other systems in the network to process work on this database.

- How should access to database objects be handled?

  Authority to objects can be granted through private authority, group authority, public authority, adopted authority, and authorization lists. While a user profile (or default profile) has to exist on the AS for the communications request to be accepted, how the user is authorized to objects can affect performance.

  Whenever possible, use group authority or authorization lists to grant access to a distributed relational database object. It takes less time and system resources to check these than to review all private authorities.

Chapter 5. Setting Up an AS/400 Distributed Relational Database

To set up an AS/400 distributed relational database, you need to work with three functions of the AS/400 system:

- Work management
- The relational database directory
- Basic data management

Work management is the set of system functions that processes work in the system. The relational database directory is used by the AS/400 system to direct work to a particular relational database in the network. After you have your systems ready for work and have your relational database directories set up, you can add data to tables on any application server (AS) in your network. This chapter introduces these three topics and helps you to set up AS/400 systems for distributed relational database work.

Connection and set up information for a distributed relational database network of unlike systems can be found in the *Distributed Relational Database Connectivity Guide*.

Work Management on the AS/400 System

All of the work done on the AS/400 system is submitted through the work management function. On an AS/400 system, you can design specialized operating environments to handle different types of work to satisfy the requirements of your system. However, when the operating system is installed, it includes a work management environment that supports interactive and batch processing, communications, and spool processing.

On the AS/400 system, all user jobs operate in an environment called a **subsystem**, defined by a subsystem description, where the system coordinates processing and resources. Users can control a group of jobs with common characteristics independently of other jobs if the jobs are placed in the same subsystem. You can easily start and end subsystems as needed to support the work being done and to maintain the performance characteristics you desire.

The basic types of jobs that run on the system are interactive, communications, batch, spooled, autostart, and prestart.

An interactive job starts when you sign on a work station and ends when you sign off. A communications batch job is a job started from a program start request from another system. A non-communications batch job is started from a job queue. Job queues are not used when starting a communications batch job. Spooling functions are available for both input and output. Autostart jobs perform repetitive work or one-time initialization work. Autostart jobs are associated with a particular subsystem, and each time the subsystem is started, the autostart jobs associated with it are started. Prestart jobs are jobs that start running before the remote program sends a program start request.

Setting Up Your Work Management Environment

One subsystem, called a **controlling subsystem**, starts automatically when you load the system. Two controlling subsystem configurations are supplied by IBM, and you can use them without change. The first configuration includes the following subsystems:

- QBASE, the controlling subsystem, supports interactive, batch, and communications jobs.
- QSPL supports processing of spooling readers and writers.

QBASE automatically starts when the system is started. An automatically started job in QBASE starts QSPL.

The second controlling subsystem configuration supplied is more complex. This configuration consists of the following subsystems:

- QCTL, the controlling subsystem, supports interactive jobs started at the console.
- QINTER supports interactive jobs started at other work stations.
- QCMN supports communications jobs.
- QBATCH supports batch jobs.
- QSPL supports processing of spooling readers and writers.

If you change your configuration to use the QCTL controlling subsystem, it starts automatically when the system is started. An automatically started job in QCTL starts the other subsystems.

You can change your subsystem configuration from QBASE to QCTL by changing the system value QCTLSBSD (controlling subsystem) to QCTL on the Change System Value (CHGSYSVAL) command and starting the system again.

You can change the IBM-supplied subsystem descriptions or any user-created subsystem descriptions by using the Change Subsystem Description (CHGSBSD) command. You can use this command to change the storage pool size, storage pool activity level, and the maximum number of jobs for the subsystem description of an active subsystem.

For more information about work management, subsystems, and jobs on the AS/400 system see, the *Work Management Guide*. For more information about work management for communications and communications subsystems, see the *Communications Management Guide*.

Considerations for Setting Up Subsystems

In a distributed relational database, communications jobs and interactive jobs are the main types of work an administrator must plan to manage on each system. Systems in the network start communications jobs to handle requests from an application requester (AR); an AR's communications requests to other systems normally originate from interactive or batch jobs on the local system. Setting up an efficient work management environment for the distributed relational database network systems can enhance your overall network performance by allocating system resources to the specific needs of each AS and AR in the network.

When the OS/400 licensed program is first installed, QBASE is the default controlling subsystem. As the controlling subsystem, QBASE allocates system resources

between the two subsystems QBASE and QSPL. Interactive jobs, communications jobs, batch jobs, and so on, allocate resources within the QBASE subsystem. Only spooled jobs are managed under a different subsystem, QSPL. This means you have less control of system resources for handling communications jobs versus interactive jobs than you would using the QCTL controlling subsystem.

Using the QCTL subsystem configuration, you have control of four additional subsystems for which the system has allocated storage pools and other system resources. Changing the QCTL subsystems, or creating your own subsystems gives you even more flexibility and control of your processing resources.

Different system requirements for some of the systems in the Spiffy Corporation distributed relational database network may require different work management environments for best network efficiency. The following discussions show how the distributed relational database administrator can plan a work management subsystem to meet the needs of each AS/400 system in the Spiffy distributed relational database network.

In the Spiffy Corporation system organization, a small dealership may be satisfied with a QBASE level of control for the various jobs its users have on the system. For example, requests to a small dealership's relational database from the regional AR (to update dealer inventory levels for a shipment) are handled as communications jobs. Requests from a dealership user to the regional AS, to request a part not currently in stock locally, is handled as an interactive job on the dealership system. Both activities are relatively small jobs because the dealership is smaller and handles fewer service orders, parts sales and so on. The system coordination of resources in the QBASE subsystem provides the level of control this enterprise requires for their interactive and communications needs.

A large dealership, on the other hand, probably manages its work through the QCTL subsystem, because of the different work loads associated with the different types of jobs.

The number of service orders booked each day can be high, requiring a query to the local relational database for parts or to the regional center AS for parts not in stock at the dealership. This type of activity starts interactive jobs on their system. The dealership also starts a number of interactive jobs that are not distributed relational database related jobs, such as enterprise personnel record keeping, marketing and sales planning and reporting, and so on. Requests to this dealership from the regional center for performance information or to update inventory or work plans are communications jobs that the dealership wants to manage in a separate environment. The large dealership can also receive a request from another dealership for a part that is out of stock at the regional center.

For a large dealership, the QCTL configuration with separate subsystem management for QINTER and QCMN provides more flexibility and control for managing its system work environment. In this example, interactive and communications jobs at the dealership system can be allocated more of the system resources than other types of jobs. Additionally, if communications jobs are typically fewer than interactive jobs for this system, resources can be targeted toward interactive jobs, by changing the subsystem descriptions for both QINTER and QCMN.

A work management environment tailored to a Spiffy Corporation regional center perspective is also important. In the Spiffy network, the regional center is an AR to

each dealership when it updates the dealership inventory table with periodic parts shipment data, or updates the service plan table with new or updated service plans for specific repair jobs. Some of these jobs can be run as interactive jobs (on the regional system) in early morning or late afternoon when system usage is typically less, or run as batch jobs (on the regional system) after regular business hours. The administrator can tailor the QINTER and QBATCH subsystems to accommodate specific processing times and resource needs.

The regional center is also an AS for each dealership when a dealership needs to query the regional relational database for a part not in stock at the dealership, a service plan for a specific service job (such as rebuilding a steering rack), or for technical bulletins or recall notifications since the last update to the dealership relational database. These communications jobs can all be managed in QCMN.

However, a closer examination of some specific aspects of distributed relational database network use by the KC000 (Kansas City) regional center and the dealerships it serves suggests other alternatives to the distributed relational database administrator at Kansas City.

The KC000 system serves several very large dealerships that handle hundreds of service orders daily, and a few small dealerships that handle fewer than 20 service orders each day. The remaining medium-sized dealerships each handle about 100 service orders daily. One problem that presents itself to the distributed relational database administrator is how to fairly handle all the communications requests to the KC000 system from other systems. A large dealership could control QCMN resources with its requests so that response times and costs to other systems in the network are unsatisfactory.

The distributed relational database administrator can create additional communications subsystems so each class of dealerships (small, medium, or large) can request support from the AS and generally receive better response. By tailoring the subsystem attributes, prestart job entries, communications work entries, and routing entries for each subsystem description, the administrator controls how many jobs can be active on a subsystem and how jobs are processed in the subsystem.

For more information on work management topics for the AS/400 system, see the *Work Management Guide*. For more information about changing attributes, work entries and routing entries for communications, see the *Communications Management Guide*.

Using the Relational Database Directory

The OS/400 program uses the relational database directory to define the relational database names that can be accessed by applications running on an AS/400 system and to associate these relational database names to their corresponding network parameters. The relational database directory allows an AR to accept a relational database name from the application and translate this name into the appropriate Systems Network Architecture (SNA) network identifier and logical unit (LU) name values for communications processing.

Each AS/400 system in the distributed relational database network must have a relational database directory configured. There is only one relational database directory on an AS/400 system. Each AR in the distributed relational database network must have an entry in its relational database directory for its local relational

database and one for each remote relational database the AR accesses. Any AS/400 system in the distributed relational database network that acts only as an AS must have an entry in its relational database directory for the local relational database, but does not need to include the relational database names of other remote relational databases in its directory.

The relational database name assigned to the local relational database must be unique. That is, it should be different from any other relational database in the network. Names assigned to other relational databases in the directory identify remote relational databases, and must match the name an AS uses to identify its local relational database.

Working with the Relational Database Directory

The following commands let you work with the relational database directory on your system:

ADDRDBDIRE Add Relational Database Directory Entry

CHGRDBDIRE Change Relational Database Directory Entry

DSPRDBDIRE Display Relational Database Directory Entry

RMVRDBDIRE Remove Relational Database Directory Entry

WRKRDBDIRE Work with Relational Database Directory Entry

The Add RDB Directory Entry (ADDRDBDIRE) display is shown below. You can use the prompts in this display or the ADDRDBDIRE command to add an entry to the relational database directory.

```
                    Add RDB Directory Entry (ADDRDBDIRE)

Type choices, press Enter.

Relational database  . . . . . > MP311          Name
Remote location  . . . . . . . > MP311          Name, *LOCAL
Text . . . . . . . . . . . . . > 'Oak Street Dealership'
```

In this example, an entry is made to add a relational database named MP311 for a system with a remote location name of MP311 to the relational database directory on the local system. The remote location name does not have to be defined before a relational database directory entry using it is created. However, the remote location name must be defined before the relational database directory entry is used in an application. The relational database name (RDB) parameter and the remote location name (RMTLOCNAME) parameter are required for the ADDRDBDIRE command. The descriptive text (TEXT) parameter is optional. As shown in this example, it is a good idea to make the relational database name the same as the system name or location name specified for this system in your network configuration. This can help you identify a database name and correlate it to a particular system in your distributed relational databasenetwork, especially if your network is complex.

To see the other optional parameters on this command, press F10 on the Add RDB Directory Entry (ADDRDBDIRE) display. These optional parameters are shown below.

```
                    Add RDB Directory Entry (ADDRDBDIRE)

Type choices, press Enter.

Relational database  . . . . . > MP311          Name
Remote location  . . . . . . . . > MP311          Name, *LOCAL
Text . . . . . . . . . . . . . . > 'Oak Street Dealership'

Device:
  APPC device description  . . .   *LOC          Name, *LOC
Local location . . . . . . . . .   *LOC          Name, *LOC, *NETATR
Remote network identifier  . . .   *LOC          Name, *LOC, *NETATR, *NONE
Mode . . . . . . . . . . . . . .   *NETATR       Name, *NETATR
Transaction program  . . . . . .   *DRDA         Character value, *DRDA
```

The system provides the following default values for the additional ADDRDBDIRE command parameters:

- Device (DEV)
- Local location (LCLLOCNAME)
- Remote network identifier (RMTNETID)
- Mode (MODE)
- Transaction program (TNSPGM)

Note: The transaction program name parameter in the AS/400 system is TNSPGM and in SNA, it is TPN.

- If you use the defaults with advanced program-to-program communications (APPC), the system determines the device, the local location, and the remote network identifier that will be used. The mode name defined in the network attributes is used and the transaction program name for Distributed Relational Database Architecture (DRDA) support is used.
- If you use the defaults with advanced peer-to-peer networking (APPN), the system ignores the device (DEV) parameter, and uses the local location name, remote network identifier, and mode name defined in the network attributes.

You can change any of these default values on the ADDRDBDIRE command. For example, you may have to change the TNSPGM parameter to communicate with an SQL/DS system. By default for SQL/DS support, the TNSPGM is the name of the SQL/DS database to which you want to connect. The default TNSPGM parameter value for DRDA (*DRDA) is X'07F6C4C2'. For more information on transaction program name, see:

- "Setting QCNTSRVC as a TPN on an OS/400 Application Requester" on page 9-26.
- "Setting QCNTSRVC as a TPN on an SQL/DS Application Requester" on page 9-26.
- "Setting QCNTSRVC as a TPN on a DB2 Application Requester" on page 9-27.
- "Setting QCNTSRVC as a TPN on an OS/2 Application Requester" on page 9-27.

The Work with RDB Directory Entries display, shown below, provides options that allow you to add, change, display, or remove a relational database directory entry.

```
                    Work with RDB Directory Entries

Position to  . . . . . .

Type options, press Enter.
  1=Add   2=Change   4=Remove   5=Display details   6=Print details

        Relational          Remote
Option  Database            Location  Text
__      KC000               KC000     Kansas City region database
__      MP000               *LOCAL    Minneapolis region database
__      MP101               MP101     Dealer database MP101
__      MP102               MP102     Dealer database MP102
__      MP211               MP211     Dealer database MP211
__      MP215               MP215     Dealer database MP215
4_      MP311               MP311     Dealer database MP311
```

As shown on the display, option 4 can be used to remove an entry from the relational database directory on the local system. If you remove an entry, you receive another display that allows you to confirm the remove request for the specified entry or select a different relational database directory entry. If you use the Remove Relational Database Directory (RMVRDBDIRE) command, you have the option of specifying a specific relational database name, generic names, all directory entries, or just the remote entries.

You have the option on the Work with RDB Directory Entries display to display the details of an entry. Output from the Work with RDB Entries display is to a display. However, if you use the Display RDB Directory Entries (DSPRDBDIRE) command, you can send the output to a printer or an output file. The relational database directory is not an AS/400 object, so using an output file provides a means of backup for the relational database directory. For more information about using the DSPRDBDIRE command with an output file for backing up the relational database directory, see "Saving and Restoring Relational Database Directories" on page 7-11.

You have the option on the Work with RDB Directory Entries display to change an entry in the relational database directory. You can also use the Change Relational Database Directory Entries (CHGRDBDIRE) command to make changes to an entry in the directory. You can change any of the optional command parameters and the remote location name of the system. You cannot change a relational database name for a directory entry. To change the name of a relational database in the directory, remove the entry for the relational database and add an entry for the new database name.

Relational Database Directory Setup Example

The Spiffy Corporation network provides an example to illustrate how the relational database directory is used on systems in a distributed relational database network and show how each is set up. A simple relationship to consider is the one between two regional offices as shown below:

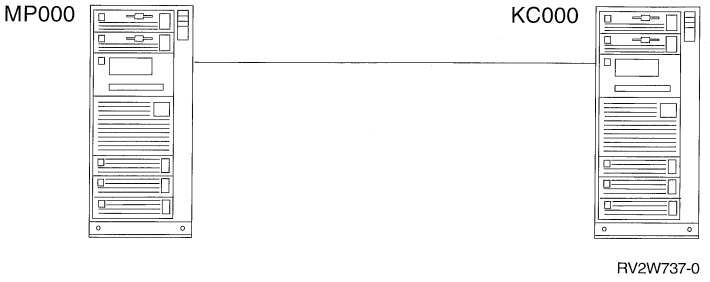


Figure 5-1. Relational Database Directory Setup for Two Systems

The relational database directory for each regional office must contain an entry for the local relational database and an entry for the remote relational database because each system is both an AR and an AS. The commands to create the relational database directory for the MP000 system are:

```
ADDRDBDIRE    RDB(MP000) RMTLOCNAME(*LOCAL) TEXT('Minneapolis region database')

ADDRDBDIRE    RDB(KC000) RMTLOCNAME(KC000) TEXT('Kansas City region database')
```

In the above example, the MP000 system identifies itself as the local relational database by specifying *LOCAL for the RMTLOCNAME parameter. There is only one relational database on an AS/400 system. You can simplify identification of your network relational databases if you make the relational database names in the directory the same as the system name and the local location name for the local system, and the same as the remote location name for the remote system.

Note: The system name is specified on the SYSNAME parameter of the Change Network Attributes (CHGNETA) command. The local system is identified on the LCLLOCNAME parameter of the CHGNETA command during communications configuration, as shown in the example on page 3-18. Remote locations are identified with the RMTCPNAME parameter on the Create Controller (CRTCTLAPPC) command during communications configuration as shown on page 3-19. Using the same names for system names, network locations, and database names can help avoid confusion, particularly in complex networks.

The corresponding entries for the KC000 system relational database directory are:

```
ADDRDBDIRE    RDB(KC000) RMTLOCNAME(*LOCAL) TEXT('Kansas City region database')

ADDRDBDIRE    RDB(MP000) RMTLOCNAME(MP000) TEXT('Minneapolis region database')
```

A more complex example to consider is that of a regional office to its dealerships. For example, to access relational databases in the network shown below, the relational database directory for MP000 system must be expanded to include an entry for each of its dealerships.

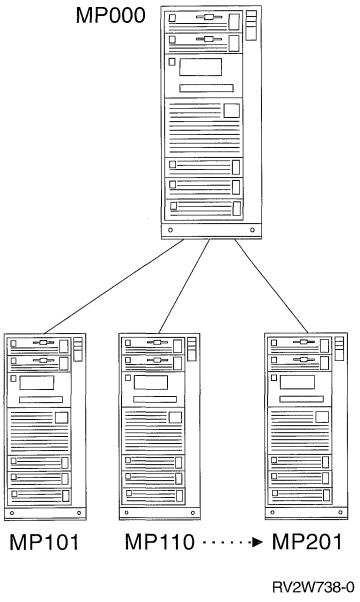


Figure 5-2. Relational Database Directory Setup for Multiple Systems

A sample of the commands used to complete the MP000 relational database directory to include all its dealer databases is as follows:

```
PGM
ADDRDBDIRE     RDB(MP000) RMTLOCNAME(*LOCAL) +
               TEXT('Minneapolis region database')
ADDRDBDIRE     RDB(KC000) RMTLOCNAME(KC000)
               TEXT('Kansas City region database')
ADDRDBDIRE     RDB(MP101) RMTLOCNAME(MP101)
               TEXT('Dealer database MP101')
ADDRDBDIRE     RDB(MP002) RMTLOCNAME(MP110)
               TEXT('Dealer database MP110')
.
.
.
ADDRDBDIRE     RDB(MP215) RMTLOCNAME(MP201)
               TEXT('Dealer database MP201')
ENDPGM
```

In the above example, each of the region dealerships is included in the Minneapolis relational database directory as a remote relational database.

Since each dealership can serve as an AR to MP000 and to other dealership ASs, each dealership must have a relational database directory that has an entry for itself as the local relational database and the regional office and all other dealers as remote relational databases. The database administrator has several options to create a relational database directory at each dealership system.

The method that uses the most time and is most prone to error is to create a relational database directory at each system by using the ADDRDBDIRE command to create each directory entry on all systems that are part of the MP000 distributed relational database network.

A better alternative is to create a control language (CL) program like the one shown in the above example for the MP000. The distributed relational database adminis-

trator can copy this CL program for each of the dealership systems. To customize this program for each dealership, the database administrator changes the remote location name of the MP000 system to MP000, and changes the remote location name of the local dealership to *LOCAL. The distributed relational database administrator can distribute the customized CL program to each dealership to be run on that system to build its unique relational database directory.

A third method is to write a program that reads the relational database directory information sent to an output file as a result of using the Display Relational Database Directory Entry (DSPRDBDIRE) command. This program can be distributed to the dealerships, along with the output file containing the relational database directory entries for the MP000 system. Each system could read the MP000 output file to create a local relational database directory. The Change Relational Database Directory Entry (CHGRDBDIRE) command can then be used to customize the MP000 system directory for the local system. For more information about using an output file to create relational database directory entries, see "Saving and Restoring Relational Database Directories" on page 7-11.

Setting Up DDM Files

The implementation of DRDA support on the AS/400 system uses Distributed Data Management (DDM) conversations for communications. Because of this, you can use DDM in conjunction with distributed relational database processing. You can use DDM to submit remote commands to a target system, copy tables from one AS/400 system to another, and process nondistributed relational database work on another system.

With distributed relational database, information the AR needs to connect to a database is provided in the relational database directory. When you use DDM, you must create a separate DDM file for each file you want to work with on the target system. The DDM file is used by the application on the source system to identify a remote file on the target system and the communications path to the target system.

Some database administration tasks discussed in Chapter 6, "Distributed Relational Database Administration and Operation Tasks" use DDM to access remote files. A DDM file is created using the Create DDM File (CRTDDMF) command. You can create a DDM file before the file and communication path named in the file have been created. However, the file named in the DDM file and the communications information must be created before the DDM file is used by an application.

The following example shows how a DDM file is created:

```
CRTDDMF FILE (TEST/KC105TST) RMTLOCNAME(KC105)
    RMTFILE(SPIFFY/INVENT)
```

This command creates a DDM file named KC105TST and stores it in the TEST library on the source system. This DDM file uses the remote location KC105 to access a remote file named INVENT stored in the SPIFFY library on the target AS/400 system.

You can use options on the Work with DDM Files display to change, delete, display or create DDM files. For more information about using DDM files, see the *DDM Guide*.

Loading Data into Tables

Applications in the distributed relational database environment operate on data stored in tables. In general, applications are used to query a table for information, to insert, update, or delete rows of a table or tables, or to create a new table. Other situations occur where data on one system must be moved to another system. This section discusses many of the methods available to load new data into a table, move data from one AS/400 system to another, or move data to an AS/400 system from a non-AS/400 system.

Loading New Data into Tables

You load data into a table by entering each data item into the table. On the AS/400 system, you can use SQL, the Query Management/400 function, or the data file utility portion of AS/400 Application Development Tools to create applications that insert data into a table.

Using SQL to Load Data into a Table

A simple method of loading data into a table is to use an SQL application and the SQL INSERT operation.

Consider a situation in which a Spiffy regional center needs to add inventory items to a dealership's inventory table on a periodic basis as regular inventory shipments are made from the regional center to the dealership.

```
INSERT INTO SPIFFY.INVENT
       (PART,
        DESC,
        QTY,
        PRICE)
  VALUES
       ('1234567',
        'LUG NUT',
        25,
        1.15 )
```

The statement above inserts one row of data into a table called INVENT in an SQL collection named SPIFFY.

For each item on the regular shipment, an SQL INSERT statement places a row in the inventory table for the dealership. In the above example, if 15 different items were shipped to the dealership, the application at the regional office could include 15 SQL INSERT statements or a single SQL INSERT statement using host variables.

In this example, the regional center is using an SQL application to load data in to a table at an AS. Run-time support for SQL is provided in the OS/400 licensed program, so the AS does not need the SQL/400 licensed program. However, the SQL/400 licensed program is required to write the application. For more information on the SQL/400 programming language see the *SQL/400* Programmer's Guide* and the *SQL/400* Reference*.

Using the Query Management/400 Function

The OS/400 licensed program provides a query management function that allows you to manipulate data in tables and files. A query is created using an SQL query statement. You can run the query through CL commands or through a query callable interface in your application program. Using the query management function, you can insert a row of data into a table for the inventory updates described in the previous section as follows.

Create a source member INVLOAD in the source physical file INVLOAD and the SQL statement:

```
INSERT INTO SPIFFY/INVENT
   (PART, DESC, QTY, PRICE)
 VALUES
   (&PARTVALUE, &DESCVALUE, &QTYVALUE, &PRICEVALUE)
```

Use a CL command to create a query management query object:

```
CRTQMQRY QMQRY(INVLOAD) SRCFILE(INVLOAD) SRCMBR(INVLOAD)
```

The following CL command places the INSERT SQL statement results into the INVENT table in the SPIFFY collection. Use of variables in the query (&PARTVALUE, &DESCVALUE, and so on) allows you to enter the desired values as part of the STRQMQRY call, rather than requiring that you create the query management query again for each row.

```
STRQMQRY QMQRY(INVLOAD) RDB(KC000)
   SETVAR((PARTVALUE '''1134567''') (DESCVALUE '''Lug Nut''')
       (QTYVALUE 25) (PRICEVALUE 1.15))
```

The query management function is dynamic, which means its access paths are built at run time instead of when a program is compiled. For this reason the Query Management/400 function is not as efficient for loading data into a table as an SQL application. However, you need the SQL/400 program to write an application; run-time support for SQL and query management is part of the OS/400 licensed program.

For more information on the query management function, see the *Query Management/400 Programmer's Guide and Reference* and the *Query Management/400 Programmer's Guide and Reference*.

Using Data File Utility

The data file utility (DFU), which is part of the AS/400 Applications Development Tools package available from IBM, is a program builder that helps you create programs to enter data, update tables, and make inquiries. You do not need a programming language to use DFU. Your data entry, maintenance, or inquiry program is created when you respond to a series of displays. An advantage in using DFU is that its generic nature allows you to create a database update program to load data to a table faster than you could by using programming languages such as SQL. You can work with data on a remote system using DFU with DDM files, or by using display station pass-through to run DFU at the target system.

For more information on the DFU program generator, see the *DFU User's Guide and Reference*.

Moving Data from One AS/400 System to Another

A number of situations occur in enterprise operations that could require moving data from one AS/400 system to another. For example, a new dealership might open in a region, and some clients from one or two other dealerships might be transferred to the new dealership as determined by client address. Perhaps a dealership closed or no longer represents Spiffy Corporation sales and service. That dealer's inventories and required service information must be allocated to either the regional office or other area dealerships. Perhaps a dealership has grown to the extent that it needs to upgrade its AS/400 system, and the entire database must be moved to the new system.

Some alternatives for moving data from one AS/400 system to another are:

- User-written application programs
- Interactive SQL
- Query Management/400 functions
- Copy to and from tape or diskette devices
- Copy file commands with DDM
- The network file commands
- AS/400 system save and restore commands

Creating a User-Written Application Program

Using distributed relational database, you can only connect to one relational database within a unit of work, so moving data from one system to another under distributed relational database does not provide the best performance. Distributed relational database support can be used with non-SQL database access methods to move data from one system to another. For example, distributed relational database can be used to get data from the remote system, and a high-level language input or output routine can be used to put the data in the local database. Either the distributed relational database SQL portion or the high-level language portion of the operation can be under commitment control, but not both.

Using Interactive SQL

Using the SQL SELECT statement and interactive SQL, you can query a database on another AS/400 system for data you need to create or update a table on the local system. The SELECT statement allows you to specify the table name and columns containing the desired data, and selection criteria or filters that determine which rows of data are retrieved. If the SELECT statement is successful, the result is one or more rows of the specified table.

In addition to getting data from one table, SQL allows you to get information from columns contained in two or more tables in the same database by using a join operation. If the SELECT statement is successful, the result is one or more rows of the specified tables. The data values in the columns of the rows returned represent a composite of the data values contained in specified tables.

Using an interactive SQL query, the results of a query can be placed in a database file on the local system. If a commitment control level is specified for the interactive SQL process, it applies to the AS; the database file on the local system is under a commitment control level of *NONE.

Interactive SQL allows you to do the following:

- Create a new file for the results of a select.

- Replace and existing file.
- Create a new member in a file.
- Replace a member.
- Append the results to an existing member.

Consider the situation in which the KC105 dealership is transferring its entire stock of part number '1234567' to KC110. KC110 queries the KC105 database for the part they acquire from KC105. The result of this inventory query is returned to a database file that already exists on the KC110 system. This is the process you can use to complete this task:

Use the Start SQL (SRTSQL) command to get the interactive SQL display. Before you enter any SQL statement (other than a CONNECT) for the new database, specify that the results of this operation are sent to a database file on the local system by doing the following steps:

1. Select the Services option from the Enter SQL Statements display.
2. Select the Change Session Attributes option from the Services display.
3. Enter the Select Output Device option from the Session Attributes Display.
4. Type a 3 for a database file in the Output device field and press Enter. The following display is shown:

```
                                Change File

Type choices, press Enter.

  File . . . . . .   QSQLSELECT   Name
    Library  . . . .   QGPL         Name
  Member . . . . . .   *FILE        Name, *FILE, *FIRST

  Option . . . . . .   1            1=Create new file
                                    2=Replace file
                                    3=Create new member
                                    4=Replace member
                                    5=Add to member


  For a new file:
    Authority  . . . . .   *LIBCRTAUT   *LIBCRTAUT, *CHANGE, *ALL
                                        *EXCLUDE, *USE
                                        authorization list name

    Text . . . . . .


F3=Exit     F5=Refresh     F12=Cancel
```

5. Specify the name of the database file that is to receive the results.

When the database name is specified, you can begin your interactive SQL processing as shown in the example below.

```
                              Enter SQL Statements

 Type SQL statement, press Enter.
      Current connection is to relational database KC000.
      CONNECT TO KC105
      Current connection is to relational database KC105.
====> SELECT * FROM INVENTORY
         WHERE PART = '1234567'
                                                                          Bottom
 F3=Exit    F4=Prompt    F6=Insert line   F9=Retrieve    F10=Copy line
 F12=Cancel              F13=Services     F24=More keys
                                           (C) COPYRIGHT IBM CORP. 1982, 1991.
```

For more information on the SQL/400 programming language and interactive SQL, see the *SQL/400* Programmer's Guide* and the *SQL/400* Reference*.

Using the Query Management/400 Function

The Query Management/400 function provides almost the same support as interactive SQL for querying a remote system and returning the results in an output file to the local system.

Both interactive SQL and the query management function can perform data manipulation operations (INSERT, DELETE, SELECT, and so on) for files or tables without the requirement that the table (or file) already exist in a collection (it can exist in a library). Also, query management uses SQL CREATE TABLE statements to provide data definition when a new table is created on the system as a result of the query. Tables created from a query management function follow the same guidelines and restrictions that apply to a table created using SQL.

However, the query management function does not allow you to specify a member when you want to add the results to a file or table. The results of a query function are placed in the first file member unless you use the OVRDBF command to specify a different member before starting the query management function.

For more information on the query management function, see the *Query Management/400 Programmer's Guide and Reference* and the *Query Management/400 Programmer's Guide and Reference*.

Copying Files to and from Tape or Diskette

You can copy a table or file to tape or diskette using the Copy to Tape (CPYTOTAP) and Copy to Diskette (CPYTODKT) commands on the AS/400 system. Data on tape or diskette can be loaded on another AS/400 system using the Copy From Tape (CPYFRMTAP) and Copy From Diskette (CPYFRMDKT) commands. For more information about using these commands, see the *Guide to Programming for Tape and Diskette*.

Using Copy File Commands

Another way to move data from one AS/400 system to another is to copy the data using the copy file commands with DDM. You can use the Copy File (CPYF), Copy Source File (CPYSRCF), and Copy from Query File (CPYFRMQRYF) commands to copy data between files on source and target systems. You can copy local relational database or device files from (or to) remote database files, and remote files can also be copied to remote files.

For example, if a dealership closes, the distributed relational database administrator can copy the client and inventory tables from the remote system to the local regional system. The administrator needs a properly authorized user profile on the target system to access and copy the tables and must create a DDM file on the source system for each table or file that is copied. The following example shows the command the database administrator would use to copy a table called INVENT in a collection called SPIFFY from a system with a remote location name of KC105 to a regional center system called KC000. A DDM file called INCOPY in a library called TEST on the source system KC000 is used for the file access. These commands are run on the KC000 system:

```
CRTDDMF   FILE(TEST/INCOPY) RMTFILE(SPIFFY/INVENT)
   RMTLOCNAME(KC105)
CPYF      FROMFILE(TEST/INCOPY)  TOFILE(TEST/INVENTDDM)
   MBROPT(*ADD)
```

In this example, the administrator runs the commands on the KC000 system. If the administrator is not on the KC000 system, then pass-through must be used to run these commands on the KC000 system. The SBMRMTCMD command cannot be used to run the above commands because the AS/400 system cannot be a source system and a target system for the same job.

Consider the following items when using this command with DDM:

- A DDM file can be specified on the FROMFILE and the TOFILE parameters for the CPYF and CPYSRCF commands.

  Note: For the Copy from Query File (CPYFRMQRYF), Copy from Diskette (CPYFRMDKT), and Copy from Tape (CPYFRMTAP) commands, a DDM file name can be specified only on the TOFILE parameter; for the Copy to Diskette (CPYTODKT) and Copy to Tape (CPYTOTAP) commands, a DDM file name can be specified only on the FROMFILE parameter.

- When a delete-capable file is copied to a non-delete capable file, you must specify COMPRESS(*YES), or an error message is sent and the job ends.
- If the remote file name on a DDM file specifies a member name, the member name specified for that file on the CPYF command must be the same as the member name on the remote file name on the DDM file. In addition, the Override Database File (OVRDBF) command cannot specify a member name that is different from the member name on the remote file name on the DDM file.
- If a DDM file does not specify a member name and if the OVRDBF command specifies a member name for the file, the CPYF command uses the member name specified on the OVRDBF command.
- If the TOFILE parameter is a DDM file that refers to a file that does not exist, CPYF creates the file. Following are special considerations for remote files created with the CPYF command:

- The user profile for the target DDM job must be authorized to the CRTPF command on the target system.
- For an AS/400 system target, the TOFILE parameter has all the attributes of the FROMFILE parameter except those described in the *Data Management Guide*.

For more information about using the Copy File commands to copy between systems, see the *DDM Guide*.

Using Network File Commands

Data can be transferred over networks protocols that support SNA distribution services (SNADS). In addition to APPC and APPN protocols used with distributed relational database processing, SNADS can be used with binary synchronous equivalence link (BSCEL) and SNA Upline Facility (SNUF) protocols. An AS/400 system supported by SNADS can send data to another system with the Send Network File (SNDNETF) command and receive a network file from another AS/400 system with the Receive Network File (RCVNETF) and Work with Network File (WRKNETF) commands.

Using System Save and Restore Commands

You can move a table from another AS/400 system using the Save Object (SAVOBJ) and Restore Object (RSTOBJ) commands. The save commands save database files on tape, diskette, or a save file. The save file can be distributed to another system through communications.

The save and restore commands used to save and restore tables or files include:

- Save Library (SAVLIB) saves one or more collections or libraries
- Save Object (SAVOBJ) saves one or more objects (including database tables and views)
- Save Changed Object (SAVCHGOBJ) saves any objects that have changed since either the last time the collection or library was saved or from a specified date
- Restore Library (RSTLIB) restores a collection or library
- Restore Object (RSTOBJ) restores one or more objects (including database tables and views)

For example, if two dealerships were merging, the save and restore commands could be used to save collections and tables for one relational database, which are then restored on the remaining system's relational database. To accomplish this an administrator would:

1. Use the SAVLIB command on System A to save a collection or use the SAVOBJ command on system A to save a table.
2. Specify whether the data is saved to a save file, which can be distributed using SNADS, or saved on tape or diskette.
3. Distribute the save file to System B or send the tape or diskette to System B.
4. Use the RSTLIB command on System B to restore a collection or use the RSTOBJ command on System B to restore a table.

A consideration when using the save and restore commands is the ownership and authorizations to the restored object. A valid user profile for the current object owner should exist on the system where the object is restored. If the current owner's profile does not exist on this system, the object is restored under the QDFTOWN default user profile. User authorizations to the object are limited by the default user profile parameters. A user with QSECOFR authority must either create the original owner's profile on this system and make changes to the restored object ownership, or specify new authorizations to this object for both local and remote users.

For more information about the save and restore commands, see the *Advanced Backup and Recovery Guide*.

Moving a Database to an AS/400 System from a Non-AS/400 System

You may need to move a file from another IBM system to an AS/400 system or from a non-IBM system to the AS/400 system. This section lists alternatives for moving data to an AS/400 system from a non-AS/400 system. However, you must refer to manuals supplied with the other system or identified for the application for specific instructions on its use.

Moving Data from Another IBM System

There are a number of methods you can use to move data from another IBM system to an AS/400 system. These methods include the following:

- A high-level language program can be written to extract data from another system. A corresponding program for the AS/400 system can be used to load data to the system.
- For systems supporting other DRDA implementations, you can use SQL functions to move data, subject to DRDA restrictions. For a discussion of ways to bypass the restrictions, see "Application Considerations" on page 2-4.
- Data can be extracted from tables and files on the other system and sent to the AS/400 system on tape or diskette or over communications lines.
  - From a DB2 database, a sample program called DSNTUIL, supplied with the database manager can be used to extract data from file or tables.
  - From an SQL/DS database, the Database Services Utility portion of the database manager can be used to extract data.
  - From both DB2 and SQL/DS databases, Data Extract (DXT) can be used to extract data.
- Data copied to tape or diskette can be handled as follows:
  - From DB2 or SQL/DS databases, standard tape management techniques can be used to write data to a tape. The AS/400 system uses the Copy From Tape (CPYFRMTAP) command to load data from tape.
  - From an Extended Services Database Manager database, standard copy operations to diskette can be used to write data to a diskette. The AS/400 system uses the Copy From Diskette (CPYFRMDKT) command to load data from a diskette.

  For more information on using tape and diskette devices with the AS/400 system, see the *Guide to Programming for Tape and Diskette*.

- Data sent over communications lines can be handled through SNADS support on the AS/400 system. SNADS support transfers network files for BSCEL and SNUF protocols in addition to the APPC or APPN protocols used for distributed relational database processing.
  - From an MVS system, data can be sent to the AS/400 system using TSO XMIT functions. The AS/400 system uses the WRKNETF or RCVNETF command to receive a network file.
  - From a VM system, data can be sent to the AS/400 system using SENDFILE functions. The AS/400 system uses the WRKNETF or RCVNETF command to receive a network file.
  - From an OS/2 system, data can be sent to the AS/400 system through the Extended Services Communications Manager functions and utilities or programs such as PC Support/400*, a separately orderable IBM product.

    Data can also be sent over communications lines that do not support SNADS, such as asynchronous communications. File transfer support (FTS), a utility that is part of the OS/400 licensed program, can be used to send and receive data. For more information about working with communications and communications files see the *ICF Programmer's Guide*.

Moving Data from a Non-IBM System

You can copy files or tables to tape or diskette from the other system and load these files on an AS/400 system if the data and media format meets AS/400 formats and requirements.

Vendor independent communications functions are also supported through two separately licensed AS/400 programs.

Peer-to-peer connectivity functions for both local and wide area networks is provided by the Transmission Control Protocol/Internet Protocol (TCP/IP). The File Transfer Protocol (FTP) function of the AS/400 TCP/IP Connectivity Utilities/400 licensed program allows you to receive many types of files, depending on the capabilities of the remote system. For more information, see the *TCP/IP Guide*.

The OSI File Services/400 licensed program (OSIFS/400) provides file management and transfer services for open systems interconnection (OSI) networks. OSIFS/400, with the prerequisite licensed program OSI Communications Subsystem/400, connects the AS/400 system to remote IBM or non-IBM systems that conform to OSI file transfer, access, and management (FTAM) standards.

OSIFS/400 provides either an interactive interface or an application programming interface (API) to copy or move files from a remote system to a local AS/400 system. For more information, see the *OSI Communications Subsystem Programming and Concepts Guide*.

Chapter 6. Distributed Relational Database Administration and Operation Tasks

As an administrator for a distributed relational database, you are responsible for work being done on several systems. Work that originates on your local system as an application requester (AR) can be monitored in the same way that any other work is monitored on an AS/400 system. When you are tracking units of work being done on the local system as an application server (AS), you use the same tools but look for different kinds of information.

This chapter discusses ways that you can administer the distributed relational database work being done across a network. Most of the commands, processes, and other resources discussed here do not exist just for distributed relational database use, they are tools provided for the operation of any AS/400 system. All administration commands, processes and resources discussed here are included with the OS/400 program, along with all its database management functions.

Monitoring Relational Database Activity

You can rely on three control language (CL) commands to give you a view of work on an AS/400 system. They are the Work with Job (WRKJOB), Work with User Jobs (WRKUSRJOB), and Work with Active Jobs (WRKACTJOB) commands. All three provide similar information but in different ways. The WRKJOB command gives you information specific to a job if you know the job name or the job from which you enter the WRKJOB command. The WRKUSRJOB command provides you with more detailed information on a job if you know the user profile under which the job is running. The WRKACTJOB command provides the most general look at work being done on the system. It shows all jobs that are currently running on the system and some statistics about each one.

Working with Jobs

The WRKJOB command presents the Work with Job menu. This menu allows you to select options to work with or to change information related to a specified job. Enter the command without any parameters to get information about the job you are currently using. Specify a job to get the same information pertaining to it by entering its name in the command like this:

```
WRKJOB JOB(job-number/user-ID/job-name)
```

You can get the information provided by the options on the menu whether the job is on a job queue, output queue, or active. However, a job is not considered to be in the system until all of its input has been completely read in. Only then is an entry placed on the job queue. The options for the job information are:

- Job status attributes
- Job definition attributes
- Spooled file information

Information about the following options can be shown only when the job is active:

- Job run attributes
- Job log information
- Program stack information

- Job lock information
- Library list information
- Open file information
- File override information
- Commitment control status
- Communications status

Option 10 (Display job log) gives you information about an active job or a job on a job queue. For jobs that have ended you can usually find the same information by using option 4 (Work with spooled files). This presents the Work with Spooled Files display, where you can use option 5 to display the file named QPJOBLOG if it is on the list.

Working with User Jobs

If you know the user profile (user name) being used by a job, you can use the WRKUSRJOB command to display or change job information. Enter the command without any parameters to get a list of the jobs in the system with your user profile. You can specify any user and the job status to shorten the list of jobs by entering its name in the command like this:

```
WRKUSRJOB USR(KCDBA)
```

The Work with User Jobs display appears with names and status information of user jobs running in the system (*ACTIVE), on job queues (*JOBQ), or on an output queue (*OUTQ). The following display shows the active and ended jobs for the user named KCDBA:

```
                          Work with User Jobs                      KC105
                                                          03/29/92  16:15:33
Type options, press Enter.
  2=Change   3=Hold   4=End   5=Work with   6=Release   7=Display message
  8=Work with spooled files   13=Disconnect

Opt   Job          User         Type      -----Status------   Function
__    KC000        KCDBA        CMNEVK    OUTQ
__    KC000        KCDBA        CMNEVK    OUTQ
__    KC000        KCDBA        CMNEVK    OUTQ
__    KC000        KCDBA        CMNEVK    OUTQ
__    KC000        KCDBA        CMNEVK    ACTIVE
__    KC0001       KCDBA        CMNEVK    ACTIVE              *  -PASSTHRU
__    KC0001       KCDBA        INTER     ACTIVE              CMD-WRKUSRJOB




                                                                      Bottom
Parameters or command
===>
F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F12=Cancel
F21=Select assistance level
```

This display lists all the jobs in the system for the user, shows the status specified (*ALL in this case), and shows the type of job. It also provides you with eight options (2 through 8 and 13) to enter commands for a selected job. Option 5 presents the Work with Job display described above.

Working with Active Jobs

Use the WRKACTJOB command if you want to monitor the jobs running for several users or if you are looking for a job and you do not know the job name or the user ID. When you enter this command, the Work with Active Jobs display appears. It shows the performance and status information for jobs that are currently active on the system. All information is gathered on a job basis and grouped by subsystem.

The display below shows the Work with Active Jobs display on a typical day at the KC105 system:

```
                          Work with Active Jobs                    KC105
                                                        03/29/92  16:17:45
CPU %:   41.7     Elapsed time:   04:37:55     Active jobs:   42

Type options, press Enter.
  2=Change   3=Hold   4=End   5=Work with   6=Release   7=Display message
  8=Work with spooled files   13=Disconnect ...

Opt  Subsystem/Job  User        Type  CPU %  Function        Status
__   QBATCH         QSYS        SBS      .0                 DEQW
__   QCMN           QSYS        SBS      .0                 DEQW
__     KC000        KCDBA       EVK      .0  *  -PASSTHRU    EVTW
__     KC0001       KCDBA       EVK      .0  *  -PASSTHRU    EVTW
__     MP000        MPPRG       EVK      .0  *  -PASSTHRU    EVTW
__   QINTER         QSYS        SBS      .0                 DEQW
__     DSP01        CLERK1      INT      .0  CMD-STRSQL      DSPW
__     DSP02        CLERK2      INT      .0  *  -CMDENT      DSPW

                                                                More...
Parameters or command
===>
F3=Exit      F5=Refresh   F10=Restart statistics   F11=Display elapsed data
F12=Cancel   F23=More options   F24=More keys
```

When you press F11 (Display elapsed data), the following display is provided to give you detailed status information.

```
                          Work with Active Jobs                    KC105
                                                        03/29/92  16:17:45
CPU %:   41.7     Elapsed time:   04:37:55     Active jobs:   42

Type options, press Enter.
  2=Change   3=Hold   4=End   5=Work with   6=Release   7=Display message
  8=Work with spooled files   13=Disconnect ...
                                          --------Elapsed---------
Opt  Subsystem/Job  Type  Pool  Pty      CPU  Int    Rsp  AuxIO  CPU %
__   QBATCH         SBS      2    0      4.4                108     .0
__   QCMN           SBS      2    0     20.7                668     .0
__     KC000        EVK      2   50       .1                  9     .0
__     KC0001       EVK      2   50       .1                  9     .0
__     MP000        EVK      2   50       .1                 14     .0
__   QINTER         SBS      2    0      7.3                  4     .0
__     DSP01        INT      2   20       .1                  0     .0
__     DSP02        INT      2   20       .1                  0     .0

                                                                More...
Parameters or command
===>
F3=Exit      F5=Refresh         F10=Restart statistics   F11=Display status
F12=Cancel   F23=More options   F24=More keys
```

The Work with Active Jobs display gives you information about job priority and system usage as well as the user and type information you get from the Work with User Jobs display. You also can use any of 11 options on a job (2 through 11 and 13), including option 5, which presents you with the Work with Job display for the selected job.

Using the Job Log

Every job on the AS/400 system has a job log that contains information related to requests entered for a job. The information in a job log includes:

- Commands that were used by a job
- Messages that were sent and not removed from the program message queues
- Commands in a CL program if the program was created with LOGCLPGM(*JOB) and the job specifies LOGCLPGM(*YES) or the CL program was created with LOGCLPGM(*YES)

At the end of the job, the job log can be written to a spooled file named QPJOBLOG and the original job log is deleted. You can control what information is written in the job log by specifying the LOG parameter of a job description.

The way to display a job log depends on the status of the job. If the job has ended and the job log is not yet printed, find the job using the WRKUSRJOB command, then select option 8 (Display spooled file). Find the spooled file named QPJOBLOG and select option 5 (Display job log). You can also display a job log by using the WRKJOB command and other options on the Work with Job display.

If the batch or interactive job is still active, or is on a job queue and has not yet started, use the WRKUSRJOB command to find the job. The WRKACTJOB command is used to display the job log of active jobs and does not show jobs on job queues. Select option 5 (Work with job) and then select option 10 (Display job log).

To display the job log of your own interactive job, do one of the following:

- Enter the Display Job Log (DSPJOBLOG) command.
- Enter the WRKJOB command and select option 10 (Display job log) from the Work with Job display.
- Press F10 (Display detailed messages) from the Command Entry display to display messages that are shown in the job log.

When you use the DSPJOBLOG command, you see the Job Log display. This display shows program names with special symbols, as follows:

`>>` The running command or the next command to be run. For example, if a CL or high-level language program was called, the call to the program is shown.

`>` The command has completed processing.

`. .` The command has not yet been processed.

`?` Reply message. This symbol marks both those messages needing a reply and those that have been answered.

Locating Distributed Relational Database Jobs

When you are looking for information about a distributed relational database job on an AR and you know the user profile that is used, you can find that job by using the WRKUSRJOB command. You can also use this command on the AS, but be aware that the user profile on the AS may be different from that used by the AR.

If there are several jobs listed for the specified user profile, enter option 5 (Work with job) to get the Work with Job display. From this display, enter option 10 (Display job log) to see the job log. The job log shows you whether this is a distributed relational database job and, if it is, to which remote system the job is connected. Page through the job log looking for the message:

CPI9150 DDM job started.

The second level text for message CPI9150 contains the job name for the AS job.

If you are on the AS and you do not know the job name, but you know the user name, use the WRKUSRJOB command. If you do not specify a user, the command returns a list of the jobs under the user profile you are using. On the Work with User Jobs display, use these columns to help you identify the AS jobs.

1 The job type column shows jobs with the type listed as CMNEVK for communications jobs.

2 The status column shows if the job is active or completed. Depending on how the system is set up to log jobs, you may see only active jobs.

3 The job column provides the job name. The job name on the AS is the same as the device being used.

```
                          Work with User Jobs                      KC105
                                                        03/29/92  16:15:33
Type options, press Enter.
  2=Change   3=Hold   4=End   5=Work with   6=Release   7=Display message
  8=Work with spooled files   13=Disconnect

Opt   Job          User         Type      -----Status------   Function
__    KC000        KCDBA        CMNEVK    OUTQ
__    MP000        KCDBA        CMNEVK    OUTQ
__    MP000        KCDBA        CMNEVK    OUTQ
__    KC000        KCDBA        CMNEVK    OUTQ
__    KC000        KCDBA        CMNEVK    ACTIVE
__    KC0001       KCDBA        CMNEVK    ACTIVE              *  -PASSTHRU
__    KC0001       KCDBA        INTER     ACTIVE              CMD-WRKUSRJOB
      3                          1         2
```

If you are looking for an active AS job and do not know the user name, the WRKACTJOB command gives you a list of those jobs for the subsystems active on the system. The following example shows you some items to look for:

```
                          Work with Active Jobs                    KC105
                                                        03/29/92  16:17:45
  CPU %:   41.7     Elapsed time:   04:37:55     Active jobs:   102
  Opt  Subsystem/Job  User        Type  CPU %  Function        Status
       QBATCH         QSYS        SBS      .0                  DEQW
4 _    QCMN           QSYS        SBS      .0                  WDEQ
__       KC000        KCDBA       EVK      .0  *  -PASSTHRU    EVTW
__       KC0001       KCCLERK     EVK      .0  *               EVTW
__       MP000        MPSUP       EVK      .0  *  -PASSTHRU    EVTW
         5                        6
```

4 Search the subsystem that is set up to handle the AS jobs. In this example, the subsystem for AS jobs is QCMN.

5 For AS jobs, the job name is the device name of the device created for AS use.

6 The job type listed is normally EVK, started by a program start request.

When you have located a job that looks like a candidate, enter option 5 to work with that job. Then select option 10 from the Work with Job Menu to display the job log. Distributed database job logs for jobs that are accessing the AS from an AS/400 application requester contain a statement near the top that reads:

CPI3E01 Local relational database accessed by *(system name)*.

After you locate a job working on the AS, you can also trace it back to the AR if the AR is an AS/400 system. While looking at the job, log place the cursor on the message that reads:

CPI9152 Target DDM job started by source system.

When you press the help key, the detailed message for the statement appears. The source system job named is the job on the AR that caused this job.

Operating Remote AS/400 Systems

As an administrator in a distributed relational database you may sometimes have to operate a remote AS/400 system. For example, you may have to start or stop a remote system. The AS/400 system provides several ways to do this either in real time or at previously scheduled times. More often, you may need to perform certain tasks on a remote system as it is operating. The two primary ways that you can do this is by using display station pass-through and the Submit Remote Command (SBMRMTCMD) command.

Starting and Stopping Other Systems Remotely

The AS/400 system provides options that help you ensure that a remote system is operating when it needs to be. Of course, the simplest way to ensure that a remote system is operating is to allow the remote location to power on their system to meet the distributed relational database requirements. But, this is not always possible. You can set up an automatic power-on and power-off schedule or enable a remote power on to a remote system. See the *Operator's Guide* for more information on system values that control IPL, power- on and power-off schedules, and remote IPLs.

Using Pass-Through

While a remote system is operating you can work on it using display station pass-through. Chapter 3, "Communications for an AS/400 Distributed Relational Database" describes how to set up and automatically configure pass-through. The system you originate the pass-through session from is called the source system; the system you are accessing is called the target. When you are using APPN, you may be routed through several other systems, depending on the network nodes available, but you do not have any control over operation on those systems. After you sign on a target system, you can use that system as though your work station were directly attached to it.

Use the Start Pass-Through (STRPASTHR) command to connect to another system. You can use several different methods to designate a target system. Since the Spiffy Corporation has chosen to automatically configure devices for pass-through throughout the network, the following command is all that must be entered to connect to one of the dealerships in the Kansas City region.

```
STRPASTHR RMTLOCNAME(KC105)
```

The RMTLOCNAME is the name of a location defined at the target system to which you wish to pass-through and is the same remote location specified for this system in the relational database directory. Depending on the configuration of your system, you may also need to specify a list of devices. If the entire route from the source system to the target system consists of systems using APPN and both the target and source systems are located in the same APPN network, the RMTLOCNAME is the only parameter you need to specify. If the target and source systems are located in different APPN networks, the target system network ID must be indicated in the remote network identifier (RMTNETID) parameter.

You can return to the source system by using the Transfer Pass-Through (TFRPASTHR) command. This command can be helpful when you are working on a large job on a target system and want to return to the source system without ending the target system session. This command does not end your pass-through session. Type

```
TFRPASTHR
```

on the command line and press the Enter key when you are working on the target system to return to the source system.

The options available on the TFRPASTHR command are:

TOJOB(*SRC)
This option suspends the job on the target system and transfers the user back to the source system. The program specified by the system request program parameter on the STRPASTHR command is given control. When the system request program ends, pass-through returns you to the job on the target system from which you transferred.

TOJOB(*ALT)
This option suspends the job on the target system and transfers the user back to the source system. The alternate job at the source system is given control. After returning to the alternate job at the source system, the Transfer Secondary Job (TFRSECJOB) command can be used to transfer from the alternate job to the original job, which gives control to the target system. Control is also given to the target system when the alternate job ends.

You can return to the source system by using the End Pass-Through (ENDPASTHR) command. Use this command when you are done working with the job on the target system and want to return to the source system. This command signs you off the target job and ends the pass-through session. Pass-through jobs at intermediate nodes end and control returns to the source system.

Submit Remote Command (SBMRMTCMD) Command

The Submit Remote Command (SBMRMTCMD) command submits a CL command using Distributed Data Management (DDM) support to run on the target system. You first need to create a DDM file. The remote location information of the DDM file is used to determine the communications line to be used. Thus, it identifies the target system that is to receive the submitted command. The remote file associated with the DDM file is not involved when the DDM file is used for submitting commands to run on the target system. See "Setting Up DDM Files" on page 5-10 for information on creating DDM files.

The SBMRMTCMD command can submit any CL command that can run in both the batch environment and via the QCAEXEC system program; that is, the command has values of *BPGM *and* *EXEC specified for the ALLOW attribute. You can display the ALLOW attributes by using the Display Command (DSPCMD) command.

The primary purpose of the SBMRMTCMD command is to allow a source system user or program to perform file management operations and file authorization activities on objects located on a target system. A secondary purpose of this command is to allow a user to perform nonfile operations (such as creating a message queue) or to submit user-written commands to run on the target system. The CMD parameter allows you to specify a character string of up to 2000 characters that represents a command to be run on the target system.

You must have the proper authority on the target system for the CL command being submitted and for the objects that the command is to operate on. If the source system user has the correct authority to do so (as determined in a target system user profile), the following actions are examples of what can be performed on remote files using the SBMRMTCMD command:

- Grant or revoke object authority to remote tables
- Verify tables or other objects
- Save or restore tables or other objects

Although the command can be used to do many things with tables or other objects on the remote system, using this command for some tasks is not as efficient as other methods on the AS/400 system. For example, you could use this command to display the file descriptions or field attributes of remote files, or to dump files or other objects, but the output remains at the target system. To display remote file descriptions and field attributes at the source system, a better method is to use the Display File Description (DSPFD) and Display File Field Description (DSPFFD) commands with SYSTEM(*RMT) specified, and specify the names of the DDM files associated with the remote files.

See the *DDM Guide* for lists of CL commands you can submit and restrictions for the use of this command.

SBMRMTCMD Example

You can use the SBMRMTCMD command to grant temporary authority to an object. For example, a programmer on one system would like to call a program on another system and watch its operation for debugging purposes. The distributed relational database administrator could get authority to the program to be checked for the local programmer by entering the following command:

```
SBMRMTCMD  CMD('GRTOBJAUT OBJ(SPIFFY/PARTS1) OBJTYPE(*PGM)
   USER(MPSUP) AUT(*USE)') DDMFILE(TEST/KC105TST)
```

This submitted command grants *USE authority to the user MPSUP to the object PARTS1. PARTS1 is a program that exists on the system identified by the DDM file named KC105TST on the local system. The authority is granted if the distributed relational database administrator has authority to use the GRTOBJAUT command on the remote system named in the KC105TST DDM file.

Monitoring Relational Databases in the Network

Much of the administration of a distributed relational database involves working with communications networks, controllers, lines, and devices. Working with the configuration status of these devices is a regular task in a network. Another important task is controlling DDM conversations throughout the distributed relational database network. Sometimes you need to check on the availability of resources used by distributed relational database applications. This section describes the methods you can use to perform these tasks.

Working with Configuration Status

You can monitor and change the communications configurations in a distributed relational database network several ways. You can display the configuration status for specific jobs using the Display Configuration Status display. You can monitor critical communications resources periodically using a CL program with the Retrieve Configuration Status command. The Work with Configuration Status display allows you to see and change the status of communication devices. You can also use CL commands to work with and change communications modes and sessions.

Displaying Communications Status Information

You can get communications status information about a distributed relational database job by using the Display Job (DSPJOB) or WRKJOB command. You can select option 17 from either the Display Job or Work with Job display when communications identifiers are active. You can also use a CL command to display jobs or work with jobs like this:

```
DSPJOB JOB(job-number/user-ID/job-name) OPTION(*CMNSTS)
```

The *CMNSTS option specifies the command to get the communications status information.

If you specify OUTPUT(*PRINT) on the DSPJOB command, the communications status information is printed. If you display the information, it appears in the following format:

```
                        Display Communications Status
                                                                  System:   KC105
Job:   KC105          User: KCDBA           Number:   007100

Type options, press Enter.
  5=Display

     -------- Communications --------  --------Number of Operations---------
Opt  Identifier        Method              Output        Input        Other

__   _______________   _____________
__   DDMDEVICE         APPC-ICF                 0            0            1
__   DDMDEVICE         APPC-ICF               150           20            3

                                                                        Bottom
F3=Exit        F5=Refresh   F11=Display operations   F12=Cancel   F16=Job menu
F17=Top        F18=Bottom
```

This display shows the general status for the communications connections the specified job has with a remote system. The identifier for a distributed relational database job is the program device name specified in the application for an active (acquired) intersystem communications function (ICF) session. The method for distributed relational database applications is always advanced program-to-program communications (APPC)-ICF. The number of operations is the count of ICF operations issued by the program. The type of operations are:

Output
The number of successful ICF write operations. Successful write operations do not include write operations in which another function, such as invite, is also specified and the length is 0. Cancel, cancel-invite, fail, negative-response, and request-write operations are not counted. Successful combined write/read operations are counted.

Input
The number of successful ICF read operations that received data.

Other
The number of all other high-level language operations such as open/acquire, acquire, release, and close.

When you press F11 (Display number of operations), the following display appears showing the current or last ICF operation issued by the program.

```
                        Display Communications Status
                                                                  System:   KC105
Job:   KC105          User: KCDBA           Number:   007100

Type options, press Enter.
  5=Display

     -------- Communications --------
Opt  Identifier        Method          Operation

__   _______________   _____________
__   DDMDEVICE         APPC-ICF        EVK
__   DDMDEVICE         APPC-ICF        RCV
```

The operation that is displayed is dependent on the communications method that is used for the communications identifier. The ICF function codes are the same as the function codes used by the trace ICF function and are described in Figure 6-1 on page 6-11.

Figure 6-1. ICF Function Codes on the Display Communications Status Display

Function Code	Description	Function Code	Description
ACQ	Acquire	AWT	Allow-Write
CFM	Confirm	CLS	Close
CNI	Cancel-Invite	CNL	Cancel
DET	Detach	EGP	End-of-Group
EOA	End-of-Session-Abnormal	EOS	End-of-Session
EVK	Evoke	FAL	Fail
FMH	Function-Management-Header	FRC	Force-Data
GTA	Get-Attributes	INV	Invite
NRP	Negative-Response	OPN	Open with Acquire-Program-Device
RCF	Respond-to-Confirm	RCV	Receive
REL	Release	RFI	Read-From-Invited-Program-Devices
RST	Restore	RWT	Request-to-Write
SDV	Subdevice-Selection	SND	Send
SPD	Suspend	TMR	Timer

When you selection option 5 (Display) for an identifier used for distributed relational database, the following display appears with more detail about the session:

```
                    Display APPC ICF Session

Program device . . . . . . . . . . :   DDMDEVICE
Remote location  . . . . . . . . . :   KC105
Device . . . . . . . . . . . . . . :   KC105
Local location . . . . . . . . . . :   KC000
Mode . . . . . . . . . . . . . . . :   BLANK
Remote network identifier. . . . . :   APPN
ICF file . . . . . . . . . . . . . :   QCNDDMF
  Library  . . . . . . . . . . . . :     QSYS
```

Retrieving Configuration Status

You can write a CL program to monitor communication devices periodically, to check on the status of the critical devices in the network. The Retrieve Configuration Status (RTVCFGSTS) command allows you to retrieve configuration status from four configuration objects (the network interface, line, controller, and device), and place the configuration status into a CL program. This command is used only in CL programs; you cannot use it from the AS/400 command line.

Communications applications can react quickly and easily to a configuration status with direct access to the objects. For example, an application can verify that the communications device used to access a remote database is active before

attempting a connect. If it is not active, it could vary on the device. The following program example shows how this command can be used:

```
DCL VAR(&STSCODE)  TYPE(*DEC)    LEN(5 0)

RTVCFGSTS CFGD(ND01) CFGTYPE(*LIN)
          STSCDE(&STSCODE)
IF COND(&STSCODE *NE 60) THEN (DO)
.
.
.
END DO
```

The name of the configuration description is specified by the CFGD parameter. The CFGTYPE is a required parameter and can specify any one of the following:

***NWI** Network interface configuration (use this for ISDN networks only)
***LIN** Line configuration
***CTL** Controller configuration
***DEV** Device configuration

Using the Work with Configuration Status Display

You can use the Work with Configuration Status (WRKCFGSTS) command to view and change communications status for the communications devices configured in the network. For example, the command a distributed relational database administrator would issue for checking on the status of the system named KC105 is:

```
WRKCFGSTS CFGTYPE(*DEV) CFGD(*LOC) RMTLOCNAME(KC105)
```

The following display appears, showing a list of device descriptions, their status, and the job using the device.

```
                      Work with Configuration Status            KC000
                                                      03/29/92 10:47:31
Position to . . . . .  __________   Starting characters

Type options, press Enter.
  1=Vary on   2=Vary off   5=Work with job    8=Work with description
  9=Display mode status ...

Opt  Description       Status              -------------Job--------------
 _   KC105            ACTIVE                KC000       KCDBA      001128









                                                                    More...
Parameters or command
===>_____________________________________________________________________
F3=Exit   F4=Prompt   F11=Display Types   F12=Cancel   F23=More options
F24=More keys
```

When you press F11 (Display Types), the TYPE column appears to show the kind of communications device.

```
                      Work with Configuration Status           KC000
                                                          03/29/92 10:47:31
Position to . . . . .  __________   Starting characters

Type options, press Enter.
  1=Vary on   2=Vary off   5=Work with job    8=Work with description
  9=Display mode status ...

Opt  Description       Status              TYPE

 _   KC105            ACTIVE              *DEV

                                                                      More...
Parameters or command
===>_____________________________________________________________________
F3=Exit   F4=Prompt   F11=Display Types   F12=Cancel   F23=More options
F24=More keys
```

The Work with Configuration Status display shows status information for selected network interfaces, lines, controllers, and devices. Attached configuration descriptions are indented under the object they are attached to, as shown on the previous display.

If the status is displayed for a specific controller, upline line descriptions as well as downline device descriptions are displayed. Status displays for any group of controllers or devices, such as all controllers or controllers whose names start with a specified (generic) character string, show only downline attachments. The status for a remote location shows devices and modes for the specified location.

From the Work with Configuration Status display you can use one of 11 options to change the status or view more information.

Figure 6-2 shows the possible status for each communications configuration object.

Figure 6-2 (Page 1 of 2). Status for Network Interfaces, Lines, Controllers, and Devices

Status Name	Status Description
Varied Off (VARIED OFF)	The system is not using the description.
Vary Off Pending (VARY OFF PENDING)	The description is being varied off. During this time, the system ends the operations that manage the resource or communicate with data circuit-terminating equipment (DCE).
Vary On Pending (VARY ON PENDING)	The description is being varied on. During this time, the system begins the operations that manage the resource, download licensed internal code to an input and output processor, and communicate with the DCE.
Varied On (VARIED ON)	The functions that manage the network interface, line, controller, or device have been put into place by the system.
Connect Pending (CONNECT PENDING)	Valid for switched SDLC lines. The line is in this status while waiting for the switched connection to be established.

Figure 6-2 (Page 2 of 2). Status for Network Interfaces, Lines, Controllers, and Devices

Status Name	Status Description
Sign On Display (SIGN ON DISPLAY)	Valid only for display devices. Either the system is preparing the device to receive the sign-on display, sending the sign-on display, or the sign-on display is at the display station.
Active (ACTIVE)	The object is successfully placed in VARIED ON status. • For network interfaces, one or more attached lines is in a VARY ON PENDING or higher status. • For lines, one or more attached controllers is in a VARY ON PENDING or higher status. • For controllers, one or more attached devices is in a VARY ON PENDING or higher status. • For devices, active status varies, depending on the type of device.
Held (HELD)	Valid only for device descriptions. The user or system held the communications device to prevent it from communicating. Use the Release Communications Device (RLSCMNDEV) command to release the device.
Recovery Pending (RCYPND)	Error recovery is pending for the network interface, line, controller, or device. A message indicating what error occurred appears on the QSYSOPR message queue.
Recovery Cancel (RCYCNL)	Error recovery is canceled for the network interface, line, controller, or device. An error occurred and the operator gave a C (cancel error recovery) reply to a message, or the operator used one of the following commands to end the error recovery process: End Network Interface Recovery (ENDNWIRCY), End Line Recovery (ENDLINRCY), End Controller Recovery (ENDCTLRCY), or End Device Recovery (ENDDEVRCY).
Failed (FAILED)	An error occurred for the network interface, line, controller, or device that can be recovered only by varying off and on again.
Diagnostic Mode (DIAGNOSTIC MODE)	The network interface, line, controller, or device is being used by problem analysis procedures to diagnose problems. The resource cannot be used by other users.
Damaged (*DAMAGED)	The network interface, line, controller, or device description is damaged. This is a system error condition. Information showing when this damage occurred exists in the history log (QHST) or in the vertical licensed internal code (VLIC) log. You must delete the description and create it again before it can be used. The VLIC log information can be used when reporting a problem to IBM service.
Locked (*LOCKED)	The actual status of the resource cannot be determined because another job has an exclusive lock on the description. Try again at a later time or use the Work with Object Lock (WRKOBJLCK) command to determine which job has the lock on the description.
Unknown (*UNKNOWN)	The status indicator of the description cannot be determined. This is a system error condition. Use the Dump Object (DMPOBJ) command to get the attributes and contents of the description to a spooled printer file and contact your IBM representative or your IBM-approved remarketer.

Controlling Modes and Maximum Sessions

All communications devices are resources that are shared throughout the system. This means that access to the AR may be denied because limited system resources are already in use. An AR uses a DDM conversation to access the AS. A communications session can be used by only one DDM conversation. The communications mode description establishes the number of sessions that can be handled by the mode. The mode used by an AR is the one defined in the relational database directory.

The system default is eight maximum sessions for a configured mode. However, a mode can have up to 512 sessions. If an AR cannot connect the AS, you may want to work with the mode status or change the maximum number of sessions allowed.

You can use the following commands to display the mode status, and to start and end modes with remote systems:

- Start Mode (STRMOD)
- Display Mode Status (DSPMODSTS)
- End Mode (ENDMOD)

The *APPC Programmer's Guide* contains more information about using these commands.

The Change Session Maximum (CHGSSNMAX) command dynamically changes the maximum number of sessions the local location allows to a mode. The specified remote location and mode must be active. If the current number of active sessions is greater than the maximum number specified on the command, no new sessions are created until the number of active sessions becomes less than the number specified in the command parameter. If the current number of active sessions is less than the maximum number specified, sessions cannot be established until the jobs requiring them begin.

See the *Communications Management Guide* for more on how this command works.

Controlling DDM Conversations

When an AR connects to an AS it uses a DDM conversation. The conversation is established with the SQL CONNECT statement from the AR, but only if a conversation using the same remote location values does not already exist for the AR job. DDM conversation can be in one of three states: active, unused, or dropped.

A DDM conversation used by distributed relational database is active while the AR is connected to the AS. After the AR connects to a different relational database, the DDM conversation then either becomes unused or it is dropped. If a DDM conversation is unused the SNA conversation to the other system is maintained by the DDM communications manager and marked as unused. If a DDM conversation is dropped, the DDM communications manager will end the SNA conversation.

The DDMCNV job attribute determines whether DDM conversations which are no longer active become unused or are dropped. If the job attribute value is *KEEP the DDM conversations will become unused and if the value is *DROP the conversations will be dropped. Because of the way DRDA defines the process for connecting to a different database, when using distributed relational database to an unlike system the DDM conversations are always dropped.

Using a DDMCNV job attribute of *KEEP is desirable when connections to remote relational databases are frequently changed. A value of *DROP is desired when the cost of maintaining the SNA conversations is high and the connections do not change frequently.

If a DDM conversation is also being used to operate on remote files through DDM, the conversation will remain active until the following conditions are met:

- All the files used in the conversation are closed and unlocked
- No other DDM-related functions are being performed
- No DDM-related function has been interrupted (by a break program, for example)
- The End Commitment Control (ENDCMTCTL) command was issued (if commitment control was used with a DDM file)
- An AR job is no longer connected to the AS

Regardless of the value of the DDMCNV job attribute, conversations are dropped at the end of a job routing step, at the end of the job, or when the job initiates a Reroute Job (RRTJOB) command. Unused conversations within an active job can also be dropped by the Reclaim DDM Conversations (RCLDDMCNV) or Reclaim Resources (RCLRSC) command. Errors, such as communications line failures, can also cause conversations to drop.

The DDMCNV parameter is changed by the Change Job (CHGJOB) command and is displayed by Display Job (DSPJOB) command with OPTION(*DFNA). Also, you can use the Retrieve Job Attributes (RTVJOBA) command to get the value of this parameter and use it within a CL program.

Reclaiming DDM Resources

The Reclaim DDM Conversations (RCLDDMCNV) command reclaims all application conversations that are not currently being used by a source job, even if the DDMCNV attribute value for the job is *KEEP. The command allows you to reclaim unused DDM conversations without closing all open files or doing any of the other functions performed by the Reclaim Resources (RCLRSC) command.

The RCLDDMCNV command applies to the DDM conversations for the job on the AR in which the command is entered. There is an associated AS job for the DDM conversation used by the AR job. The AS job ends automatically when the associated DDM conversation ends.

Although this command applies to all DDM conversations used by a job, using it does not mean that all of them will be reclaimed. A conversation is reclaimed only if it is not being actively used. If commitment control is used, a COMMIT or ROLLBACK operation may have to be done before a DDM conversation can be reclaimed.

Displaying Objects Used by Programs

You can use the Display Program References (DSPPGMREF) command to determine which tables, data areas, and other programs are used by a program or SQL package. This information is available for SQL packages and compiled programs only. The information can be displayed, printed, or written to a database output file.

When a program or package is created, the information about certain objects used in the program or package is stored. This information is then available for use with the Display Program References (DSPPGMREF) command. Information retrieved can include:

- The name of the program or package and its text description
- The name of the library or collection containing the program or package
- The number of objects referred to by the program package
- The qualified name of the system object
- The information retrieval dates
- The object type of the referenced object

For files and tables, the record contains the following additional fields:

- The name of the file or table in the program or package (possibly different from the system object name if an override was in effect when the program or package was created)

  Note: Any overrides apply only on the AR.

- The program or package use of the file or table (input, output, update, unspecified, or a combination of these four)
- The number of record formats referenced, if any
- The name of the record format used by the file or table and its record format level identifier
- The number of fields referenced for each format

Before the objects can be shown in a program, the user must have *USE authority for the program. Also, of the libraries specified by the library qualifier, only the libraries for which the user has read authority are searched for the programs.

Figure 6-3 on page 6-18 shows the objects for which the high-level languages and utilities save information.

Figure 6-3. How High-level Languages Save Information About Objects

Language	Files	Programs	Data Areas	See Note
CL	Yes	Yes	Yes	1
COBOL/400* Language	Yes	Yes	No	2
PL/I	Yes	Yes	N/A	2
RPG/400* Language	Yes	No	Yes	3
SQL/400 Language	Yes	N/A	N/A	4

Notes:

1. All system commands that refer to files, programs, or data areas specify in the command definition that the information should be stored when the command is compiled in a CL program. If a variable is used, the name of the variable is used as the object name (for example, &FILE). If an expression is used, the name of the object is stored as *EXPR. User-defined commands can also store the information for files, programs, or data areas specified on the command. See the description of the FILE, PGM, and DTAARA parameters on the PARM or ELEM command statements in the *CL Programmer's Guide*.
2. The program name is stored only when a literal is used for the program name (this is a static call, for example, `CALL 'PGM1'`), not when a COBOL/400 identifier is used for the program name (this is a dynamic call, for example, `CALL PGM1`).
3. The use of the local data area is not stored.
4. Information about SQL packages.

The stored file information contains an entry (a number) for the type of use. In the database file output of the Display Program References (DSPPGMREF) command (built when using the OUTFILE parameter), this entry is a representation of one or more codes listed below. There can only be one entry per object, so combinations are used. For example, a file coded as a 7 would be used for input, output, and update.

Code	Meaning
1	Input
2	Output
3	Input and Output
4	Update
8	Unspecified

Display Program Reference Example

To see what objects are used by an AR program, you can enter a command such as the following:

```
DSPPGMREF PGM(SPIFFY/PARTS1) OBJTYPE(*PGM)
```

On the requester you can get a list of all the collections and tables used by a program, but you are not able to see on which relational database they are located. They may be located in multiple relational databases. The output from the command can go to a database file or to a displayed spooled file. The output looks like this:

```
File  . . . . . :   QPDSPPGM                        Page/Line   1/1
Control . . . . .                                   Columns     1 - 78
Find  . . . . . .

   3/29/92              Display Program References
DSPPGMREF Command Input
  Program . . . . . . . . . . . . . . . . . . . :   PARTS1
    Library . . . . . . . . . . . . . . . . . . :     SPIFFY
  Output  . . . . . . . . . . . . . . . . . . . :   *
  Include SQL packages  . . . . . . . . . . . . :   *YES
Program . . . . . . . . . . . . . . . . . . . . :   PARTS1
  Library . . . . . . . . . . . . . . . . . . . :     SPIFFY
  Text 'description'. . . . . . . . . . . . . . :     Check inventory for parts
  Number of objects referenced  . . . . . . . . :         3
  Object  . . . . . . . . . . . . . . . . . . . :     PARTS1
    Library . . . . . . . . . . . . . . . . . . :       SPIFFY
    Object type . . . . . . . . . . . . . . . . :       *PGM
  Object  . . . . . . . . . . . . . . . . . . . :     QSQROUTE
    Library . . . . . . . . . . . . . . . . . . :       *LIBL
    Object type . . . . . . . . . . . . . . . . :       *PGM
  Object  . . . . . . . . . . . . . . . . . . . :     INVENT
    Library . . . . . . . . . . . . . . . . . . :       SPIFFY
    Object type . . . . . . . . . . . . . . . . :       *FILE
    File name in program  . . . . . . . . . . . :
    File usage  . . . . . . . . . . . . . . . . :       Input
```

To see what objects are used by an AS SQL package, you can enter a command such as the following:

```
DSPPGMREF PGM(SPIFFY/PARTS1) OBJTYPE(*SQLPKG)
```

The output from the command can go to a database file or to a displayed spooled file. The output looks like this:

```
File  . . . . . :   QPDSPPGM                        Page/Line   1/1
Control . . . . .                                   Columns     1 - 78
Find  . . . . . .

   3/29/92              Display Program References
DSPPGMREF Command Input
  Program . . . . . . . . . . . . . . . . . . . :   PARTS1
    Library . . . . . . . . . . . . . . . . . . :     SPIFFY
  Output  . . . . . . . . . . . . . . . . . . . :   *
  Include SQL packages  . . . . . . . . . . . . :   *YES
SQL package . . . . . . . . . . . . . . . . . . :   PARTS1
  Library . . . . . . . . . . . . . . . . . . . :     SPIFFY
  Text 'description'. . . . . . . . . . . . . . :     Check inventory for parts
  Number of objects referenced  . . . . . . . . :         1
  Object  . . . . . . . . . . . . . . . . . . . :     INVENT
    Library . . . . . . . . . . . . . . . . . . :       SPIFFY
    Object type . . . . . . . . . . . . . . . . :       *FILE
    File name in program  . . . . . . . . . . . :
    File usage  . . . . . . . . . . . . . . . . :       Input
```

Dropping a Collection

Attempting to delete a collection that contains journal receivers may cause an inquiry message to be sent to the QSYSOPR message queue for the AS job. The AS and AR job wait until this inquiry is answered.

The message that appears on the message queue is:

CPA7025 Receiver (*name*) in (*library*) never fully saved. (I C)

When the AR job is waiting, it may appear as if the application is hung. Consider the following when your AR job has been waiting for a time longer than anticipated:

- Be aware that an inquiry message is sent to QSYSOPR message queue and needs an answer to proceed.
- Move the journal receivers to a different library other than the one that is being dropped.
- Have the AS reply to the message using its system reply list.

This last consideration can be accomplished by changing the job that appears to be currently hung, or by changing the job description for all AS jobs running on the system. However, you must first add an entry to the AS system reply list for message CPA7025 using the Add Reply List Entry (ADDRPYLE) command:

```
ADDRPYLE  SEQNBR(...)  MSGID(CPA7025) RPY(I)
```

To change the job description for the job that is currently running on the AS, use the SBMRMTCMD command. The following example shows how the database administrator on one system in the Kansas City region changes the job description on the KC105 system (the system addressed by the TEST/KC105TST DDM file):

```
SBMRMTCMD  CMD('CHGJOB JOB(KC105ASJOB) INQMSGRPY(*SYSRPYL)')
    DDMFILE(TEST/KC105TST)
```

You can prevent this situation from happening on the AS more permanently by using the Change Job Description (CHGJOBD) command so that any job that uses that job description uses the system reply list. The following example shows how this command is entered on the same AS:

```
CHGJOBD JOBD(KC105ASJOB)  INQMSGRPY(*SYSRPYL)
```

This method should be used with caution. Adding CPA7025 to the system reply list affects all jobs which use the system reply list. Also changing the job description affects all jobs that use a particular job description. You may want to create a separate job description for AS jobs. For additional information on creating job descriptions, see the *Work Management Guide*.

Job Accounting

The job accounting function on the AS/400 system gathers data so you can determine who is using the system and what system resources they are using. Typical job accounting details the jobs running on a system and resources used, such as use of the processing unit, printer, display stations; and database and communications functions.

Job accounting is optional and must be set up on the system. To set up resource accounting on the system you must:

1. Create a journal receiver by using the Create Journal Receiver (CRTJRNRCV) command.
2. Create the journal named QSYS/QACGJRN by using the Create Journal (CRTJRN) command. You must use the name QSYS/QACGJRN and you must have authority to add items to QSYS to create this journal. Specify the names of the journal receivers you created in the previous step on this command.
3. Change the accounting level system value QACGLVL using the Work with System Value (WRKSYSVAL) or Change System Value (CHGSYSVAL) command.

   The VALUE parameter on the CHGSYSVAL command determines when job accounting journal entries are produced. A value of *NONE means the system does not produce any entries in the job accounting journal. A value of *JOB means the system produces a job (JB) journal entry. A value of *PRINT produces a direct print (DP) or spooled print (SP) journal entry for each file printed.

When a job is started, a job description is assigned to the job. The job description object contains a value for the accounting code (ACGCDE) parameter, which may be an accounting code or the default value *USRPRF. If *USRPRF is specified, the accounting code in the job's user profile is used.

You can add accounting codes to user profiles using the accounting code parameter ACGCDE on the Create User Profile (CRTUSRPRF) command or the Change User Profile (CHGUSRPRF) command. You can change accounting codes for specific job descriptions by specifying the desired accounting code for the ACGCDE parameter on the Create Job Description (CRTJOBD) command or the Change Job Description (CHGJOBD) command.

When a job accounting journal is set up, job accounting entries are placed in the journal receiver starting with the next job that enters the system after the CHGSYSVAL command takes effect.

You can use the OUTFILE parameter on the Display Journal (DSPJRN) command to write the accounting entries to a database file that you can process.

For more information about job accounting, see the *Work Management Guide*.

Canceling Distributed Relational Database Work

Whether you are testing an application, handling a user problem, or monitoring a particular device, there are times when you may want to end work that is being done on a system. When you are using an interactive job, you normally end the job by signing off the system. There are other ways that you can cancel or discontinue jobs on the system. They depend on what kind of a job it is and what system it is on. The ways are:

- End job
- End request

End Job (ENDJOB) Command

The End Job (ENDJOB) command ends any job. The job can be active, on a job queue, or already ended. You can end a job immediately or by specifying a time interval so that end of job processing can occur.

Ending an AR job ends the job on both the AR and the AS. If the application is under commitment control, all uncommitted changes are rolled back.

End Request (ENDRQS) Command

The End Request (ENDRQS) command cancels a local operation (request) that is currently stopped at a breakpoint. This means the command cancels an AR operation or request. You can cancel a request by entering ENDRQS on a command line or you can select option 2 from the System Request menu.

If it cannot be processed immediately because a system function that cannot be interrupted is currently running, the command waits until interruption is allowed.

When a request is ended, an escape message is sent to the request processing program that is currently called at the request level being canceled. Request processing programs can monitor for the escape message so that cleanup processing can be done when a request is canceled. The static storage and open files are reclaimed for any program that was called by the request processing program. None of the programs called by the request processing program are notified of the cancel, so they have no opportunity to stop processing.

Warning: Using the ENDRQS command on an AR job may produce unpredictable results and can cause the loss of the connection to the AS.

Auditing the Relational Database Directory

Accesses to the relational database directory are recorded in the security auditing journal when either:

- The value of the system QAUDLVL is *SYSMGT.
- The value of the user AUDLVL is *SYSMGT.

With the *SYSMGT value, the system audits all accesses made with the following commands:

- Add Directory Entry (ADDRDBDIRE)
- Change Directory Entry (CHGRDBDIRE)
- Display Directory Entry (DSPRDBDIRE)
- Remove Directory Entry (RMVRDBDIRE)
- Work Directory Entry (WRKRDBDIRE)

The relational database directory is a database file (QSYS/QADBXRDBD) that can be read directly without the directory entry commands. To audit direct accesses to this file, set auditing on with the Change Object Auditing (CHGOBJAUD) command.

Chapter 7. Data Availability and Protection

In a distributed relational database environment, data availability not only involves protecting data on an individual system in the network, but also ensuring that users have access to the data across the network.

This chapter discusses tools and techniques to protect programs and data on an AS/400 system and reduce recovery time in the event of a problem. It also provides information about alternatives that ensure your network users have access to the relational databases and tables across the network when it is needed.

Recovery Support

Failures that can occur on a computer system are a system failure (when the entire system is not operating); a loss of the site due to fire, flood or similar catastrophe; or the damage or loss of an object. For a distributed relational database, a failure on one system in the network prevents users across the entire network from accessing the relational database on that system. If the relational database is critical to daily business activities at other locations, enterprise operations across the entire network can be disrupted for the duration of one system's recovery time. Clearly, planning for data protection and recovery after a failure is particularly important in a distributed relational database.

Each system in a distributed relational database is responsible for backing up and recovering its own data. Each system in the network also handles recovery procedures after an abnormal system end. However, backup and recovery procedures can be done by the distributed relational database administrator using display station pass-through for those systems with an inexperienced operator or no operator at all.

The most common type of loss is the loss of an object or group of objects. An object can be lost or damaged due to several factors, including power failure, hardware failures, system program errors, application program errors, or operator errors. The AS/400 system provides several methods for protecting the system programs, application programs, and data from being permanently lost. Depending on the type of failure and the level of protection chosen, most of the programs and data can be protected, and the recovery time can be significantly reduced. These protection methods include:

- Physical protection of the system from power failure
- System save and restore functions to ensure Structured Query Language (SQL) objects such as tables, collections, packages and relational database directories can be saved and restored
- Protection from disk related failures such as auxiliary storage pools to control where objects are stored, checksum protection for auxiliary storage pools, and mirrored protection for disk-related hardware components
- Journal management for auxiliary records of relational database changes and journaling indexes to data
- Commitment control to ensure relational database transactions can be applied or removed in a uniform manner

Uninterruptible Power Supply

Making sure your system is protected from sudden power loss is an important part of ensuring that your application server (AS) is available to an application requester (AR). An uninterruptible power supply, that can be ordered separately, protects the system from loss because of power failure, power interruptions, or drops in voltage, by supplying power to the system devices until power can be restored. Normally, an uninterruptible power supply does not provide power to all work stations. With the AS/400 system, the uninterruptible power supply allows the system to:

- Continue operations during brief power interruptions or momentary drops in voltage.
- End operations normally by closing files and maintaining object integrity.

Data Recovery after Disk Failures

Recovery is not possible for recently entered data if a disk failure occurs and all objects are not saved on tape or disk immediately before the failure. After previously saved objects are restored, the system is operational, but the database is not current.

Auxiliary storage pools (ASPs), checksum protection, and mirrored protection are OS/400 disk recovery functions that provide methods to recover recently entered data after a disk related failure. These functions use additional system resources, but provide a high level of protection for systems in a distributed relational database. Since some systems may be more critical as ASs than others, the distributed relational database administrator should review how these disk data protection methods can best be used by individual systems within the network.

Auxiliary Storage Pools

An ASP is one or more physical disk units assigned to the same storage area. ASPs allow you to isolate certain types of objects on specified physical disk units.

The system ASP isolates system programs and the temporary objects that are created as a result of processing by system programs. User ASPs can be used to isolate some objects such as libraries, SQL objects, journals, journal receivers, applications, and data. The AS/400 system supports up to 15 user ASPs. Isolating libraries or objects in a user ASP protects them from disk failures in other ASPs and reduces recovery time.

Note: SQL collections cannot be created in a user ASP, however libraries can be created in user ASPs. SQL objects can be created in a library which allows you to take advantage of a user ASP for SQL objects. However, if you create SQL objects in a library, you lose the capability of using SQL catalog views to describe the SQL objects in the library.

In addition to reduced recovery time and isolation of objects, placing objects in an ASP can improve performance. If a journal receiver is isolated in a user ASP, the disks associated with that ASP are dedicated to that receiver. In an environment that requires many read and write operations to the database files, this can reduce arm contention on the disks in that ASP, and can improve journaling performance.

Checksum Protection

Checksum protection guards against losing the data on any disk in an ASP. The checksum software maintains a coded copy of ASP data in special checksum data areas within that ASP. Any changes made to permanent objects in a checksum protected ASP are automatically maintained in the checksum data of the checksum set. If any single disk unit in a checksum set is lost, the system reconstructs the contents of the lost device using the checksum and the data on the remaining functional units of the set. In this way, if any one of the units fails, its contents may be recovered. This reconstructed data reflects the most up-to-date information that was on the disk at the time of the failure. Checksum protection can affect system performance significantly. In a distributed relational database this may be a concern.

Mirrored Protection

Mirrored protection increases the availability of a system by duplicating different disk-related hardware components such as a disk controller, a disk I/O processor, or a bus. The system can remain available after a failure, and service for the failed hardware components can be scheduled at a convenient time.

Different levels of mirrored protection provide different levels of system availability. For example, if only the disk units on a system are mirrored, all disk units have disk unit-level protection, so the system is protected against the failure of a single disk unit. In this situation, if a controller, I/O processor, or bus failure occurs, the system cannot run until the failing part is repaired or replaced. All mirrored units on the system must have identical disk unit-level protection and reside in the same ASP. The units in an ASP are automatically paired by the system when mirrored protection is started.

Journal Management

Journal management can be used as a part of the backup and recovery strategy for relational databases and indexes. AS/400 journal support provides an audit trail and forward and backward recovery. Forward recovery can be used to take an older version of a table and apply changes logged in the journal to the table. Backward recovery can be used to remove changes logged in the journal from the table.

When a collection is created, a journal and an object called a journal receiver are created in the collection. The journal and journal receiver are not created on a user ASP. However, because placing journal receivers on ASPs can improve performance, the distributed relational database administrator may wish to create all future journal receivers on an ASP.

When a table is created, it is automatically journaled to the journal SQL created in the collection. After this point, you are responsible for using the journal functions to manage the journal, journal receivers, and the journaling of tables to the journal. For example, if a table is moved into a collection, no automatic change to the journaling status occurs. If a table is restored, the normal journal rules apply. That is, if a table is journaled when it is saved, it is journaled to the same journal when it is restored on that system. If the table is not journaled at the time of the save, it is not journaled at restore time. You can stop journaling on any table using the journal functions, but doing so prevents SQL operations from running under commitment control. SQL operations can still be performed if you have specified COMMIT(*NONE), but this does not provide the same level of integrity that journaling and commitment control provide.

With journaling active, when changes are made to the database, the changes are journaled in a journal receiver before the changes are made to the database. The journal receiver always has the latest database information. All activity is journaled for a database table regardless of how the change was made.

Journal receiver entries record activity for a specific row (added, changed, or deleted), and for a table (opened, table or member saved, and so on). Each entry includes additional control information identifying the source of the activity, the user, job, program, time, and date.

The system journals some file-level changes, including moving a table and renaming a table. The system also journals member-level changes, such as initializing a physical file member, and system-level changes, such as initial program load (IPL). You can add entries to a journal receiver to identify significant events (such as the checkpoint at which information about the status of the job and the system can be journaled so that the job step can be restarted later) or to help in the recovery of applications.

For changes that affect a single row, row images are included following the control information. The image of the row after a change is made is always included. Optionally, the row image before the change is made can also be included. You control whether to journal both before and after row images or just after row images by specifying the IMAGES parameter on the Start Journaling Physical File (STRJRNPF) command.

All journaled database files are automatically synchronized with the journal when the system is started (IPL time). If the system ended abnormally, some database changes may be in the journal, but not yet reflected in the database itself. If that is the case, the system automatically updates the database from the journal to bring the tables up to date.

Journaling can make saving database tables easier and faster. For example, instead of saving entire tables everyday, you can simply save the journal receivers that contain the changes to the tables. You might still save the entire tables on a regular basis. This method can reduce the amount of time it takes to perform your daily save operations.

The Display Journal (DSPJRN) command, can be used to convert journal receiver entries to a database file. Such a file can be used for activity reports, audit trails, security, and program debugging.

Index Recovery

An index describes the order in which rows are read from a table. When indexes are recorded in the journal, the system can recover the index to avoid spending a significant amount of time rebuilding indexes during the IPL following an abnormal system end.

When you journal tables, images of changes to the rows in the table are written to the journal. These row images are used to recover the table should the system end abnormally. However, after an abnormal end, the system may find that indexes built over the table are not synchronized with the data in the table. If an access path and its data are not synchronized, the system must rebuild the index to ensure that the two are synchronized and usable.

When indexes are journaled, the system records images of the index in the journal to provide known synchronization points between the index and its data. By having that information in the journal, the system can recover both the data and the index, and ensure that the two are synchronized. In such cases, the lengthy time to rebuild the indexes can be avoided.

The AS/400 system provides several functions to assist with index recovery. All indexes on the system have a maintenance option that specifies when the index is maintained. SQL indexes are created with an attribute of *IMMED maintenance.

In the event of a power failure or abnormal system failure, indexes that are in the process of change may need to be rebuilt to make sure they agree with the data. All indexes on the system have a recovery option that specifies when the index should be rebuilt if necessary. All SQL indexes with an attribute of UNIQUE are created with a recovery attribute of *IPL, which means these indexes are rebuilt before the OS/400 licensed program has been started. All other SQL indexes are created with the *AFTIPL recovery attribute, which means they are rebuilt after the operating system has been started. During an IPL, you can see a display showing indexes needing to be rebuilt and their recovery option, and you may override these recovery options.

SQL indexes are not journaled automatically. You can use the Start Journal Access Path (STRJRNAP) command to journal any index created by SQL operations. The system save and restore functions allow you to save indexes when a table is saved by using ACCPTH(*YES) on the Save Object (SAVOBJ) or Save Library (SAVLIB) commands. If you must restore a table, there is no need to rebuild the indexes. Any indexes not previously saved and restored are automatically and asynchronously rebuilt by the database.

Before journaling indexes, you must start journaling for the tables associated with the index. In addition, you must use the same journal for the index and its associated table.

Index journaling is designed to minimize additional output operations. For example, the system writes the journal data for the changed row and the changed index in the same output operation. However, you should seriously consider isolating your journal receivers in user ASPs when you start journaling your indexes. Placing journal receivers in their own user ASP provides the best journal management performance, while helping to protect them from a disk failure.

Designing Tables to Reduce Index Rebuilding Time

Table design can also help reduce index recovery time. For example, you can divide a large master table into a history table and a transaction table. The transaction table is then used for adding new data, the history table is used for inquiry only. Each day, you can merge the transaction data into the history table, then clear the transaction file for the next day's data. With this design, the time to rebuild indexes can be shortened, because if the system abnormally ends during the day, the index to the smaller transaction table might need to be rebuilt. However, because the index to the large history table, is read-only for most of the day, it would probably not be out of synchronization with its data, and would not have to be rebuilt.

Consider the trade-off between using table design to reduce index rebuilding time and using system-supplied functions like access path journaling. The table design

described above may require a more complex application design. After evaluating your situation, you may decide to use system-supplied functions like access path journaling rather than design more complex applications.

Transaction Recovery through Commitment Control

Commitment control is an extension of the journal management function on the AS/400 system. The system can identify and process a group of relational database changes as a single unit of work (transaction).

An SQL COMMIT statement guarantees that the group of operations is completed. An SQL ROLLBACK statement guarantees that the group of operations is backed out. The only SQL statements that cannot be committed or rolled back are:

- DROP COLLECTION
- GRANT or REVOKE if an authority holder exists for the specified object

Under commitment control, tables and rows used during a transaction are locked from other jobs. This ensures that other jobs do not use the data until the transaction is complete. At the end of the transaction, the program issues an SQL COMMIT or ROLLBACK statement, freeing the rows. If the system or job ends abnormally before the commit operation is performed, all changes for that job since the last time a commit or rollback operation occurred are rolled back. Any affected rows that are still locked are then unlocked. The lock levels are as follows:

***NONE** Commitment control is not used. SQL COMMIT and ROLLBACK statements are not allowed. Uncommitted changes in other jobs can be seen.

***CHG** Objects referred to in SQL COMMENT ON, CREATE, DROP, GRANT, LABEL ON, and REVOKE statements and the rows updated, deleted, and inserted are locked until the unit of work (transaction) is completed. Uncommitted changes in other jobs can be seen.

***CS** Objects referred to in SQL COMMENT ON, CREATE, DROP, GRANT, LABEL ON, and REVOKE statements and the rows updated, deleted, and inserted are locked until the unit of work (transaction) is completed. A row that is selected, but not updated, is locked until the next row is selected. Uncommitted changes in other jobs cannot be seen.

***ALL** Objects referred to in SQL COMMENT ON, CREATE, DROP, GRANT, LABEL ON, and REVOKE statements and the rows read, updated, deleted, and inserted are locked until the end of the unit of work (transaction). Uncommitted changes in other jobs cannot be seen.

Figure 7-1 on page 7-7 shows the record lock duration for each of these lock level values.

If you request COMMIT (*CHG), COMMIT (*CS), or COMMIT (*ALL) when the program is precompiled or when interactive SQL is started, then SQL sets up the commitment control environment by implicitly calling the Start Commitment Control (STRCMTCTL) command. The LCKLVL parameter specified when SQL starts commitment control is the lock level specified on the COMMIT parameter on the CRTSQL*xxx* commands. NFYOBJ(*NONE) is specified when SQL starts commitment control. To specify a different NFYOBJ parameter, issue a STRCMTCTL command before starting SQL.

Note: When running with commitment control, the tables referred to in the application program by data manipulation language statements must be journaled. The tables do not have to be journaled at precompile time, but they must be journaled when you run the application.

If a remote relational database is accessing data on the AS/400 system and requesting commit level repeatable read (*RR), the tables will be locked until the query is closed. If the cursor is read only, the table will be locked (*SHRNUP). If the cursor is in update mode, the table will be locked (*EXCLRD).

The journal created in the SQL collection is normally the journal used for logging all changes to SQL tables. You can, however, use the system journal functions to journal SQL tables to a different journal. This is necessary if tables from multiple collections need to be used in the same unit of work. This is because AS/400 commitment control requires that all files under commitment control are journaled to the same journal.

Commitment control can handle up to 32 768 distinct row changes in a unit of work. If COMMIT(*ALL) is specified, all rows read are also included in the 32 768 limit. (If a row is changed or read more than once in a unit of work, it is only counted once toward the 32 768 limit.) Maintaining a large number of locks adversely affects system performance and does not allow concurrent users to access rows locked in the unit of work until the unit of work is completed. It is, therefore, more efficient to keep the number of rows processed in a unit of work small. Commitment control allows up to 256 tables either open under commitment control or closed with pending changes in a unit of work.

The HOLD value on COMMIT and ROLLBACK statements allows you to keep the cursor open and start another unit of work without issuing an OPEN again. The HOLD value is not available when you are connected to a remote database that is not on an AS/400 system. If ALWBLK(*ALLREAD) and COMMIT(*CHG) are specified when the program is precompiled, all read-only cursors will allow blocking of rows and a ROLLBACK HOLD statement will not roll the cursor position back.

If there are locked rows (records) pending from running a SQL precompiled program or an interactive SQL session, a COMMIT or ROLLBACK statement can be issued from the system Command Entry display. Otherwise, an implicit ROLLBACK operation occurs when the job is ended.

For more information on commitment control see the *SQL/400* Programmer's Guide*.

Figure 7-1 (Page 1 of 3). Record Lock Duration

SQL Statement	COMMIT Parameter	Duration of Record Locks	Lock Type
SELECT INTO	*NONE	No locks	
	*CHG	No locks	
	*CS	Row locked when read and released	READ
	*ALL (See note 2)	From read until ROLLBACK or COMMIT	READ
FETCH (read-only cursor)	*NONE	No locks	
	*CHG	No locks	
	*CS	From read until the next FETCH	READ
	*ALL (See note 2)	From read until ROLLBACK or COMMIT	READ

Figure 7-1 (Page 2 of 3). Record Lock Duration

SQL Statement	COMMIT Parameter	Duration of Record Locks	Lock Type
FETCH (update or delete capable cursor) See note 1	*NONE	When record not updated or deleted from read until next FETCH When record is updated or deleted from read until UPDATE or DELETE	UPDATE
	*CHG	When record not updated or deleted from read until next FETCH When record is updated or deleted from read until UPDATE or DELETE	UPDATE
	*CS	When record not updated or deleted from read until next FETCH When record is updated or deleted from read until UPDATE or DELETE	UPDATE
	*ALL	From read until ROLLBACK or COMMIT	UPDATE[3]
INSERT (target table)	*NONE	No locks	
	*CHG	From insert until ROLLBACK or COMMIT	UPDATE
	*CS	From insert until ROLLBACK or COMMIT	UPDATE
	*ALL	From insert until ROLLBACK or COMMIT	UPDATE[4]
INSERT (tables in subselect)	*NONE	No locks	
	*CHG	No locks	
	*CS	Each record locked while being read	READ
	*ALL	From read until ROLLBACK or COMMIT	READ
UPDATE (non-cursor)	*NONE	Each record locked while being updated	UPDATE
	*CHG	From read until ROLLBACK or COMMIT	UPDATE
	*CS	From read until ROLLBACK or COMMIT	UPDATE
	*ALL	From read until ROLLBACK or COMMIT	UPDATE
DELETE (non-cursor)	*NONE	Each record locked while being deleted	UPDATE
	*CHG	From read until ROLLBACK or COMMIT	UPDATE
	*CS	From read until ROLLBACK or COMMIT	UPDATE
	*ALL	From read until ROLLBACK or COMMIT	UPDATE
UPDATE (with cursor)	*NONE	Lock released when record updated	UPDATE
	*CHG	From read until ROLLBACK or COMMIT	UPDATE
	*CS	From read until ROLLBACK or COMMIT	UPDATE
	*ALL	From read until ROLLBACK or COMMIT	UPDATE
DELETE (with cursor)	*NONE	Lock released when record deleted	UPDATE
	*CHG	From read until ROLLBACK or COMMIT	UPDATE
	*CS	From read until ROLLBACK or COMMIT	UPDATE
	*ALL	From read until ROLLBACK or COMMIT	UPDATE
Subqueries (update or delete capable cursor or UPDATE or DELETE non-cursor)	*NONE	From read until next FETCH	READ
	*CHG	From read until next FETCH	READ
	*CS	From read until next FETCH	READ
	*ALL (see note 2)	From read until ROLLBACK or COMMIT	READ

Figure 7-1 (Page 3 of 3). Record Lock Duration

SQL Statement	COMMIT Parameter	Duration of Record Locks	Lock Type
Subqueries (read-only cursor or SELECT INTO)	*NONE	No locks	
	*CHG	No locks	
	*CS	Each record locked while being read	READ
	*ALL	From read until ROLLBACK or COMMIT	READ

Notes:

1. A cursor is open with UPDATE or DELETE capabilities if the result table is not read-only (see description of DECLARE CURSOR in *SQL/400* Reference*) and if one of the following is true:
   - The cursor is defined with a FOR UPDATE clause.
   - The cursor is defined without a FOR UPDATE, FOR FETCH ONLY, or ORDER BY clause and the program contains at least one of the following:
     - Cursor UPDATE referring to the same cursor-name
     - Cursor DELETE referring to the same cursor-name
     - An EXECUTE or EXECUTE IMMEDIATE statement with ALWBLK(*READ) or ALWBLK(*NONE) specified on the CRTSQL*xxx* command
2. A table or view can be locked exclusively in order to satisfy COMMIT(*ALL). If a subselect is processed that includes a group by or union, or if the processing of the query requires the use of a temporary result, an exclusive lock is acquired to protect you from seeing uncommitted changes.
3. If the row is not updated or deleted, the lock is reduced to *READ.
4. An UPDATE lock on rows of the target table and a READ lock on the rows of the subselect table.
5. A table or view can be locked exclusively in order to satisfy repeatable read. Row locking is still done under repeatable read. The locks acquired and their duration are identical to *ALL.

Writing Data to Auxiliary Storage

The Force-Write Ratio (FRCRATIO) parameter on the Create File command can be used to force data to be written to auxiliary storage. A force-write ratio of one causes every add, update, and delete request to be written to auxiliary storage immediately for the table in question. However, choosing this option can reduce system performance. Therefore, saving your tables and journaling tables should be considered the primary methods for protecting the database.

Save and Restore Processing

Saving and restoring data and programs allows recovery from a program or system failure, exchange of information between systems, or storage of objects or data off-line. A sound backup policy at each system in the distributed relational database network ensures a system can be restored and made available to network users quickly in the event of a problem.

Saving the system on external media such as tape, protects system programs and data from disasters, such as fire or flood. However, information can also be saved to a disk file called a save file. A save file is a disk-resident file used to store data until it is used in input and output operations or for transmission to another AS/400 system over communication lines. Using a save file allows unattended save operations because an operator does not need to load diskettes or tapes. In a distributed relational database, save files can be sent to another system as a protection method.

When information is restored, the information is written from diskette, tape, or a save file into auxiliary storage, where it can be accessed by system users.

The AS/400 system has a full set of commands to save and restore your database tables and SQL objects:

- Save Library (SAVLIB) saves one or more collections
- Save Object (SAVOBJ) saves one or more objects such as SQL tables, views and indexes
- Save Changed Object (SAVCHGOBJ) saves any objects that have changed since either the last time the collection was saved or from a specified date
- Save Save File Data (SAVSAVFDTA) saves the contents of a save file
- Save System (SAVSYS) saves the operating system, security information, device configurations, and system values
- Restore Library (RSTLIB) restores a collection
- Restore Object (RSTOBJ) restores one or more objects such as SQL tables, views and indexes
- Restore User Profiles (RSTUSRPRF), Restore Authority (RSTAUT) and Restore Configuration (RSTCFG) restore user profiles, authorities, and configurations saved by a SAVSYS command

See the *Advanced Backup and Recovery Guide* for more information about these functions and commands.

Saving and Restoring Indexes

Restoring an SQL index can be faster than rebuilding it. Although times vary depending on a number of factors, rebuilding a database index takes approximately 1 minute for every 10,000 rows.

After restoring the index, the table may need to be brought up to date by applying the latest journal changes (depending on whether journaling is active). Normally, the system can apply approximately 80,000 to 100,000 journal entries per hour. (This assumes that each of the tables to which entries are being applied has only one index or view built over it.) Even with this additional recovery time, you will usually find it is faster to restore indexes rather than to rebuild them.

The system ensures the integrity of an index before you can use it. If the system determines that the index is unusable, the system attempts to recover it. You can control when an index will be recovered. If the system ends abnormally, during the next IPL the system automatically lists those tables requiring index or view recovery. You can decide whether to rebuild the index or to attempt to recover it at one of the following times:

- During the IPL
- After the IPL
- When the table is first used

For more information, see the *Advanced Backup and Recovery Guide* topics about saving and restoring access paths.

Saving and Restoring Security Information

If you make frequent changes to your system security environment by updating user profiles and updating authorities for users in the distributed relational database network, you can save security information to media or a save file without a complete Save System (SAVSYS) command, a long-running process that uses a dedicated system. With the Save Security Data (SAVSECDTA) command you can save security data in a shorter time without using a dedicated system. Data saved using the SAVSECDTA command can be restored using the Restore User Profile (RSTUSRPRF) or Restore Authority (RSTAUT) commands.

Saving and Restoring SQL Packages

When an application program that refers to a relational database on a remote system is precompiled and bound, an SQL package is created on the AS to contain the control structures necessary to process any SQL statements in the application.

An SQL package is an AS/400 object, so it can be saved to media or a save file using the Save Object (SAVOBJ) command and restored using the Restore Object (RSTOBJ) command.

An SQL package must be restored to a collection having the same name as the collection from which it was saved, and it cannot be renamed.

Saving and Restoring Relational Database Directories

The relational database directory is not an AS/400 object. The relational database directory is made up of files that are opened by the system at IPL time, so the SAVOBJ command cannot used to directly save these files. You can save the relational database directory by creating an output file from the relational database directory data. This output file can then be used to add entries to the directory again if it is damaged.

When entries have been added and you want to save the relational database directory, specify the OUTFILE parameter on the Display Relational Database Directory Entry (DSPRDBDIRE) command to send the results of the command to an output file. The output file can be saved to tape, diskette, or a save file and restored to the system. If your relational database directory is damaged or your system needs to be recovered, you can restore the output file containing relational database entry data using a control language (CL) program. The CL program reads data from the restored output file and creates the CL commands that add entries to a new relational database directory.

For example, the relational database directory for the Spiffy Corporation MP000 system is sent to an output file named RDBDIRM as follows:

```
DSPRDBDIRE OUTPUT(*OUTFILE) OUTFILE(RDBDIRM)
```

The sample CL program that follows reads the contents of the output file RDBDIRM and creates a relational database directory entry using the Add Relational Database Directory Entry (ADDRDBDIRE) command. In this example, the old directory is cleared before the new entries are made.

```
/*       *** Program to Restore RDB Entries***                          */
/*                                                                      */
PGM
          DCLF          FILE(RDBDIRM)
          RMVRDBDIRE    RDB(*ALL)
LOOP:     RCVF
          MONMSG        MSGID(CPF0864) EXEC(GOTO CMDLBL(ENDLOOP))
          IF (&RWRLOC = '*LOCAL') DO
             ADDRDBDIRE RDB(&RWRDB) RMTLOCNAME(&RWRLOC) TEXT(&RWTEXT)
          ENDDO
          ELSE DO
             ADDRDBDIRE    RDB(&RWRDB) RMTLOCNAME(&RWRLOC) +
                           TEXT(&RWTEXT) DEV(&RWDEV) +
                           LCLLOCNAME(&RWLLOC) RMTNETID(&RWNTID) +
                           MODE(&RWMODE) TNSPGM(&RWTPN)
          ENDDO
/*     ***** Add the entry to the RDB Directory *****                   */
GOTO      LOOP
ENDLOOP:
ENDPGM
```

The files that make up the relational database directory are saved when a SAVSYS command is run. The physical file that contains the relational database directory can be restored from the save media to your library with the following Restore Object (RSTOBJ) command:

```
RSTOBJ      OBJ(QADBXRDBD)  SAVLIB(QSYS)
            DEV(TAP01)  OBJTYPE(*FILE)
            LABEL(Q5738SS1vrmxx003)
            RSTLIB(your lib)
```

In this example, the relational database directory is restored from tape. The *vrm* in the LABEL parameter is the version, release, and modification level of the OS/400 operating system. The *xx* in the LABEL parameter is the last two digits of the current system language value. For example, 2924 is for the English language; therefore, the value of *xx* is 24.

After you restore this file to your library, you can use the information in the file to re-create the relational database directory.

Ensuring Data Availability

The more critical certain data is to your enterprise, the more ways you should have for accessing that data. This means you might also consider aspects of network redundancy as well as data redundancy when planning your strategy to ensure the optimum availability of data across your network.

Network Redundancy Issues

Network redundancy provides different ways for users on the distributed relational database network to access a relational database on the network. If there is only one communications path from an AR to an AS, when the communications line is down, users on the AR do not have access to the AS relational database. For this reason network redundancy issues are important to the distributed relational database administrator for the Spiffy Corporation.

For example, consider service booking or customer parts purchasing issues for a dealership. When a customer is waiting for service or to purchase a part, the service clerk needs access to all authorized tables of enterprise information to schedule work or sell parts.

If the local system is down, no work can be done. If the local system is running but a request to a remote system is needed to process work and the remote system is down, the request can not be handled. In the Spiffy Corporation example, this might mean a dealership cannot request parts information from a regional inventory center. Also, if an AS that handles many AR jobs is down, none of the ARs can complete their requests. In the case of the Spiffy Corporation network, if a regional center is down, none of the ASs it supports can order parts.

Providing the region's dealerships with access to regional inventory data is important to the Spiffy Corporation distributed relational database administrator. Providing paths through the network to data can be addressed several ways. The original network configuration for the Spiffy Corporation linked the end node dealerships to their respective network node regional centers.

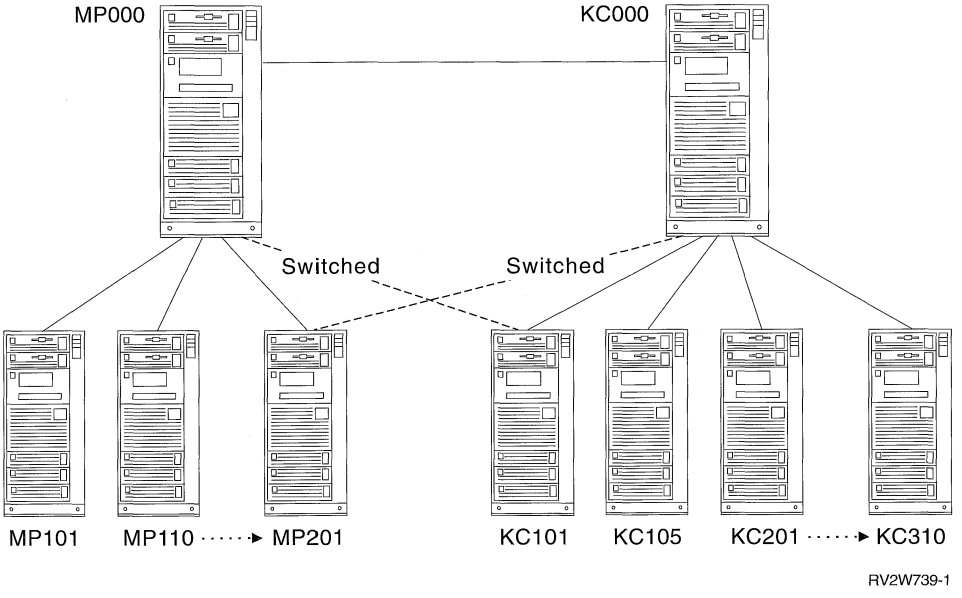


Figure 7-2. Alternative Network Paths

An alternative for some dealerships might be a switched-line connection to a different regional center. If the local regional center is unavailable to the network, access to another AS allows the requesting dealership to obtain information needed to do their work. In Figure 7-2, some ARs served by the MP000 system establish links to the KC000 system, which can be used whenever the MP000 system is unavailable. The Vary Configuration (VRYCFG) or Work With Configuration Status (WRKCFGSTS) commands can be used by a system operator or distributed relational database administrator to vary the line on when needed and vary the line off when the primary AS is available.

Another alternative could be if one of the larger area dealerships also acted as an AS for other dealerships. As shown in Figure 7-3 on page 7-14, an end node is only an AS to other end nodes through its network node. In Figure 7-2, if the link to Minneapolis is down, none of the dealerships could query another (end node) for inventory. The configuration illustrated above could be changed so that one of the

dealerships is configured as an APPN network node, and lines to that dealership from other area dealerships are set up.

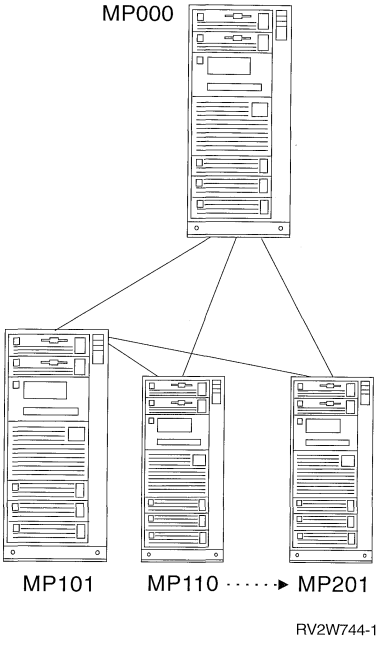


Figure 7-3. Alternate Application Server

Data Redundancy in Your Network

Data redundancy in a distributed relational database also provides different ways for users on the distributed relational database network to access a database on the network. The issues a distributed relational database administrator examines to create a data redundancy strategy are more complex than ensuring communications paths are available to the data. Tables can be replicated across systems in the network, or a snapshot of data can be used to provide data availability, but in each case user applications must be developed to provide the capability.

The figure below shows that a copy of the MP000 system distributed relational database can be stored on the KC000 system, and a copy of the KC000 system distributed relational database can be stored on the MP000 system. The ARs from one region can link to the other AS to query or to update a replicated copy of their relational database.

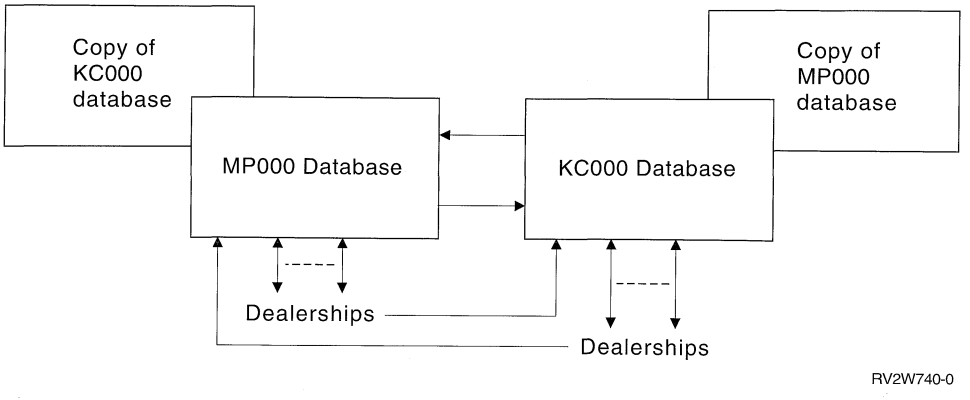


Figure 7-4. Data Redundancy Example

The administrator must decide what is the most efficient, effective strategy to allow distributed relational database processing. Alternative strategies might include these scenarios.

One alternative may be that when MP000 is unavailable, its ARs connect to the KC000 system to query a read-only snapshot of the MP000 distributed relational database so service work can be scheduled.

The snapshot application, designed by the user, provides a read-only copy of the tables to a remote system on a regular basis. For the Spiffy Corporation, this might be at the end or the beginning of each business day. In this example, the MP000 database snapshot provides a 24-hour-old, last-point-in-time picture for dealerships to use for scheduling only. When the MP000 system is back on line, its ARs query the MP000 distributed relational database to completely process inventory requests or other work queried on the snapshot.

Another alternative may be that Spiffy Corporation wants dealership users to be able to update a replicated table at another AS when their regional AS is unavailable.

For example, an AR that normally connects to the MP000 database could connect to a replicated MP000 database on the KC000 system to process work. When the MP000 system is available again, the MP000 relational database can be updated by applying journal entries from activity originating in its replicated tables at the KCOOO location. When these journal entries have been applied to the original MP000 tables, distributed relational database users can access the MP000 as an AS again.

Journal management processes on each regional system update all relational databases. The amount of journal management copy activity in this situation should be examined because of potential adverse performance effects at these systems. Also, setup of the replicated tables, sending and receiving journal receivers between the local and remote system, and applying journal changes to the local database require sophisticated user-written applications.

Chapter 8. Distributed Relational Database Performance

The performance of your distributed relational database depends on the design of your network, the system, and your database.

Improving Performance Through the Network

The performance of your network depends on the following:

- Line speed
- Connection type (nonswitched versus switched)
- Frame size
- RU sizing
- Pacing

Line Speed

An application program cannot move data faster than the line or network can carry it. When analyzing the effect the line speed is having on the performance of your network, consider the following:

- Amount of information being moved:
  - The size of the file for a batch file transfer
  - The typical transaction for an interactive application program and the amount of data that is sent and received for each transaction
  - The number of application programs or users that will be using the line at the same time
- Environment (whether it is batch or interactive):
  - For interactive application programs, make sure that the average line use does not exceed 50%. Batch application programs can usually approach 100% line use with no adverse performance effects.
  - Mixing interactive and batch application programs on the same line is not recommended. If you do choose to combine them, careful planning is required to ensure that both interactive and batch transactions are given sufficient opportunity to use the line.
- Bandwidth:
  - For line speeds of 19 200 bits per second (bps) or less, most application programs can fully use the bandwidth (data capacity) of the line. The performance in this environment depends largely on the ability of the line to transfer the data.
  - For line speeds greater than 19 200 bps, some application programs cannot fully use the bandwidth of the line. The performance in this environment depends more on the performance of the application program than on the ability of the line to transfer data. For example, a batch program that involves multiple actions on data before requesting the output operation may not deliver data to the line fast enough to take full advantage of the line speed. Multiple jobs running at the same time can take advantage of a greater line speed.

- Local area networks are intended to be shared by many stations on the same network. A token-ring network contains 4 to 6 million bps of bandwidth and Ethernet contains 10 million bps of bandwidth using Ethernet, so a single hardware adapter cannot use the total network bandwidth of the network. In all cases, the performance in a network environment depends on the performance of the application programs. The hardware adapter can be the factor limiting performance in cases where multiple application programs are using the same adapter.

  Another consideration is that a local area network that handles large amounts of data from a large number of stations can affect the performance of the entire network to which the local area network is attached.

Connection Type (Nonswitched Versus Switched)

The communications lines must be either nonswitched (the line is always available) or switched (the connection is established by dialing, and only made available when needed). A switched line should probably be used for short and infrequent transmission of batch files. When evaluating performance based on connection type, consider the following:

- Cost of the line (switched lines are usually less expensive)
- Distance between stations
- Frequency of use
- Duration of use
- Time to establish a connection
- Use of the line (batch or interactive transfer)
- Switched lines can only be point-to-point (only two devices are connected on a line)

The AS/400 system supports telephone number lists for V.25, V.25bis, X.21, and X.25 switched lines. You can use the Create Controller (CRTCTLxxx) or Change Controller (CHGCTLxxx) commands to specify up to 256 telephone numbers. The system will try to make a switched connection, trying one telephone number after another until a connection is made or the list is completed.

Frame Size

The performance of your network also depends on the frame size. Usually the larger the frame size used, the better the performance. The following compares frame sizes for various networks.

For integrated services digital network (ISDN) data link control (IDLC), the AS/400 system supports frame sizes of 256 to 8196 bytes. The default frame size is 2048 bytes. Although the best data throughput is usually achieved with the largest frame size possible, large frame sizes may not work well if one or more of the following occurs:

- The data consists of many small blocks
- Some condition causes frequent sending again of frames
- Very large frames cause data storage problems

For a synchronous data link controller (SDLC), the AS/400 system supports frame sizes up to 2057 bytes. For X.25, the AS/400system supports up to 1024 bytes.

To achieve most efficient system performance, try to make the maximum Systems Network Architecture (SNA) request unit or response unit (RU) size as close to the

frame size as possible without exceeding the frame size. If you do choose to make the RU size larger than the frame size, try to use an RU size that is slightly less than a multiple of the frame size. Avoid the situation where the RU size divided by the frame size results in an integer and a small remainder. This small remainder requires an additional small frame to be sent for each RU, which results in additional overhead.

Larger frame sizes may not work in an environment where the line has a high probability for errors. If the frames need to be transmitted again, larger frames take longer.

For a token-ring network, the AS/400 system supports frame sizes up to 8156 bytes or up to 16393 bytes on a 16 million bps line. (The frame size depends on the installed token-ring adapter.) A token-ring network operating on a 4 million bit-per-second line supports a range of frame sizes up to 1994 bytes or up to 4060 bytes, depending on the token-ring adapter. The frame size is specified with the MAXFRAME parameter in the line and controller descriptions.

For Ethernet, the AS/400 system supports frame sizes up to 1496 bytes on a 10 million bps line. (The frame size depends on the protocol selection.) The frame size is based on the value in the Ethernet standard field of the line and controller descriptions. If the value is Ethernet version II (*ETHV2), the frame size can be up to 1493 bytes. If the value is *IEEE8023, the frame size can be up to 1496 bytes.

The effective MAXFRAME value for the controller description is not necessarily the same as the MAXFRAME value on the controller description. The effective MAXFRAME is the minimum of the MAXFRAME on the controller description, the MAXFRAME on the line description, and the source service access point (*SSAP) maximum frame size on the line description.

The larger frame sizes generally supply better performance. The improved performance caused by larger frame sizes is noticeable with Ethernet as with a token-ring network because of the faster media speed. However, if the network is subject to errors, the maximum allowable frame size may not be the most efficient because of the time required to send large frames again.

RU Sizing

In SNA, the **request unit** is the record transmitted to the other system. The **response unit** is the record sent to respond to a request.

The maximum SNA RU size should be as large as the data link protocol allows. The system automatically calculates the RU size based on the maximum frame size, unless you specify a maximum RU size when you create the mode description.

For SDLC, Ethernet, or a token-ring network, try to make the maximum RU size (the MAXLENRU parameter for token-ring) as close to the frame size as possible without exceeding the frame size. If you do choose to make the RU size larger than the frame size, try to use an RU size that is slightly less than a multiple of the frame size. Avoid the situation where the RU size divided by the frame size results in an integer and a small remainder. This small remainder requires an additional small frame to be sent for each RU, which results in additional overhead.

For an X.25 network, try to make the maximum RU size (the MAXLENRU parameter) as close to the packet size as possible without being equal to or exceeding the packet size. If you do choose to make the RU size larger than the packet size, try to use an RU size that is slightly less than a multiple of the packet size. Avoid the situation where the RU size divided by the packet size results in an integer and a small remainder. This small remainder requires an additional packet to be sent, which results in additional overhead.

Pacing

Pacing is a technique by which the receiving system controls the rate of transmission of the sending system to prevent overrun of the receiving storage. **Overrun** is the loss of data because a receiving device is unable to accept data at the rate it is transmitted.

Pacing is required if there is a possibility of overflowing data buffers in the controller or the host system. This usually occurs if the controller or host must pass the data to a device operating at a slow speed.

Pacing applies only to application programs using SNA support. Pacing specifies the number of request units (RUs) that can be sent before a response or acknowledgement is required from the receiving device. A large pacing size generally provides better performance for a single application program on a line. Sometimes the pacing size must be balanced between interactive and batch application programs on the same line or network. If the pacing size is too large for batch application programs, it can cause the line to be busy for long periods of time, and the interactive application programs experience long and inconsistent response times.

If communicating to a remote system using advanced program-to-program communications (APPC) or advanced peer-to-peer networking (APPN), the pacing values used are negotiated between the two systems. APPC support on the AS/400 system regulates pacing dynamically and does this balancing for you if the attached remote system also regulates pacing dynamically.

Improving Performance Through the System

Achieving efficient system performance requires a proper balance among system resources. Overusing any resource adversely affects performance.

Understanding the System Components that Affect Performance

The following system components affect system performance:

- Pools
- Activity level
- Job states
- Ineligible queue
- System objects
- PURGE parameter
- Time slice

This section discusses the ways these components affect system performance.

Pools

On the AS/400 system, all main storage can be divided into logical allocations called **storage pools**. Two basic types of storage pools exist on the system: shared and private.

A **shared storage pool** is a pool in which multiple subsystems can run jobs. Using shared storage pools allows the system to distribute the storage requirements of interactive users across multiple subsystems, still allowing their jobs to run in the same storage pool.

There are 14 shared storage pools defined on the system, 13 of which you can specify for use when creating subsystem descriptions (the machine pool is reserved for system use). Shared storage pools include the following:

- *MACHINE is the machine storage pool that is used for highly shared machine and OS/400 licensed programs. The size for this storage pool is specified in the system value QMCHPOOL. No user jobs run in this storage pool. (On the Work with System Status (WRKSYSSTS) display, the machine storage pool appears as system pool identifier 1.)
- *BASE is the base storage pool that contains all unassigned main storage on the system; that is, all storage that is not required by the machine storage pool or by another pool. The system value QBASPOOL specifies the minimum size of the base storage pool. The activity level for this storage pool is specified in the system value QBASACTLVL. The base storage pool is used for batch work and miscellaneous system functions. (On the WRKSYSSTS display, the base storage pool appears as system pool identifier 2.)
- *INTERACT is the interactive storage pool used for interactive jobs.
- *SPOOL is the storage pool used for spool writers.
- *SHRPOOL1 through *SHRPOOL10 are storage pools that you can use for your own use.

A **private storage pool** is a pool in which a single subsystem can run jobs. A private pool does not have to be large enough to contain your programs. The machine manages the transfer of data (and programs) into the private storage pool if necessary. If data is already in main storage, it can be referred to independently of the storage pool it is in. However, if needed data does not exist in any storage pool, it is brought into the same storage pool for the job that referred to it (this is known as a page fault). As data is transferred into a storage pool, other data is displaced and, if changed, is automatically recorded in auxiliary storage (this is called paging). The storage pool size should be large enough to keep data transfers (paging) at a reasonable level as the rate affects performance.

When you create or change a subsystem description, you can define one or private more pools, which are labeled 1, 2, 3, and so on. These are the designations of the subsystem pools, and do not correspond to the pool numbers shown on the system status display. The Work with Subsystems (WRKSBS) display relates the subsystem pool identifiers and the column headings to the system pool identifiers.

A system job, called a subsystem monitor job, controls all the activity in a subsystem. Storage for the subsystem monitor comes from pool 1 of the subsystem, but does not count toward an activity level.

On the AS/400 system, the shipped default causes the storage for the subsystem monitor to come from the base storage pool. If you want storage for the subsystem monitor to come from a separate pool, use the Create Subsystem Description (CRTSBSD) or Change Subsystem Description (CHGSBSD) command with a specific pool size and activity level such as:

```
POOLS((1 300 2))
```

In addition to the machine storage pool and the base storage pool, you can create user-defined storage pools. User-defined storage pools can be used by IBM-supplied subsystems or user-defined subsystems. To create a user-defined storage pool, use the POOLS parameter on the Create Subsystem Description (CRTSBSD) or Change Subsystem Description (CHGSBSD) commands.

Activity Level

The **activity level** of a storage pool is the number of jobs that can run at the same time in a storage pool. The machine controls this level. Often during processing in a job, a program waits for a system resource or a response from a work station user. During such waits, a job gives up its use of the storage pool in order that another job that is ready to be processed can take its place.

More than one job can be active at the same time in a storage pool because the processing for a job can be briefly interrupted while needed data is accessed from auxiliary storage. During this delay, which is usually short, another job can run. Using the activity level, the machine can process a large number of jobs in a storage pool and, at the same time, hold the level of contention to the limit you specify.

Once the maximum activity level for a storage pool has been reached, additional jobs needing the storage pool are automatically put in a queue to wait for available main storage. Jobs in this queue are placed in an ineligible state and are shown on the Work with Active Jobs display. As soon as a job gives up its use of the storage pool, the jobs in the queue become eligible to run by their priority. For example, if a running job is waiting for a response from a work station, it gives up its activity level and the activity level is no longer at its maximum.

Defining storage pools and activity levels correctly is generally dependent on the number of jobs a subsystem must support at the same time and the characteristics and use of the programs that perform functions in those jobs.

The storage pool that user jobs get their storage from is always the same pool that limits their activity level. System jobs (such as SCPF, QSYSARB, and QLUS) get their storage from the base pool, but use the machine pool activity level. Subsystem monitors get their storage from the first subsystem description pool but not the activity level. This allows a subsystem monitor to run regardless of the activity level setting.

Job States

Jobs running on the system can be in any of the following states:

- *Active*: The job exists in main storage and processes work requested by the application.
- *Wait*: The job needs a resource that is not available. When a job is waiting for a resource, it may wait in main storage or it may be removed from main

storage until the resource is available. There are two types of waits: short waits and long waits.

A **short wait** is a period of time no longer than two seconds. It occurs in main storage and causes one of the available activity level positions to be unavailable until the requested resource is available or two seconds have passed. Some typical causes of short waits are:

- Sending a WRITE instruction to a display when *NO is specified on the Defer Write (DFRWRT) parameter
- Sending break messages to work stations
- Specifying *YES on the Restore Display (RSTDSP) parameter on display files

When using remote lines, avoid short waits because they cause the wait time in main storage to be much longer for a job than if the job was waiting for resources at a local work station.

A **long wait** is a period of time longer than two seconds. When a long wait occurs, job's position in main storage becomes available for another job. Other examples of long waits are:

- Record lock conflicts
- Distributed Data Management data requests
- Distributed relational database requests

A specialized form of long wait, called **key/think wait**, occurs outside the activity level when a job completes a work assignment and returns to request more work. This is a user-specified time period giving the user time to decide what data should be entered and to type this data. When the job receives a new assignment or runs out of time (times out), it attempts to run again. If no activity level space is available, the job becomes ineligible.

- *Ineligible*: The job has work to do, but the system is unable to accept more work at that time.

Ineligible Queue

For interactive environments, typically there are more jobs running than there is space for them to run. When a job attempts to run, there must be space for the job in main storage. To restrict the number of jobs in main storage at one time, the activity level is specified for each pool in the system. Before a job can become active, an activity level must be available.

If an activity level is available, the job becomes active and begins processing in main storage. If no activity level is available, the job becomes ineligible. When a job becomes ineligible, it is placed in the ineligible queue until an activity level is available.

If the job enters the long wait state by other than a lock conflict, it is placed behind all other jobs of equal priority already on the ineligible queue. This is called a **first-in, first-out priority queue**.

However, if a job becomes ineligible after a long wait caused by a lock conflict, it is placed in front of jobs of equal priority already on the ineligible queue. The most common reasons for this change to normal queue placement are:

- The job entered a long wait as a result of a lock conflict because it was active (referring to objects in main storage) before the conflict occurred. If the wait was short (and many are), you may be able to get the job back into an activity level before all of the objects the job was using are removed from main storage.
- When the job has been granted the lock, it leaves the wait state. If other jobs on the ineligible queue are to use the same object, they must wait until the object is once again available. Therefore, you want jobs holding locks on objects to use them and make them available for other jobs to use. To accomplish this, the job moves ahead of any potential requesters.

By correctly managing the ineligible queue, the system may avoid unnecessary job transitions and disk operations. As a result, throughput and response time are more consistent.

System Objects

Each job running in the system is assigned to a storage pool. When a job is active, it resides in its assigned storage pool. Active jobs refer to many different system objects. For jobs to use these objects, they must be in main storage. If they are not in main storage, they must be read into main storage from their locations on disk (auxiliary) storage. Some of the objects used by jobs are:

- Data areas
- File override information
- Device files
- Application codes
- System codes
- Open file information
- Queues
- Tables
- Views
- Subfile work areas
- Program variables

A **process access group** (PAG) is a group of job-related objects that can be paged in and out of storage in a single operation when a job (process) enters or leaves a long wait. Although many different object types are found in the PAG, they fall into two main categories: objects that are shared by jobs and objects that are unique to a specific job. When an object is shared, only one copy of the object exists. For example, application code used by 20 jobs resides in main storage in only one place but is used by all the jobs. However, the variables and data in the application do not have the same values for all jobs using the application. These portions of the application and other unique objects are packaged as an object called a process access group (PAG).

When a job enters an activity level, the PAG is automatically transferred from auxiliary storage to main storage. After the PAG has been written to auxiliary storage, the main storage space is available for other jobs. Whenever a job is active, the pages of the PAG that are actually used must be in main storage.

PURGE Parameter

The PURGE parameter indicates if the working storage for a routing step is to be removed from main storage and placed in auxiliary storage at the end of a time slice or when entering a long wait. PURGE is a work management tuning parameter. To retrieve a job's PAG into main storage, the system refers to the value specified on the PURGE parameter in the job class, which is resolved when the job first enters the system. The value for the PURGE parameter is either *YES or *NO.

- If *YES is specified for the PURGE parameter, the system has the following characteristics:
  - Adapts to work load and storage size and operates as *YES in limited storage
  - Adapts to work load and storage size and operates as *NO in adequate storage
  - Performs better with small main storage
- If *NO is specified for the PURGE parameter, the system has the following characteristics:
  - May use fewer nondatabase READ operations per transaction
  - May cause fewer WRITE operations per transaction
  - May transfer fewer pages
  - May reduce disk use
  - May use less processing unit time per transaction
  - Requires more main storage

  You can multiply the number of jobs running in the pool by 250KB to determine the size a pool should be if PURGE(*NO) is specified. If the result of your multiplication is approximately equal to your pool size, set the PURGE attribute to *NO. To calculate the initial activity level of the pool, refer to the *Work Management Guide*.

Time Slice

Time slice is another of the work management tuning parameters. The time slice value is specified in the job's class. The value represents the amount of processing unit time a job is allowed to use while processing a transaction. It does not represent the elapsed time of a transaction.

If a job fails to complete a transaction in the specified time slice, one of the following occurs:

- If no jobs of equal or higher priority are on the ineligible queue, the job is given another time slice, remains in main storage, and continues the transaction.
- If jobs of equal or higher priority are on the ineligible queue, the job currently running is removed from main storage and placed on the ineligible queue. A job from the ineligible queue is moved into main storage and processing continues.
- If the job is interactive and a time slice end pool is specified for the job, the job is moved to the time slice end pool.

Reducing the Affect of Long-Running Interactive Transactions

As explained in "Time Slice" on page 8-9, a job is allocated a time slice (an amount of processing time) when it begins to process a transaction. The time slice is used to prevent processing unit-intensive transactions from using all the resources of the pool in which the transaction is running. When a job fails to complete a transaction within its assigned time slice, one of the following may occur:

- The job moves to another pool.
- The transaction run is temporarily suspended by the system.

If you specify *BASE in the system value QTSEPOOL, the system attempts to reduce the affect of a long-running transaction on other users in the pool. To do this, the system runs the remainder of the transaction in the *BASE pool rather than allowing it to complete in the interactive pool.

At the completion of the long-running transaction, the system returns the job to the interactive pool. This action assumes that batch is running in *BASE and that the time slice for the interactive jobs is exceeded by only a small percentage of transactions. Usually the transactions that exceed the default time slice value of two seconds for an interactive job are transactions that are characteristic of batch-type activity.

In general, these actions have a positive affect on system performance and *BASE should be specified for this system value. However, the following situations can cause a negative affect on system performance:

- *BASE is very small. If the pool is too small, there is not enough storage to contain the work being moved to the pool. When this occurs, the system begins to perform a large number of disk operations. As a result, jobs are unable to perform productive disk requests and system performance is poor. If this situation occurs in your environment, add storage to the *BASE pool.
- The activity level in *BASE is not set properly. If the activity level is too large, either add storage to the *BASE pool or reduce the activity level. If the activity level is too small, increase the activity level and increase the main storage in *BASE. Jobs running in *BASE should have approximately 500KB (KB equals 1024 bytes) per activity level to perform efficiently.

By properly sizing *BASE and choosing an appropriate activity level, moving long-running transactions from the interactive pool to *BASE should provide better system performance. If system performance is not better after several tries, set QTSEPOOL to *NONE and reset the system pool sizes and activity levels to their original values.

Observing System Performance

The following system commands are available to help you observe the performance of your system:

- Work with System Status (WRKSYSSTS)
- Work with Disk Status (WRKDSKSTS)
- Work with Active Jobs (WRKACTJOB)

Also, you can use the AS/400 Performance Tools licensed program to help analyze your performance.

This section discusses only the system commands to gather meaningful statistics, you should observe system performance during typical levels of activity. For example, statistics gathered while no jobs are running on the system are of little value in assessing system performance. To observe the system performance, complete the following steps:

1. Enter the WRKSYSSTS, WRKDSKSTS, or WRKACTJOB command.
2. Allow the system to collect data for a minimum of 5 minutes.
3. Press F5 (Refresh) to refresh the display and present the performance data.
4. Tune your system based on the new data.

Press F10 (Restart) to restart the elapsed time counter.

Working with System Status

The Work with System Status display shows the current status of the system. When tuning the system, make sure that the machine pool is treated separately from the other pools. Type the WRKSYSSTS command on the command line and press the Enter key. The Work with System Status display appears.

```
                          Work with System Status                 MP000
                                                         03/29/92  09:52:11
% CPU used . . . . . :       35.6    Auxiliary storage:
Elapsed time . . . . . :   00:01:32      System . . . . . :     1803 M
Jobs in system . . . . . :       96      % used . . . . . :    64.3508
% addresses used:                        Total  . . . . . :     1803 M
  Permanent  . . . . . :     2.805      Current unprotect used :    285 M
  Temporary  . . . . . :     5.906      Maximum unprotect  . . :    307 M

Type changes (if allowed), press Enter.

System    Pool    Reserved   Max    -----DB-----  ---Non-DB---
 Pool   Size (K)  Size (K)  Active  Fault  Pages  Fault  Pages
   1       9639      4517    +++       .0     .0     .3     .4
   2       4000         0      6       .0    5.4    1.8    4.0
   3      35163         0     58       .0     .2    1.1    6.5
   4        350         0      5       .0     .0     .0     .0

                                                                  Bottom
Command
===>
F3=Exit   F4=Prompt           F5=Refresh   F9=Retrieve      F10=Restart
F11=Display transition data   F12=Cancel   F24=More keys
```

Figure 8-1 shows the nondatabase fault rate. Since the machine pool contains objects used system-wide, page faulting in the machine pool affects all jobs on the system. Therefore, it is desirable to maintain a low page fault rate in this pool. The guidelines you should apply are listed in Figure 8-1.

Figure 8-1. Nondatabase Paging Faults

Main Storage Size	Good	Acceptable	Poor
Less than or equal to 12MB	<2	2 – 5	>5
More than 12MB	<1	1 – 3	>3

The only way to affect paging in the machine pool is to adjust the size of the pool (values are represented in the second column of the Work with System Status display).

If you want information about the transition data, press F11 to view the Work with System Status display.

```
                          Work with System Status                    MP000
                                                              03/29/92  09:52:11
% CPU used . . . . . . . :       35.6    Auxiliary storage:
Elapsed time . . . . . . :   00:01:32      System . . . . . . . . . :    1803 M
Jobs in system . . . . . :        96       % used . . . . . . . . . :   64.3508
% addresses used:                          Total  . . . . . . . . . :    1803 M
  Permanent  . . . . . . :      2.805      Current unprotect used :     285 M
  Temporary  . . . . . . :      5.906      Maximum unprotect  . . :     307 M

Type changes (if allowed), press Enter.

System    Pool    Reserved   Max    Active->  Wait->  Active->
 Pool   Size (K)  Size (K)  Active    Wait     Inel     Inel
   1       9639      4517    +++       2.6       .0       .0
   2       4000         0      6      77.5       .0       .0
   3      35163         0     58      18.2       .0       .0
   4        350         0      5        .0       .0       .0

                                                                      Bottom
Command
===>
F3=Exit   F4=Prompt     F5=Refresh   F9=Retrieve   F10=Restart
F11=Display pool data   F12=Cancel   F14=Work with subsystems   F24=More keys
```

As the Work with System Status display shows, all other pools require attention to page faulting rates (both database and otherwise) and the job transitions. The guidelines for the faulting rates in these pools are based on the sum of the faults per second, both database and otherwise. In addition, you can adjust the processor speed and the PURGE attribute.

Figure 8-2 shows the general guidelines for each pool.

Figure 8-2. Sum of Database and Nondatabase Faulting Levels per Pool

Model	PURGE(*YES)			PURGE(*NO)		
	Good	**Acceptable**	**Poor**	**Good**	**Acceptable**	**Poor**
B10 B20 C04 C06 C10 C20 D04 D06 D10	<10	10–15	>15	<15	15–20	>20
B30 B35 B40 B45 C25 D20 D35	<15	15–20	>20	<20	20–25	>25
B50 B60 B70 D25 D45	<20	20–30	>30	<25	25–30	>30
D70 D80	<25	25–35	>35	<30	30–40	>40

In addition, you should observe the total number of faults in all pools. Figure 8-3 shows the general guideline for the total of the faults per second, both database and otherwise, in all pools.

Figure 8-3. Sum of Paging Faults, Database and Otherwise, in All Pools

Model	Good	Accept-able	Poor
B10 B20 C04 C06 C10 C20 D04 D06 D10	<20	20–25	>25
B30 B35 B40 B45 C25 D20 D35	<25	25–35	>35
B50 B60 B70 D25 D45	<30	30–45	>45
D70 D80	<40	40–50	>50

When observing job transitions, consider the following:

- Wait-to-ineligible transitions do not need to be 0 all the time. In heavy-use periods, it may be advisable to cause jobs to become ineligible to avoid excessive page fault rates.

  To determine the proper number of wait-to-ineligible transitions, divide the number of wait-to-ineligible transitions by the active-to-wait transitions. Compare your results to the following list.

  Good <.1
  Acceptable .1 – .25
  Poor >.25

- It is usually advisable to complete a transaction during a single time slice. This reduces the number of times the job enters and leaves main storage. Therefore, the active-to-ineligible transitions should be 0. However, long running transactions should not occupy activity levels for the entire transaction. You should establish a time slice value that allows 90% of the transactions in your environment to finish in a single time slice.

When using this display, remember that page fault rates are much more important than the job transition values. If you correctly tune the page fault rates, the transition rates usually fall within the guidelines.

You can increase or decrease the pool size or the activity level to get the desired values for pools 2 through 16. The mechanics of these actions are discussed under "Understanding the System Components that Affect Performance" on page 8-4.

Working with Disk Status

The WRKDSKSTS command shows you your system's disk activity and helps you determine the performance capabilities of your system's disks. Type the WRKDSKSTS command on the command line and press the Enter key. The Work with Disk Status display appears:

```
                              Work with Disk Status                    MP000
                                                                 03/29/92  11:04:19
Elapsed time:   00:01:55

                Size    %      I/O   Request   Read  Write   Read  Write    %
Unit  Type      (M)   Used     Rqs  Size (K)   Rqs   Rqs     (K)   (K)   Busy
  1   9332      200   98.9     .4      1.2      .0    .3     1.9   1.0      1
  2   9332      200   60.1     .6      1.0      .4    .2     1.0   1.0      2
  3   9332      200   61.8     .5      1.6      .2    .2     1.4   1.7      2
  4   9332      200   61.5     .6       .9      .3    .3      .9    .8      2
  5   9332      200   61.7     .7      1.9      .3    .4     2.5   1.4      2
  6   9332      200   60.5     .3      1.2      .1    .2     2.0    .7      1
  7   9332      200   63.2     .9      1.2      .3    .5     1.8    .9      3
  8   9332      200   56.3     .8      5.8      .4    .4     3.0   9.1      3
  9   9332      200   56.1     .6      4.2      .2    .3     1.3   6.4      2



                                                                      Bottom
Command
===>
F3=Exit    F5=Refresh    F12=Cancel    F24=More keys
```

Before observing disk status, tune your system according to the paging guidelines described in the topic "Working with System Status" on page 8-11. When viewing the Work with Disk Status display, observe the percent busy data. Each unit (actuator) should be less than 40% busy. If each unit is between 40% and 60% busy, you may experience variable response times. If each unit is more than 60% busy, you do not have enough actuators to provide good performance. The **actuator** is the device within an auxiliary storage device that moves the read and write heads. If you have a well-tuned system with actuators that exceed 40% busy, you should increase the number of disk actuators.

It is possible to experience inadequate performance even if only one actuator exceeds the 40% busy guideline. This may be caused by the placement of frequently used data on a single actuator. If this occurs on your system, use the AS/400 Performance Tools licensed program disk report to determine which data is causing the heavy use. You can save, delete, or restore some objects to improve performance.

An actuator may exceed the 40% guideline for a short period of time. This condition may be caused by a batch job that is accessing data. If the data is not concentrated on a particular actuator, the high level of use should migrate from actuator to actuator while the batch job is running. Also, an actuator in an auxiliary storage pool (ASP) may be used heavily. But typically this is not considered to exceed the guidelines. If you observe this activity, do not change the disk configuration.

Working with Active Jobs

The WRKACTJOB command measures system performance. The Work with Active Jobs display is shown in "Working with Active Jobs" on page 6-3.

Use both the WRKSYSSTS and the WRKACTJOB commands when observing your system's performance. With each observation period, you should examine and evaluate the measures of system performance against the goals you have set. Some of the typical measures include:

- Interactive throughput and response time, available from the WRKACTJOB display.
- Batch throughput. Observe the AuxIO and CPU% values for active batch jobs.
- Spool throughput. Observe the AuxIO and CPU% values for active writers.

Each time you make tuning adjustments, you should measure and compare all of your main performance measures. Make and evaluate adjustments one at a time.

Improving Performance Through the Database

Distributed relational database performance is affected by the overall design of the database as mentioned in Chapter 2, "Planning and Design for Distributed Relational Database" on page 2-1. Where you locate distributed data, the level of commitment control you use, and the design of your SQL indexes all affect performance.

Deciding Data Location

Because putting a network between an application and the data it needs will probably slow performance, consider the following when deciding where to put data:

- Transactions that use the data
- How often the transactions are performed
- How much data the transactions send or receive

If an application involves transactions that run frequently or that send or receive a lot of data, you should try to keep it in the same location as the data. For example, an application that runs many times a second or that receives hundreds of rows of data at a time will have better performance if the application and data are on the same system. Conversely, consider placing data in a different location than the application that needs it if the application includes low-use transactions or transactions that send or receive only moderate amounts of data at a time.

Factors that Affect Blocking

A very important performance factor is whether blocking occurs when data is transferred between the application requestor (AR) and the application server (AS). A group of rows transmitted as a block of data requires much less communications overhead than the same data sent one row at a time. One way to control blocking when connected to another AS/400 system is to use the SQL multiple-row INSERT and multiple-row FETCH statements in Version 2 Release 2 and later versions of the OS/400 operating system. The multiple-row FETCH forces the blocking of the number of rows specified in the FOR n ROWS clause, unless a hard error or end of data is encountered. The following discussion gives rules for determining if blocking will occur for single-row FETCHs.

Conditions that inhibit the blocking of query data between the AR and the AS are also listed in the following discussion. These conditions do not apply to the use of the multiple-row FETCH statement. Any condition listed under each of the following cases is sufficient to prevent blocking from occurring.

Case 1: AS/400 to AS/400

Blocking will not occur if:

- The cursor is updatable (see Note 1).
- The cursor is potentially updatable (see Note 2).
- The ALWBLK(*NONE) precompile option was used.
- The commitment control level is *ALL and the outer subselect does not contain one of the following:
  - The DISTINCT keyword
  - The UNION operator
  - An ORDER BY clause and the sum of the lengths of the fields in the clause requires a sort
  - A system database file reference
- The row size is greater than approximately 2K or, if the SBMRMTCMD command was used to extend the size of the default AS database buffer, the row size is greater than approximately half of the size of the database buffer resulting from specification of the OVRDBF SEQONLY number-of-records parameter. (Note that for the OVRDBF command to work remotely, OVRSCOPE(*JOB) must be specified.)
- The cursor is declared to be scrollable (DECLARE...SCROLL CURSOR...) and a scroll option specified in a FETCH statement is one of the following: RELATIVE, PRIOR, or CURRENT (unless a multiple-row FETCH was done, as mentioned above.)

Case 2: AS/400 to Non-AS/400

Blocking will not occur if:

- The cursor is updatable (see Note 1).
- The cursor is potentially updatable (see Note 2).
- The ALWBLK(*NONE) precompile option is used.
- The row size is greater than approximately 16K.

Case 3: Non-AS/400 to AS/400

Blocking will not occur if:

- The cursor is updatable (see Note 1).
- The cursor is potentially updatable (see Note 2).
- A precompile or bind option is used that caused the package default value to be force-single-row protocol.
  - For DB2, there is no option to do this.
  - For SQL/DS, this is the NOBLOCK keyword on SQLPREP (the default).
  - For OS/2, this is /K=NO on SQLPREP or SQLBIND.
- The row size is greater than approximately 0.5*QRYBLKSIZ. (The default QRYBLKSIZ values for DB2, SQL/DS, and OS/2 are 32K, 8K, and 4K, respectively.)

Summarization of rules

In general, what these rules (including the notes) say is that if you have a program that does not include dynamic statements and you bind it with the precompile/bind option for single-row protocol (/K=UNAMBIG for OS/2 and RS/6000, ALWBLK(*READ) for OS/400, CURRENTDATA(YES) for DB2, and SBLOCK for SQL/DS), blocking will occur if:

- The cursor is read-only (see note 3), or,
- There is no FOR UPDATE OF clause in the SELECT, and,
- There are no UPDATE or DELETE WHERE CURRENT OF statements against the cursor in the program.

Note that the above query blocking precompile options are the defaults for all ARs listed except SQL/DS.

If you do have dynamic statements in the program, but you bind the program with the precompile/bind option for limited-block protocol (/K=ALL for OS/2 and RS/6000, ALWBLK(*ALLREAD) for OS/400, CURRENTDATA(NO) for DB2, and BLOCK for SQL/DS), data is still blocked for cursors that satisfy the other conditions listed in the previous rule (i.e., they are read-only, or they have no FOR UPDATE OF clause in the SELECT and there are no UPDATE or DELETE WHERE CURRENT OF statements against the cursor in the program).

Other factors that prevent blocking are:

- Retrieval of row data that is too long to be blocked (unlikely).
- Use of a precompile/bind option to force single-row protocol (uncommon, except possibly with an SQL/DS AR where NOBLOCK is the default).

In an AS/400 only environment, two other conditions can prevent blocking:

- The use of COMMIT(*ALL), except in certain cases.
- The use of certain scroll options on a scrollable cursor.

Notes:

1. A cursor is updatable if it is not read-only (see Note 3), and one of the following is true:
   - The select statement contained the FOR UPDATE OF clause, or
   - There exists in the program an UPDATE or DELETE WHERE CURRENT OF against the cursor.
2. A cursor is potentially updatable if it is not read-only (see Note 3), and if the program includes an EXECUTE or EXECUTE IMMEDIATE statement (or when connected to a non-AS/400 system, any dynamic statement), and a precompile or bind option is used that caused the package default value to be single-row protocol.
   - For AS/400, this is the ALWBLK(*READ) precompile option (the default).
   - For DB2, this is CURRENTDATA(YES) on BIND PACKAGE (the default).
   - For SQL/DS, this is the SBLOCK keyword on SQLPREP.
   - For OS/2, this is /K=UNAMBIG on SQLPREP or SQLBIND (the default).
3. A cursor is read-only if one or more of the following conditions are true:
   - The DECLARE CURSOR statement specified an ORDER BY clause but did not specify a FOR UPDATE OF clause.

- The DECLARE CURSOR statement specified a FOR FETCH ONLY clause.
- The DECLARE CURSOR statement specified the SCROLL keyword without DYNAMIC (OS/400 only).
- One or more of the following conditions are true for the cursor or a view or logical file referenced in the outer subselect to which the cursor refers:
  - The outer subselect contains a DISTINCT keyword, GROUP BY clause, HAVING clause, or a column function in the outer subselect.
  - The select contains a join function.
  - The select contains a UNION operator.
  - The select contains a subquery that refers to the same table as the table of the outer-most subselect.
  - The select contains a complex logical file that had to be copied to a temporary file.
  - All of the selected columns are expressions, scalar functions, or constants.
  - All of the columns of a referenced logical file are input only (OS/400 only).

Factors That Affect the Size of Query Blocks

If a large amount of data is being returned on a query, performance may be improved by increasing the size of the block of query data. The way that this is done depends upon the types of systems participating in the query. In an unlike environment, the size of the query block is determined at the application requester by a parameter sent with the Open Query command. When an AS/400 system is the AR, it always requests a query block size of 32K. Other types of ARs give the user a choice of what block size to use. The default query block sizes for DB2, SQL/DS, and OS/2 are 32K, 8K, and 4K, respectively. See the product documentation for the platform being used as AR when an AS/400 server is connected to an unlike AR.

In the AS/400 to AS/400 environment, the query block size is determined by the size of the buffer used by the database manager. The default size is 4K. This can be changed on application servers that are at the Version 2, Release 3 or higher level. In order to do this, use the SBMRMTCMD CL command to send and execute an OVRDBF command on the AS. Besides the name of the file being overridden, the OVRDBF command should contain OVRSCOPE(*JOB) and SEQONLY(*YES nnn). The number of records desired per block replaces nnn in the SEQONLY parameter. Increasing the size of the database buffer not only can reduce communications overhead, but can also reduce the number of calls to the database manager to retrieve the rows.

Effectively Designing an SQL Index

SQL provides two basic means for accessing tables: a table scan (sequential) and an index-based (direct) retrieval. Index-based retrieval is usually more efficient than table scan. However, when a very large percentage of pages are retrieved, table scan is more efficient than index-based retrieval.

If SQL cannot use an index to access the data in a table, it must read all the data in the table. Very large tables present a special performance problem: the high cost of retrieving all the data in the table. The following suggestions help you to design code that allows SQL to take advantage of available indexes:

- Avoid numeric conversions

- Avoid character string padding
- Avoid the use of LIKE patterns beginning with % or _
- Avoid the use of like patterns specified in host variables
- Be aware that SQL does not use an index in some instances
- Specify ORDER BY when using OR or IN predicates

Avoid Numeric Conversions

When comparing a column value and a host variable (or literal value), try to specify the same data types and attributes. SQL does not use an index for the named column if the host variable or literal value has a greater precision than the precision of the column. If the two items being compared have different data types, SQL will have to convert one or the other of the values, which can result in inaccuracies (because of limited machine precision). For example, EDUCLVL is a halfword integer value (SMALLINT). Specify:

```
... WHERE EDUCLVL < 11 AND
          EDUCLVL >= 2
```

instead of

```
... WHERE EDUCLVL < 1.1E1 AND
          EDUCLVL > 1.3
```

Avoid Character String Padding

Try to use the same data length when comparing a fixed-length character string column value to a host variable or literal value. SQL does not use an index if the literal value or host variable is longer than the column length. For example, EMPNO is CHAR(6) and DEPTNO is CHAR(3). Specify:

```
... WHERE EMPNO > '000300' AND
          DEPTNO < 'E20'
```

instead of

```
... WHERE EMPNO > '000300 ' AND
          DEPTNO < 'E20 '
```

Avoid the Use of LIKE Patterns Beginning With % or _

The percent sign (%), and the underline (_), when used in the pattern of a LIKE predicate, specify a character string that is similar to the column value of rows you want to select. When used to denote characters in the middle or at the end of a character string, as in

```
... WHERE LASTNAME LIKE 'J%SON%'
```

they can take advantage of SQL indexes. However, when used at the beginning of a character string, as in

```
... WHERE LASTNAME LIKE '%SON'
```

they can prevent SQL from using any indexes that might be defined on the *LASTNAME* column to limit the number of rows scanned. You should therefore avoid using these symbols at the beginning of character strings, especially if you are accessing a particularly large table.

Avoid the Use of Like Patterns Specified in Host Variables

When the character pattern of a LIKE predicate is specified in a host variable, an index will not be used to select the rows. If the character pattern cannot be specified as a constant then, you might consider using dynamic SQL to prepare the SELECT statement with the current value of the like pattern placed into the SELECT statement text as a character string literal.

Be Aware That SQL Does Not Use an Index in Some Instances

The SQL does not use an index in the following instances:

- For a column that is expected to be updated; for example, your program might include

```
EXEC SQL
  DECLARE DEPTEMP CURSOR FOR
    SELECT EMPNO, LASTNAME, DEPTNO
      FROM USER1.TEMPL
      WHERE (DEPTNO = 'D11' OR
             DEPTNO = 'D21') AND
            EMPNO >= '000190'
      FOR UPDATE OF EMPNO, DEPTNO
END-EXEC.
```

  even if you do not intend to update the employee's serial number. In this example, SQL cannot use an index with a key of EMPNO or DEPTNO.

  SQL can operate more efficiently if the FOR UPDATE OF column list only names the column you intend to update: *DEPTNO*. Therefore, do not specify a column in the FOR UPDATE OF column list unless you intend to update the column.

- For a column being compared with another column from the same row. For example:

```
EXEC SQL
  DECLARE DEPTDATA CURSOR FOR
    SELECT DEPTNO, DEPTNAME
      FROM USER1.TDEPT
      WHERE DEPTNO = ADMRDEPT
END-EXEC.
```

  Even though there is an index for *DEPTNO* and another index for *ADMRDEPT*, SQL will not use either index. The index has no added benefit because every row of the table needs to be looked at.

Specify ORDER BY When Using OR or IN Predicates

The database manager will use an index to select rows of data for queries which use the IN predicate with a list of values or have multiple predicates connected by the OR Boolean operator if:

- Each predicate can use an index to evaluate. That is:
  - The data type attributes are compatible.
  - The predicates do not include arithmetic expressions.
  - The predicates do not include subqueries.
- The predicates refer to the same column.
- The ORDER BY clause specifies the comparison column as the first column.

To enable possible index usage, it is best to use only the AND Boolean operator. However, the application programmer can cause the database manager to use an index for the selection by specifying the ORDER BY clause. The database manager will always attempt to use an index to provide the ordering specified by the ORDER BY clause. If it is known that an index is available over the column being selected then specifying an ORDER BY for that column can improve performance. Care must be taken to ensure that an index matching the ORDER BY is available otherwise the database manager may create a temporary index.

The ORDER BY clause specified in the following statement will ensure that an index will be used to select the rows:

```
EXEC SQL
  DECLARE DEPTTEMP CURSOR FOR
  SELECT EMPNP
    FROM USER1.TEMPL
    WHERE DEPTNO = 'D11'
    OR DEPTNO = 'D21'
    ORDER BY DEPTNO
END-EXEC.
```

Performance Effect of Host Variables in WHERE Clauses

The OS/400 SQL optimizer may not use otherwise appropriate indexes in constructing a plan to access table data. This can happen when certain relationships exist between the data fields being compared in a WHERE clause of a static SQL statement. It is safest to always keep the data type of a host variable identical to that of the SQL column to which it is being compared. The rules that govern the choice of index are:

1. An index is not used to resolve numeric selection using single precision floating point host variables when compared against any numeric format other than floating point.
2. An index is not used to resolve numeric selection using double precision floating point host variables when compared against any numeric format other than double precision floating point.
3. An index is not used to resolve numeric selection using small integer host variables when compared against numeric or decimal keys that have fewer than five digits to the left of the decimal point.
4. An index is not used to resolve numeric selection using large integer host variables when compared against small integer keys or numeric or decimal keys that have fewer than eleven digits to the left of the decimal point.

Performance Information Messages

You can evaluate the response to and performance of the SQL statements in a program by using informational messages put in the job log by the database manager. The database manager issues the messages for a program running in debug mode. The resulting messages found in the job log also provide a trace of the SQL statements processed in an application.

For a description of the messages that can be issued when running in debug mode, refer to the performance verification phase information in the *SQL/400 Programmer's Guide*

When an application uses DRDA, the SQL statements are run in the application server job. Because of this, you must start debug mode for the application server job that is running on the OS/400 operating system. For a procedure on doing this, see "Starting a Service Job to Diagnose Application Server Problems" on page 9-26.

Chapter 9. Handling Distributed Relational Database Problems

When a problem occurs accessing a distributed relational database it is the job of the administrator to:

- Determine the nature of the problem, and
- Determine if it is a problem with the application or a problem with the local or remote system.

You must then resolve the problem or obtain customer support assistance to resolve the problem. To do this, you need:

- An understanding of the OS/400 program support.
- A good idea of how to decide if a problem is on an application requester (AR) or an application server (AS).
- Familiarity with using OS/400 problem management functions.

This chapter provides a distributed relational database problem handling overview. It describes how to isolate distributed relational database problems, how to work with users to look at displays and messages, and how to look at problems for an application that has failed. It also presents information on using the job log and AS/400 system alerts to manage problems. Finally, this chapter provides information on obtaining data to report a failure and help diagnose a problem.

For more information about diagnosing problems in a distributed relational database, see the *Distributed Relational Database Problem Determination Guide*.

AS/400 Problem Handling Overview

The OS/400 program helps you manage problems for both user- and system-detected problems that occur on local and remote AS/400 systems. Problem handling support includes:

- Messages with initial problem handling information
- Automatic alerting of system-detected problems
- Alert management focal point capability
- Integrated problem logging and tracking
- First failure data capture (FFDC) support
- Electronic customer support service requisition
- Electronic customer support, program temporary fix (PTF) requisition

The AS/400 system and its attached devices are able to detect some types of problems. These are called **system-detected problems**. When a problem is detected, several operations take place:

- An error log entry is created
- A problem record is created
- A message is sent to the QSYSOPR message queue
- An alert may be created

Information is recorded in the error log and the problem record. The alert is then sent to the service provider if the service provider is either an alert focal point or

the network node server for the system with the problem. When some alerts are sent, a spooled file of FFDC information is also created. The error log and the problem record may contain the following information:

- Vital product data
- Configuration information
- Reference code
- The name of the associated device
- Additional failure information

User-detected problems are usually related to program errors that can cause any of the following problems to occur:

- Job problems
- Incorrect output
- Messages indicating a program failure
- Device failure not detected by the system
- Poor performance

When a user detects a problem, no information is gathered by the system until problem analysis is run or you select the option to save information to help resolve a problem from the Operational Assistant USERHELP menu.

The AS/400 system tracks both user- and system-detected problems using the problem log and problem manager. A problem state is maintained from when a problem is detected (OPENED) to when it is resolved (CLOSED) to assist you with tracking. Alert and alert management capabilities extend the problem management support to include problems occurring on other AS/400 systems in a distributed relational database network. For more information, see "AS/400 Problem Log" on page 9-18.

Isolating Distributed Relational Database Problems

A problem you encounter when running a distributed relational database application can exhibit two general symptoms:

- The user receives incorrect output
- The application does not complete in the expected time

The diagrams and procedures below show generally how you can classify problems as application program problems, performance related problems, and system related problems, so you can use standard AS/400 system problem analysis methods to resolve the problem.

Incorrect Output

If you receive an error message, use the error message, SQLCODE, or SQLSTATE to determine the cause of the problem. See Figure 9-1 on page 9-3. The message description indicates what the problem is and provides corrective actions. If you do not receive an error message, you must determine whether distributed relational database is causing the failure. To do this, run the failing statement locally on the AS or use interactive Structured Query Language (SQL) to run the statement on the AS. If you can create the problem locally, the problem is not with distributed relational database support. Use AS/400 problem analysis methods to provide specific information for your support staff depending on the results of this operation.

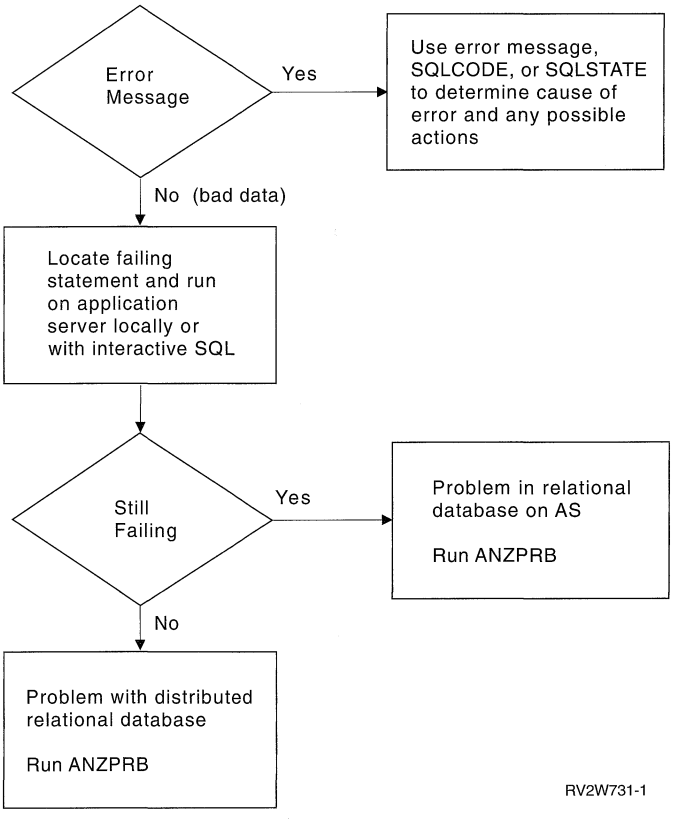


Figure 9-1. Resolving Incorrect Output Problem

Application Does Not Complete in the Expected Time

If the request takes longer than expected to complete, the first place to check is at the AR. Check the job log for message SQL7969 which indicates that a connect to a relational database is complete. This tells you the application is a distributed relational database application. Check the AR for a loop by using the Work with Job (WRKJOB) command to display the program stack, and check the program stack to determine whether the system is looping. See Figure 9-2 on page 9-4. If the application itself is looping, contact the application programmer for resolution. If you see QAPDEQUE and QCNSRCV on the stack, the AR is waiting for the AS. See Figure 9-3 on page 9-5. If the system is not in a communications wait state, use problem analysis procedures to show whether there is a performance problem or a wait state somewhere else.

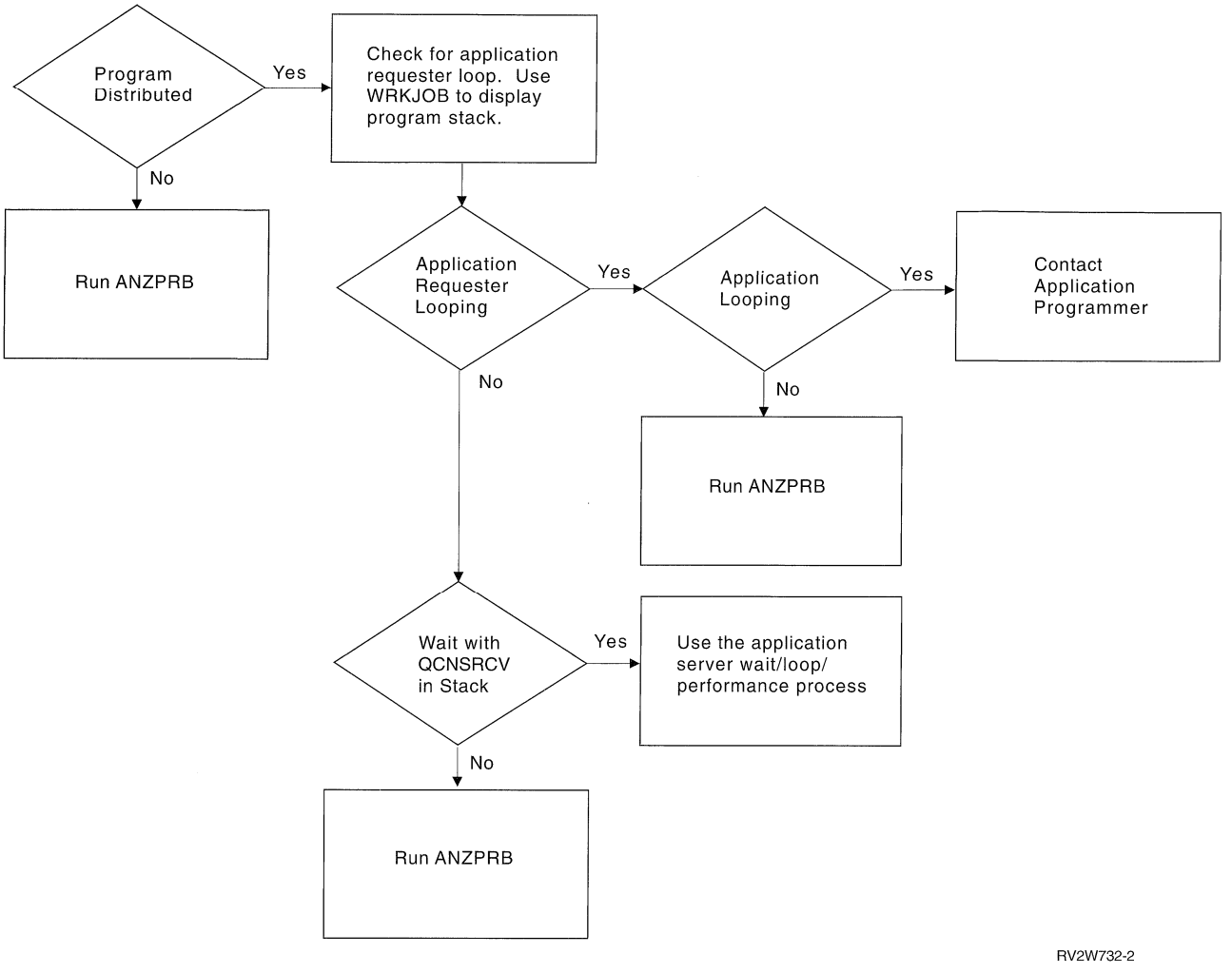


Figure 9-2. Resolving Wait, Loop, or Performance Problems on the Application Requester

You can find the AS job name by looking at the job log on the AS. For more information about finding jobs on the AS, see "Locating Distributed Relational Database Jobs" on page 6-5. When you need to check the AS job, use the Work with Job (WRKJOB), Work with Active Job (WRKACTJOB), or Work with User Job (WRKUSRJOB) commands to locate the job on the AS. For information on using these commands, see "Working with Jobs" on page 6-1, "Working with User Jobs" on page 6-2, and "Working with Active Jobs" on page 6-3. From one of these job displays, look at the program stack to see if the AS is looping. If it is looping, use problem analysis to handle the problem. If it is not looping, check the program stack for WAIT with QCNSRCV, which means the AS is waiting for the AR. If both systems are in this communications wait state, there is a problem with your network. If the AS is not in a wait state, there is a performance issue that may have to be addressed.

Two common sources of slow query performance are:

- An accessed table does not have an index. If this is the case, create an index using an appropriate field or fields as the key. For more information, see "Effectively Designing an SQL Index" on page 8-18.
- The rows returned on a query request are not blocked. Whether the rows are blocked can cause a significant difference in query performance. It is important

to understand the factors that affect blocking, and tune the application to take advantage of it. For more information, see "Factors that Affect Blocking" on page 8-15.

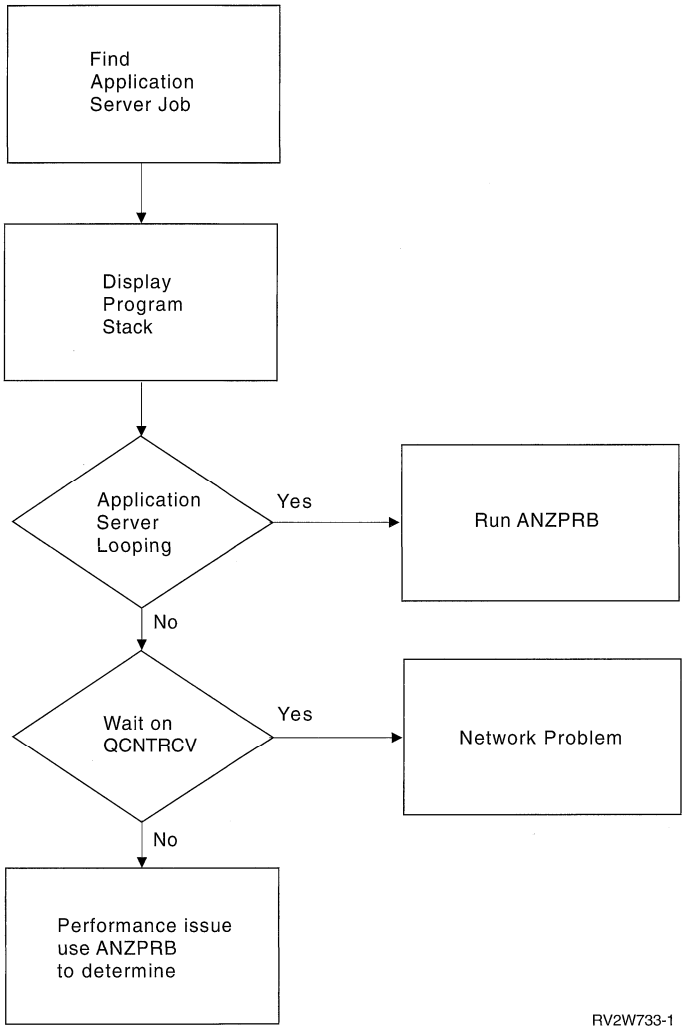


Figure 9-3. Resolving Wait, Loop, or Performance Problems on the Application Server

Working with Users

Investigating a problem usually begins with the user. Users may not be getting the results they expect when running a program or they may get a message indicating a problem. Sometimes the best way to diagnose and solve a problem is to step through the procedure with a user. The copy screen function allows you to do this either in real time with the user or in examining a file of the displays the user saw previously.

You can also gather more information from a message than just the line of text that appears at the bottom of a display. This section discusses how you can copy displays being viewed by another user and how you can obtain more information about messages you or a user receive when doing distributed relational database work.

Copy Screen

The Start Copy Screen (STRCPYSCN) command allows you to be signed on to your work station and see the same displays being viewed by someone else at another work station. You must be signed on to the same AS/400 system as the user. If that user is on a remote system, you can use display station pass-through to sign on that system and then enter the STRCPYSCN command to see the other displays. Screen images can be copied to a database file at the same time they are copied to another work station or when another work station cannot be used. This allows you to process this data later and prepares an audit trail for the operations that occur during a problem situation.

To copy the display image to another display station the following requirements must be met:

- Both displays are defined to the system
- Both displays are color or both are monochrome, but not one color and the other monochrome
- Both displays have the same number of character positions horizontally and vertically

If you type your own display station ID as the sending device, the receiving display station must have the sign on display shown when you start copying screen images. Graphics are copied as blanks.

If not already signed on to the same system, use the following process to see the displays that another user sees on a remote system:

1. Enter the Start Pass-Through (STRPASTHR) command.

   ```
   STRPASTHR RMTLOCNAME(KC105)
   ```

2. Log on to the target system.
3. Enter the STRCPYSCN command.

   ```
   STRCPYSCN SRCDEV(KC105)
             OUTDEV(*REQUESTER)
             OUTFILE(KCHELP/TEST)
   ```

   - SRCDEV specifies the name of the source device, the display station that is sending the display image. To send your display to command to another device, enter the *REQUESTER value for this parameter.
   - OUTDEV specifies the name of the output device to which the display image is sent. In this example the display image is sent to the display station of the person who enters the command (*REQUESTER). You can also name another display station, another device (where a third user is viewing), or to no other device (*NONE). When the *NONE value is used, specify an output file for the display images.
   - OUTFILE specifies the name of the output file that will contain an image of all the displays viewed while the command is active.

4. An inquiry message is sent to the source device to notify the user of that device that the displays will be copied to another device or file. Type a `g` (Go) to start sending the images to the requesting device.

The sending display station's screens are copied to the other display station. The image shown at the receiving display station trails the sending display station by one screen. If the user at the sending display station presses a key that is not active (such as the Home key), both display stations will show the same display.

While you are copying screens, the operator of the receiving display station cannot do any other work at that display station until the copying of screens is ended.

To end the copy screen function from the sending display station, enter the End Copy Screen (ENDCPYSCN) command from any command line and press the Enter key.

```
ENDCPYSCN
```

The display you viewed when you started the copy screen function is shown.

Messages

The AS/400 system sends a variety of system messages that indicate conditions ranging from simple typing errors to problems with system devices or programs. The message may be one of the following:

- An error message on your current display.

  These messages can interrupt your job or sound an alarm. You can display these messages by typing `DSPMSG` on any command line.

- A message regarding a system problem that is sent to the system operator message queue and displayed on a separate Work with Messages display.

  To see these messages, type `DSPMSG QSYSOPR` on any system command line.

- A message regarding a system problem that is sent to the message queue specified in a device description.

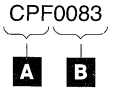
  To see these messages, type `DSPMSG message-queue-name` on any system command line.

- A message regarding a system problem that is sent to another system in the network.

  These messages are called alerts. See "Alerts" on page 9-19 for how to view and work with alerts.

The system sends informational or inquiry messages for certain system events. **Informational messages** give you status on what the system is doing. **Inquiry messages** give you information about the system, but also request a reply.

In some message displays a message is accompanied by a letter and number code such as:



RV2W742-0

The first two or three letters **A** indicate the message category. Some message categories for distributed relational database are:

Figure 9-4. Message Categories

Category	Description	Library
CPA through CPZ	Messages from the operating system	QSYS/QCPFMSG
MCH	Licensed internal code messages	QSYS/QCPFMSG
SQ and SQL	Structured Query Language (SQL) messages	QSYS/QSQLMSG

The remaining four digits **B** (five digits if the prefix is SQ) indicate the sequence number of the message. The example message ID shown indicates this is a message from the operating system, number 0083.

To obtain more information about a message on the message line of a display or in a message queue, do the following:

1. Move the cursor to the same line as the message.
2. Press the Help key. The Additional Message Information display is shown.

```
                      Additional Message Information

Message ID  . . . . . :   CPD6A64          Severity . . . . . :   30
Message type  . . . . :   DIAGNOSTIC
Date sent . . . . . :   03/29/92          Time sent  . . . . :   13:49:06
From program  . . . . :   QUIACT          Instruction  . . . . :   080D
To program  . . . . . :   QUIMGFLW        Instruction  . . . . :   03C5

Message . . . . :   Specified menu selection is not correct.
Cause . . . . . :   The selection that you have specified is not correct for
  one of the following reasons:
    -- The number selected was not valid.
    -- Something other than a menu option was entered on the option line.
Recovery  . . . :   Select a valid option and press the Enter or Help key
  again.




                                                                      Bottom
Press Enter to continue.

F3=Exit          F10=Display messages in job log         F12=Cancel
```

You can get more information about a message that is not showing on your display if you know the message identifier and the library in which it is located. To get this information enter the Display Message Description (DSPMSGD) command:

```
DSPMSGD RANGE(SQL0204) MSGF(QSYS/QSQLMSG)
```

This command produces a display that allows you to select the following information about a message:

- Message text
- Field data
- Message attributes
- All of the above

The text is the same message and message help text that you see on the Additional Message Information display. The field data is a list of all the substitution variables defined for the message and their attributes. The message attributes are

the values (when defined) for severity, logging, level of message, alert, default program, default reply, and dump parameters. You can use this information to help you determine what the user was doing when the message appeared.

Message Types

On the Additional Message Information display you see the message type and severity code for the message. Figure 9-5 shows the different message types for AS/400 messages and their associated severity codes:

Figure 9-5. Message Severity Codes

Message Type	Severity Code
Informational messages. For informational purposes only; no reply is needed. The message can indicate that a function is in progress or that a function has completed successfully.	00
Warning. A potential error condition exists. The program may have taken a default, such as supplying missing data. The results of the operation are assumed to be successful.	10
Error. An error has been found, but it is one for which automatic recovery procedures probably were applied; processing has continued. A default may have been taken to replace the wrong data. The results of the operation may not be correct. The function may not have completed; for example, some items in a list ran correctly, while other items did not.	20
Severe error. The error found is too severe for automatic recovery procedures and no defaults are possible. If the error was in the source data, the entire data record was skipped. If the error occurred during a program, it leads to an abnormal end of program (severity 40). The results of the operation are not correct.	30
Severe error: abnormal end of program or function. The operation has ended, possibly because the program was not able to handle data that was not correct or because the user canceled it.	40
Abnormal end of job or program. The job was not started or failed to start, a job-level function may not have been done as required, or the job may have been canceled.	50
System status. Issued only to the system operator message queue. It gives either the status of or a warning about a device, a subsystem, or the system.	60
Device integrity. Issued only to the system operator message queue, indicating that a device is not working correctly or is in some way no longer operational.	70
System alert and user messages. A condition exists that, although not severe enough to stop the system now, could become more severe unless preventive measures are taken.	80
System integrity. Issued only to the system operator message queue. Describes a condition where either a subsystem or system cannot operate.	90
Action. Some manual action is required, such as entering a reply or changing printer forms.	99

Distributed Relational Database Messages

If an error message occurs at either an AS or an AR, the system message is logged on the job log to indicate the reason for the failure. See "Using the Job Log" on page 6-4 for information on how to use a job log and locate one on an AS.

A system message exists for each SQLCODE returned from an SQL statement supported by the SQL/400 program. The message is made available in precompiler listings, on interactive SQL, or in the job log when running in the debug mode. However, when you are working with an AS that is not an AS/400 system, there may not be a specific message for every error condition in the following cases:

- The error is associated with a function not used by the AS/400 system.

  For example, the AS/400 system does not support referential integrity, so SQLCODE -532 (SQLSTATE 23504) "The operation is prevented by the RESTRICT update or delete rule" does not exist.

- The error is product-specific and will never occur on the AS/400 system.

  The AS/400 system will never have SQLCODE -925 (SQLSTATE 56021), "SQL commit or rollback is invalid in an IMS or CICS environment."

For SQLCODEs that do not have corresponding messages, a generic message is returned that identifies the unrecognized SQLCODE, SQLSTATE, and tokens, along with the relational database name of the AS which generated the message. To determine the specific condition and how to interpret the tokens, consult the product documentation corresponding to the particular release of the connected AS. For more information on SQLCODEs, see "SQLCODEs and SQLSTATEs" on page 9-15.

Messages in the ranges CPx3E00 through CPx3EFF and CPI9100 through CPI91FF are used for distributed relational database system messages. The following list is not inclusive, but shows more common system messages you may see in a distributed database job log on an AS/400 system. See the *SQL/400* Programmer's Guide* for a list of SQL/400 messages for distributed relational database.

Figure 9-6 (Page 1 of 2). Distributed Relational Database Messages

MSG ID	Description
CPD3ECA	RDB directory operation timed out
CPD3E01	DBCS or MBCS CCSID not supported.
CPD3E03	Local RDB name not in RDB directory
CPD3E05	DDM conversation path not found
CPF3EC9	Scope message for interrupt RDB
CPF3E0A	Resource limits error
CPF3E0B	Query not open
CPF3E0C	FDOCA LID limit reached
CPF3E0D	Interrupt not supported
CPF3E01	DDM parameter value not supported
CPF3E02	AR cannot support operations
CPF3E04	SBCS CCSID not supported

Figure 9-6 (Page 2 of 2). Distributed Relational Database Messages

MSG ID	Description
CPF3E05	Package binding not active
CPF3E06	RDB not found
CPF3E07	Package binding process active
CPF3E08	Open query failure
CPF3E09	Begin bind error
CPF3E10	AS does not support DBCS or MC
CPF3E12	Commit/rollback HOLD not supported
CPF3E13	Commitment control operation failed
CPF3E14	End RDB Request failed
CPF3E16	Not authorized to RDB
CPF3E17	End RDB request is in progress
CPF3E18	COMMIT/ROLLBACK with SQLCA
CPF3E19	Commitment control operation failed
CPF3E20	DDM conversation path not found
CPF3E21	RDB interrupt fails
CPF3E80 *	Data stream syntax error
CPF3E81 *	Invalid FDOCA descriptor
CPF3E82 *	ACCRDB sent twice
CPF3E83 *	Data mismatch error
CPF3E84 *	DDM conversational protocol error
CPF3E85 *	RDB not accessed
CPF3E86 *	Unexpected condition
CPF3E87 *	Permanent agent error
CPF3E88 *	Query already open
CPF3E89 *	Query not open
CPF3E99	End RDB request has occurred
CPI9150	DDM job started
CPI9152	Target DDM job started by source system
CPI3E01	Local RDB accessed successfully
CPI3E02	Local RDB disconnected successfully

Note: An asterisk (*) means an alert is associated with the error condition.

Handling Program Start Request Failures

When a program start request is received by an OS/400 subsystem on the AS, the system attempts to start a job based on information sent with the program start request. The AR user's authority to the AS system, existence of the requested database, and many other items are checked.

If the AS subsystem determines that it cannot start the job (for example, the user profile does not exist on the AS, the user profile exists but is disabled, or the user is not properly authorized to the requested objects on the AS), the subsystem

sends a message, CPF1269, to the QSYSMSG message queue (or QSYSOPR when QSYSMSG does not exist). The CPF1269 message contains two reason codes (one of the reason codes may be zero, which can be ignored).

The nonzero reason code gives the reason the program start request was rejected. Because the remote job was to have started on the AS, the message and reason codes are provided on the AS system, and not the AR system. The user at the AR only knows that the program start request failed, not why it failed. The user on the AR must either talk to the system operator at the AS system, or use display station pass-through to the AS to determine the reason why the request failed.

For a complete description of the reason codes and their meanings, refer to the *ICF Programmer's Guide*.

Application Problems

The best time to handle a problem with an application is before it goes into production. However, it is impossible to anticipate all the conditions that will exist for an application when it gets into general use. The job log of either the AR or the AS can tell you that a package failed; the listing of the program or the package can tell you why it failed. The SQL/400 compilers provide diagnostic tests that show the SQLCODEs generated by the precompile process on the diagnostic listing. You can optionally specify *SOURCE and *XREF on the OPTIONS parameter of the Create SQL Program (CRTSQL*xxx*) commands to print a precompile source and cross-reference listings.

Listings

The listing from the Create SQL program (CRTSQL*xxx*) command shown in Figure 9-7 on page 9-13 provides the following kinds of information:

- The values supplied for the parameters of the precompile command
- The program source
- The identifier cross-references
- The messages resulting from the precompile

Precompiler Listing

```
5738ST1 V2R1M1 920329                    IBM SAA SQL/400       UPDATEPGM                          03/29/92 11:25:51
Source type..............C
Program name.............WULF/UPDATEPGM
Source file..............WULF/QCSRC
Member...................UPDATEPGM
Options..................*SRC      *XREF
Target release...........*CURRENT
INCLUDE file.............*LIBL/*SRCFILE
Commit...................*NONE
Allow copy of data.......*YES
Close SQL cursor.........*ENDPGM
Allow blocking...........*READ
Delay PREPARE............*NO
Generation level.........10
Margins..................*SRCFILE
Printer file.............*LIBL/QSYSPRT
Date format..............*JOB
Date separator...........*JOB
Time format..............*HMS
Time separator ..........*JOB
Replace..................*YES
Relational database.....KC000
Default collection.......*NONE
Package name.............*PGMLIB/*PGM
SAA flagging.............*NOFLAG
ANS flagging.............*NONE
Text.....................*SRCMBRTXT
Source file CCSID.......37
Job CCSID...............37
Source member changed on 03/29/92  11:20:13


5738ST1 V2R1M1 920329                    IBM SAA SQL/400       UPDATEPGM                          03/29/92 11:25:51
Record  *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8    SEQNBR  Last change
    1   /*****************************************************************/                  100    03/29/92
    2   /*  This program is called to update the DEPTCODE of file RWDS/DPT1 */               200    03/29/92
    3   /* to NULL.  This is run once a month to clear out the old         */               300    03/29/92
    4   /* data.                                                           */               400    03/29/92
    5   /*                                                                 */               500    03/29/92
    6   /*  NOTE: Because this program was compiled with an RDB name, it is  */             600    03/29/92
    7   /* not necessary to do a connect, as an implicit connect will take  */             700    03/29/92
    8   /* place when the program is called.                               */               800    03/29/92
    9   /*****************************************************************/                  900    03/29/92
   10   #include <stdio.h>                                                                 1000    03/29/92
   11   #include <stdlib.h>                                                                1100    03/29/92
   12   exec sql include sqlca;                                                            1200    03/29/92
   13                                                                                      1300    03/29/92
   14   main()                                                                             1400    03/29/92
   15     {                                                                                1500    03/29/92
   16        /*  Just update RWDS/DPT1, setting deptcode = NULL        */                  1600    03/29/92
   17        exec sql update RWDS/DPT1                                                   1700    03/29/92
   18            set deptcode = NULL;                                                    1800    03/29/92
   19     }                                                                              1900    03/29/92
                          * * * * *  E N D  O F  S O U R C E  * * * * *
```

Figure 9-7 (Part 1 of 2). Listing From a Precompiler

```
                                                IBM INTERNAL USE ONLY
5738ST1 V2R1M1 920329                     IBM SAA SQL/400       UPDATEPGM                    03/29/92 11:25:51
                                             CROSS REFERENCE
Data Names                      Define    Reference
"DEPTCODE"                         ****     COLUMN
                                            18
"DPT1"                             ****     TABLE IN "RWDS"
                                            17
"RWDS"                             ****     COLLECTION
                                            17


5738ST1 V2R1M1 920329                      IBM SAA SQL/400       UPDATEPGM                    03/29/92 11:25:51
                                             DIAGNOSTIC MESSAGES
MSG ID  SEV  RECORD  TEXT
SQL0088   0      17  Position 15 UPDATE applies to entire table.
SQL1108  10      17  Field definitions for file DPT1 in RWDS at KC000 not found.
                                               Message Summary
Total    Info    Warning    Error    Severe    Terminal
    2       1          1        0         0           0
10 level severity errors found in source
   19 Source records processed
                             * * * * *  E N D  O F  L I S T I N G  * * * * *
```

Figure 9-7 (Part 2 of 2). Listing From a Precompiler

CRTSQLPKG Listing

The listing from the Create SQL Package (CRTSQLPKG) command shown in Figure 9-8 provides two types of information:

- The values used on the parameters of the command
- The statement in error, if any
- The messages resulting from running the CRTSQLPKG command

```
5738SS1 V2R1M1 920329                        Create SQL package                              03/29/92 11:26:33
Record  *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8    SEQNBR  Last change
Program name..............WULF/UPDATEPGM
Relational database.......*PGM
Replace...................*YES
Default Collection........*PGM
Generation level..........10
Printer file..............*LIBL/QSYSPRT
Text......................*PGMTXT
Source file...............WULF/TEST
Member....................UPDATEPGM
```

Figure 9-8 (Part 1 of 2). Listing from CRTSQLPKG

```
5738SS1 V2R1M1 920329                    Create SQL package                    03/29/92 11:26:33
Record  *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8   SEQNBR  Last change
   17   UPDATE RWDS / DPT1 SET deptcode = NULL
                                          DIAGNOSTIC MESSAGES
MSG ID  SEV  RECORD  TEXT
SQL0204  10      17  Position 8 Object DPT1 in RWDS type *FILE not found.
SQL5037              SQL Package UPDATEPGM in WULF created at relational database KC000.
                                           Message Summary
Total    Info    Warning    Error    Severe    Terminal
   1      0         1         0         0          0
10 level severity errors found in source
                             * * * * *  E N D  O F  L I S T I N G  * * * * *
```

Figure 9-8 (Part 2 of 2). Listing from CRTSQLPKG

SQLCODEs and SQLSTATEs

SQL returns error codes to the application program when an error occurs. SQLCODEs and their corresponding SQLSTATEs are returned in the SQL communication area (SQLCA) structure. An SQLCA is a collection of variables that is updated with information about the SQL statement most recently run.

When an SQL error is detected, a return code called an SQLCODE is returned. If SQL encounters a hard error while processing a statement, the SQLCODE is a negative number (for example, SQLCODE -204). If SQL encounters an exceptional but valid condition (warning) while processing a statement, the SQLCODE is a positive number (for example, SQLCODE +100). If SQL encounters no error or exceptional condition while processing a statement, the SQLCODE is 0. Every SQL/400 SQLCODE has a corresponding message in message file QSQLMSG in library QSYS. For example, SQLCODE -204 is logged as message ID SQL0204.

SQLSTATE is an additional return code provided in the SQLCA. SQLSTATE provides application programs with return codes for common error conditions. SQLCODE does not return the same return code for the same error condition among the current four IBM relational database products. SQLSTATE has been designed so that application programs can test for specific error conditions or classes of errors regardless of whether the application program is connected to a DB2, SQL/DS, or AS/400 AS.

Because the SQLCA is a valuable problem-diagnosis tool, it is a good idea to include in your application programs the instructions necessary to display some of the information contained in the SQLCA. Especially important are the following SQLCA fields:

SQLCODE	Return code.
SQLERRD(3)	The number of rows updated, inserted, or deleted by SQL.
SQLSTATE	Return code.
SQLWARN0	If set to W, at least one of the SQL warning flags (SQLWARN1 through SQLWARNA) is set.

For more information about the SQLCA, see the information on SQLCA and SQLDA control blocks in the *SQL/400* Reference*.

The *SQL/400* Programmer's Guide* lists each SQLCODE, the associated message ID, the associated SQLSTATE, and the text of the message. The complete message can be viewed online by using the Display Message Description (DSPMSGD) CL command.

Distributed Relational Database SQLCODEs and SQLSTATEs

The following list provides some of the more common SQLCODEs and SQLSTATEs associated with distributed relational database processing. See the *SQL/400* Programmer's Guide* for all SQLCODEs and SQLSTATEs. In these brief descriptions of the SQLCODEs (and their associated SQLSTATEs), message data fields are identified by an ampersand (&) and a number (for example, &1). The replacement text for these fields is stored in SQLERRM in the SQLCA. More detailed cause and recovery information for any SQLCODE can be found by using the Display Message Description (DSPMSGD) CL command.

Figure 9-9 (Page 1 of 3). SQLCODEs and SQLSTATEs

SQLCODE	SQLSTATE	Description
+114	01536	Relational database name &1 not the same as current server &2.
+331	01520	Character conversion cannot be performed.
+335	01517	Character conversion has resulted in substitution characters.
+551	01548	Not authorized to object & in &2 type &3.
+552	01542	Not authorized to 1.
+595	01526	Commit level &1 has been escalated to &2 lock.
+863	01539	Only SBCS characters allowed to relational database &1.
-114	56061	Relational database &1 not the same as current server &2.
-144	58003	Section number &1 not valid. Current high section number is &3. Reason &2.
-145	55005	Recursion not supported for heterogeneous application server.
-174	54007	Maximum number of sections exceeded. Program has too many SQL statements.
-189	22522	Coded Character Set identifier &1 is not valid.
-250	52018	Local relational database not defined in the directory.
-251	37502	Character in relational database name &1 is not valid.
-302	22003 22502 22510	Conversion error on input host variable &2.
-330	22517	Character conversion cannot be performed.
-331	22518	Character conversion cannot be performed.
-332	57017	Character conversion between CCSID &1 and CCSID &2 not valid.
-334	22524	Character conversion resulted in truncation.
-525	51015	Statement is in error.
-551	42501	Not authorized to object &1 in &2 type *&3.

Figure 9-9 (Page 2 of 3). SQLCODEs and SQLSTATEs

SQLCODE	SQLSTATE	Description
-552	42502	Not authorized to &1.
-622	56031	FOR MIXED DATA clause or IGC CCSID not allowed.
-683	53042	FOR DATA clause or CCSID clause not valid for specified type.
-752	51011	Application process is not in a connectable state. Reason code &1.
-805	51002	SQL package &1 in &2 not found.
-818	51003	Consistency tokens do not match.
-862	55029	Local program attempted to connect to remote relational database.
-871	54019	Too many CCSID values specified.
-900	51018	Application process is not in a connectable state.
-950	52005	Relational database &1 not in relational directory.
-952	57014	Processing of the SQL statement was ended by ENDRDBRQS command.
-7017	56071	Commitment control is already active to a DDM target.
-7018	56070	COMMIT HOLD or ROLLBACK HOLD is not allowed.
-7021	57043	Local program attempting to run on application server.
-30000	58008	Distributed Relational Database Architecture (DRDA) protocol error.
-30001	57042	Call to distributed SQL program not allowed.
-30020	58009	Distributed Relational Database Architecture (DRDA) protocol error.
-30021	58010	Distributed relational database not supported by remote system.
-30040	57012	DDM resource &2 at relational database &1 unavailable.
-30041	57013	DDM resources at relational database &1 unavailable.
-30050	58011	DDM command &1 is not valid while bind process in progress.
-30051	58012	Bind process with specified package name and consistency token not active.
-30052	56032	Program preparation assumptions are incorrect.
-30053	42506	Not authorized to create package for owner&1.
-30060	42507	User not authorized to relational database &1.
-30061	52017	Relational database &1 not found.
-30070	58014	Distributed Data Management (DDM) command &1 not supported.
-30071	58015	Distributed Data Management (DDM) object &1 not supported.
-30072	58016	Distributed Data Management (DDM) parameter &1 not supported.

Figure 9-9 (Page 3 of 3). SQLCODEs and SQLSTATEs

SQLCODE	SQLSTATE	Description
-30073	58017	Distributed Data Management (DDM) parameter value &1 not supported.
-30074	58018	Distributed Data Management (DDM) reply message &1 not supported.
-30080	58019	Communication error occurred during distributed database processing.
-30090	56026	Change request not valid for read-only application server.

System and Communications Problems

AS/400 Problem Log

System-detected problems are automatically entered into the problem log. You can also enter a user-detected problem in the problem log. You can run problem analysis on logged problems at any time by entering the Analyze Problem (ANZPRB) command from any system command line. This command takes you through an analysis procedure and stores additional problem-related information in the problem log.

Use the Work with Problems (WRKPRB) command to view the problem log. The following displays show the two views of the problem log:

```
                          Work with Problems
                                                          System:   KC000
Position to . . . . . .                  Problem ID

Type options, press Enter.
  2=Change   4=Delete   5=Display details    6=Print details
  8=Work with problem   9=Work with alerts   12=Enter notes

Opt  Problem ID  Status         Problem Description
  __  9114350131  READY          User detected a hardware problem on a differen
  __  9114326436  OPENED         System cannot call controller . No lines avail
  __  9114326281  OPENED         Line  failed during insertion into the token-r
  __  9114324416  OPENED         Device  failed, recovery stopped.
  __  9114324241  OPENED         System cannot call controller . No lines avail
  __  9114324238  OPENED         System cannot call controller . No lines avail
  __  9114324234  OPENED         System cannot call controller . No lines avail
  __  9114324231  OPENED         System cannot call controller . No lines avail
  __  9114324227  OPENED         System cannot call controller . No lines avail
  __  9114324224  OPENED         System cannot call controller . No lines avail
  __  9114324218  OPENED         System cannot call controller . No lines avail
  __                                                                  More...
F3=Exit      F5=Refresh   F6=Print list     F11=Display dates and times
F12=Cancel   F16=Report prepared problems   F24=More keys
```

Press F11 on the first view to see the following display:

```
                          Work with Problems
                                                              System:   KC000
Position to . . . . . .                    Problem ID

Type options, press Enter.
  2=Change   4=Delete   5=Display details    6=Print details
  8=Work with problem   9=Work with alerts   12=Enter notes

Opt  Problem ID  Date      Time      Origin
 __  9114350131  03/29/92  14:36:05  APPN.KC000
 __  9114326436  03/29/92  07:41:59  APPN.KC000
 __  9114326281  03/29/92  07:39:17  APPN.KC000
 __  9114324416  03/29/92  07:06:42  APPN.KC000
 __  9114324241  03/29/92  07:03:38  APPN.KC000
 __  9114324238  03/29/92  07:03:35  APPN.KC000
 __  9114324234  03/29/92  07:03:31  APPN.KC000
 __  9114324231  03/29/92  07:03:27  APPN.KC000
 __  9114324227  03/29/92  07:03:24  APPN.KC000
 __  9114324224  03/29/92  07:03:20  APPN.KC000
 __  9114324218  03/29/92  07:03:14  APPN.KC000
                                                                      More...
F3=Exit   F5=Refresh   F6=Print list   F11=Display descriptions   F12=Cancel
F14=Analyze new problem   F16=Report prepared problems   F18=Work with alerts
```

AS/400 problem log support allows you to display a list of all the problems that have been recorded on the local system. You can also display detailed information about a specific problem such as the following:

- Product type and serial number of device with a problem
- Date and time of the problem
- Part that failed and where it is located
- Problem status

From the problem log you can also analyze a problem, report a problem, or determine any service activity that has been done. For more information about handling AS/400 problems, see the chapter on problem handling in the *Operator's Guide*.

Alerts

Alert support on the AS/400 system is based on the AS/400 message support, which is built into the operating system. Any message sent to the system operator message queue or to the history log can be defined as an alert.

For full support in handling distributed relational database problems, alerts and alert logging should be enabled using the Change Network Attributes (CHGNETA) command. OS/400 alert support is discussed in "Alert Support" on page 3-3, and procedures and examples for setting up alerts are provided in "Configuring Alert Support" on page 3-26.

Whenever an alert occurs, a system informational message (alert created) is displayed interactively or put on the job log. You can display an alert with the Work with Alerts (WRKALR) command. When you enter WRKALR, the following display appears:

```
                              Work with Alerts                      KC000
                                                          03/28/92  15:44:34
Type options, press Enter.
  4=Delete   5=Display recommended actions   6=Print details
  8=Display alert detail

     Resource
Opt    Name     Type  Date   Time   Alert Description: Probable Cause
     KC000*     UNK   05/28  15:19  Resource unavailable: Printer
     AS         SRV   05/27  21:31  Distributed process failed: Command not re
     KC000*     LU    05/23  08:29  Operator intervention required: Printer
     KC000*     UNK   05/23  08:27  Resource unavailable: Printer
     AS         SRV   05/20  11:49  Distributed process failed: Command not re
     KC000*     UNK   05/20  11:26  Resource unavailable: Printer
     AS         SRV   05/20  10:47  Distributed process failed: Relational dat
     AS         SRV   05/20  10:31  Distributed process failed: Command not re
     KC000*     CTL   05/20  09:46  Unable to communicate with remote node: Co
     KC000*     CTL   05/20  03:23  Unable to communicate with remote node: Co
     KC000*     UNK   05/19  15:32  Resource unavailable: Printer
     AS         SRV   05/19  14:37  Distributed process failed: Invalid data s
                                                                    More...
F3=Exit   F10=Show new alerts   F11=Display problem ID       F12=Cancel
F13=Change attributes     F20=Right   F21=Automatic refresh   F24=More keys
```

Alert message descriptions are contained in the QHST log. Use the Display Log (DSPLOG) command and specify QHST, or the Display Message (DSPMSG) command and specify QSYSOPR to see the alert message description.

The AS/400 system enables a subset of the DRDB messages listed in Figure 9-6 on page 9-10 to trigger alerts for distributed relational database support. If an error is detected at the AS, a DDM message is sent to the AR. The AR generates an alert based on that DDM message.

Distributed relational database alerts contain the following information:

- Identification number (alert ID)
- Type
- Description
- Probable causes
- Failure causes
- Recommended action

These alerts also contain additional information, such as:

- Product set identifier.
- Product identifier (IBM product number, version, release, modification, and product common name).
- Hierarchy Name List and Associated Resources List. These two fields show the resource name and type that detected the error condition; for example, the resource name of SQL/DS and the type of AS. If the detecting resource is not known, the identifier of the system that sent the alert is displayed as the lowest hierarchical entry.
- Local date and time from either the AS or AR.
- Other details such as relational database name and logical unit of work identifier (LUWID).

The following alerts are generated for AS/400 distributed relational database support:

Figure 9-10. Distributed Relational Database Messages that Create Alerts

Message ID	Alert ID	Text
CPF3E80	407B 879C	Syntax error detected in DDM data stream.
CPF3E81	2257 C33F	The data descriptor received is not valid.
CPF3E82	B452 CCF9	Relational database already accessed.
CPF3E83	2257 C337	Data descriptor does not match data received. FD:OCA[1] data.
CPF3E84	DA23 E856	DDM conversational protocol error was detected.
CPF3E85	B452 CCF9	Relational database (RDBNAME) not accessed.
CPF3E86	D67E 885A	Error occurred during distributed database processing.
CPF3E87	2E0A A333	Permanent error condition detected.
CPF3E88	B80F ACF5	The SQL cursor had been previously opened at the remote location.
CPF3E89	B80F ACF5	Query is not opened within this unit of work.

1 Formatted Data: Object Content Architecture (FD:OCA) used by the AS/400 system to describe the data format of the columns of a database table.

When alerts are sent from some modules that support a distributed relational database, a spooled file that contains extensive diagnostic information is also created. This data is called first-failure data capture (FFDC) information.

For more information about AS/400 alerts, see the *Alerts and DSNX Guide*.

Getting Data to Report a Failure

There are eight kinds of data that you can print to help you diagnose a problem in a distributed relational database on AS/400 systems. They are:

1. Job logs
2. Error logs
3. Trace job output
4. Communications trace output
5. First-failure data capture output
6. System operator messages
7. The application program and possibly its output
8. Data to recreate the problem

This data is produced by the OS/400 program. This section discusses the forms of output numbered 1 through 5.

Printing a Job Log

Every job on the AS/400 system has a job log that contains information related to requests entered for that job. When a user is having a problem at an AR, the information in the job log may be helpful in diagnosing the problem. One easy way to get this information is to have the user sign off with the command:

```
SIGNOFF *LIST
```

This command prints a copy of the user's job log, or places it in an output queue for printing.

Another way to print the job log is by specifying LOG(4 00 *SECLVL) on the application job description. After the job is finished, all messages are logged to the job log for that specific job. You can print the job log by locating it on an output queue and running a print procedure. See "Using the Job Log" on page 6-4 for information on how to locate jobs and job logs on the system.

The job log for the AS may also be helpful in diagnosing problems. See "Locating Distributed Relational Database Jobs" on page 6-5 for information on how to find the job name for the AS job.

Printing the Error Log

The error log on the AS/400 system is a record of machine checks, device errors, and tape and diskette statistics. It also contains FFDC information including the first 1000 bytes of each FFDC dump. By reviewing these errors you may be able to determine the nature of a problem.

To print the error log for a system on which you are signed on, do the following:

1. Type the Print Error Log (PRTERRLOG) command on any command line and press F4 (Prompt). The Print Error Log display is shown.
2. Type the parameter value for the kind of error log information you want to print and press the Enter key. The error log information is sent to the output queue identified for your job.
3. Enter the Work with Job (WRKJOB) command. The Work with Job display is shown.
4. Select the option to work with spooled files. The Work with Job Spooled Files display is shown.
5. Look for the error log file you just created at or near the bottom of the spooled file list.
6. Type the work with printing status option in the *Opt* column next to the error log file. The Work with Printing Status display is shown.
7. On the Work with Printing Status display, use the change status option to change the status of the file and specify the printer to print the file.

Trace Job

Sometimes a problem cannot be tracked to a specific program.

You can trace module flow, OS/400 data acquisition (including CL commands), or both using the Trace Job (TRCJOB) command. TRCJOB logs all of the called programs. As the trace records are generated, the records are stored in an internal trace storage area. When the trace is ended, the trace records can be written to a spooled printer file (QPSRVTRC) or directed to a database output file.

The TRCJOB command should be used when the problem analysis procedures do not supply sufficient information about the problem. For distributed database applications, the command is useful for capturing distributed database request and response data streams.

A sample trace scenario is as follows:

```
TRCJOB SET(*ON) TRCTYPE(*ALL) MAXSTG(2000)
       TRCFULL(*WRAP) EXITPGM($SCFTRC)
CALL QCMD
TRCJOB SET(*OFF) OUTPUT(*PRINT)
WRKOUTQ output-queue-name
```

You will see a spooled file with a name of QPSRVTRC. The spooled file contains your trace. For more information on the use of trace job, see Appendix C, "Interpreting Trace Job and FFDC Data" on page C-1.

Communications Trace

If you get a message in the CPF3Exx range or the CPF91xx range, you should run a communications trace. The following list shows common messages you might see in these ranges.

Figure 9-11. Communications Trace Messages

MSG ID	Description
CPF3E80	DDM data stream syntax error.
CPF91xx	DDM protocol error.
CPF3E83	Invalid FD0:CA descriptor.
CPF3E84	Data mismatch error.

The communications trace function lets you start or stop a trace of data on communications configuration objects. After you have run a trace of data, the data can be formatted for printing or viewing. You can view the printer file only in the output queue.

Communication trace options run under system service tools (SST). SST lets you use the configuration objects while communications trace is active. Data can be traced and formatted for any communications type you can use in a distributed database network.

The AS/400 communications trace can run from any display connected to the system. Anyone, with a special authority (SPCAUT) of *SERVICE can run the trace on an AS/400 system. Communications trace supports all line speeds. See the *Communications Management Guide* for the maximum aggregate line speeds on the protocols available on the 9404 System Unit and 9406 System Unit communications controllers.

Communications trace should be used in the following situations:

- The problem analysis procedures do not supply sufficient information about the problem.
- You suspect a protocol violation is the problem.
- You suspect a line noise to be the problem.
- The error messages indicate there is an Systems Network Architecture (SNA) BIND problem.

You must have detailed knowledge of the line protocols being used to correctly interpret the data generated by a communications trace. For information on interpreting DRDA data streams see "Analyzing the RW Trace Data Example" on page C-2.

Whenever possible, start the communications trace before varying on the lines. This gives you the most accurate sample of your line as it is varied on.

To run a trace and to work with its output, you have to know on what line, controller, and device you are running. If you do not have this information, refer to "Finding Your Line, Controller and Device Descriptions."

The following commands start, stop, print, and delete communications traces:

Start Communications Trace (STRCMNTRC)
Starts a communications trace for a specified line or network interface description. A communications trace continues until you run the End Communications Trace (ENDCMNTRC) command.

End Communications Trace (ENDCMNTRC)
Ends the communications trace running on the specified line or network interface description.

Print Communications Trace (PRTCMNTRC)
Moves the communications trace data for the specified line or network interface description to a spooled file or an output file. Specify *YES for the format SNA data only parameter.

Delete Communications Trace (DLTCMNTRC)
Deletes the communications trace for a specified line or network interface description.

If you are running on a Version 2, Release 1.1 or earlier system, the preceding commands are not available. Instead, you have to use the System Service Tools (SST). Start SST with the Start System Service Tools (STRSST) command. For more information about the STRSST command and details on communication traces see the *AS/400 Service: Diagnostics Aids* manuals.

Finding Your Line, Controller and Device Descriptions

Use the Work with Configuration Status (WRKCFGSTS) command to find what controller and device your application server job is being started under. For example:

```
WRKCFGSTS CFGTYPE(*DEV)
          CFGD(*LOC)
          RMTLOCNAME(DB2ESYS)
```

The value for the RMTLOCNAME keyword is the application server's system name.

The WRKCFGSTS command displays all devices that have the specified system name as the remote location name. You can tell which device is in use because you can vary on only one device at a time. Use option 8 to work with the device description and then option 5 to display it. The attached controller field gives the name of your controller. You can use the WRKCFGSTS command to work with the controller and device descriptions. For example:

```
WRKCFGSTS CFGTYPE(*CTL)
          CFGD(PCXZZ1205)     /* workstation */
WRKCFGSTS CFGTYPE(*CTL)
          CFGD(LANSLKM)       /* AS/400 on token ring */
```

The CFGD values are the controller names acquired from the device descriptions in the first example in this section.

The output from this command also includes the name of the line description that you need when working with communications traces. If you select option 8 and then option 5 to display the controller description, the active switched line parameter displays the name of the line description. The LAN remote adapter address gives the token-ring address of the remote system.

Finding First-Failure Data Capture (FFDC) Data

Note: No FFDC data is produced unless the QSFWERRLOG system value is set to *LOG.

The following are tips on how to locate FFDC data on an AS/400 system. This information is most useful if the failure causing the FFDC data output occurred on the application server (AS). The FFDC data for an application requester (AR) can usually be found in one of the spooled files associated with the job running the application program.

1. Execute a DSPMSG QSYSOPR command and look for a `Software problem detected in Qccxyyyy` message in the QSYSOPR message log. (`cc` in the program name is usually RW, but could be CN or SQ.) The presence of this message indicates that FFDC data was produced. You can use the help key to get details on the message. The message help gives you the problem ID, which you can use to identify the problem in the list presented by the WRKPRB command. You may be able to skip this step because the problem record, if it exists, may be at or near the top of the list.
2. Enter the WRKPRB command and specify the program name (Qccxyyyy) from the `Software problem detected in Qccxyyyy` message. Use the program name to filter out unwanted list items. When a list of problems is presented, specify option 5 on the line containing the problem ID to get more problem details, such as symptom string and error log ID.
3. When you have the error log ID, enter the STRSST command. On the first screen, select `Start a service tool`. On the next screen, enter 1 to select `Error log utility`. On the next screen, enter 2 to select `Display or print by error log ID`. In the next screen, you can:
   - Enter the error log ID.
   - Enter Y to get the hexadecimal display.
   - Select the Print or Display option.

   The Display option gives 16 bytes per line instead of 32. This can be useful for on-line viewing and printing screens on an 80-character workstation printer. If you choose the Display option, use F6 to see the hexadecimal data after you press Enter.

The hexadecimal data contains the first 1K bytes of the FFDC dump data, preceded by some other data. The start of the FFDC data is identified by the FFDC data index. The name of the target job (if this is on the application server) is before the data index. If the FFDC dump spool file has not been deleted, use this fully qualified job name to find the spool file. If the spool file is missing, either:

- Use the first 1K of the dump stored in the error log.
- Recreate the problem if the 1K of FFDC data is insufficient.

Interpreting FFDC Data from the Error Log

The FFDC data in the error log is not formatted for reading as well as the data in the spooled files. Each section of the FFDC dump in the error log is prefixed by a 4-byte header. The first two bytes of the header are the length of the following section (not counting the prefix). The second two bytes, which are the section number, correspond to the section number in the index (see "FFDC Dump Output Description" on page C-8).

Starting a Service Job to Diagnose Application Server Problems

When an application uses DRDA, the SQL statements are run in the application server job. Because of this, you may need to start debug or a job trace for the application server job that is running on the OS/400 operating system. When the OS/400 application server recognizes a special transaction program name (TPN), it causes the application server to send a message to the system operator and then wait for a reply (see 1). This allows you to issue a Start Service Job (STRSRVJOB) command that allows job trace or debug to be started for the application server job. The following steps allow you to stop the OS/400 application server job and restart it in debug mode.

1. Specify QCNTSRCV as the transaction program name (TPN) at the application requester. There is a different method of doing this for each platform. The following sections describe the different methods.
2. When the OS/400 application receives a TPN of QCNTSRVC, it sends a CPF9188 message to QSYSOPR and waits for a `G` (for go) reply.
3. Before entering the `G` reply, use the STRSRVJOB command to start a service job for the application server job and put it into debug mode. (Request help on the CPF9188 message to display the jobname.)
4. Enter the Start Debug (STRDBG) command.
5. After starting debug for the application server job, reply to the QSYSOPR message with a `G`.
6. After receiving the `G` reply, the application server continues with normal DRDA processing.
7. After the application runs, you can look at the application server joblog to see the SQL debug messages.

Setting QCNTSRVC as a TPN on an OS/400 Application Requester

Specify the QCNTSRVC on the TNSPGM parameter of the Add DRDB Directory Entry (ADDRDBDIRE) or Change DRDB Directory Entry (CHGRDBDIRE) commands.

Setting QCNTSRVC as a TPN on an SQL/DS Application Requester

Change the UCOMDIR NAMES file to specify QCNTSRVC in the TPN tag.

For example:

```
:nick.RCHASLAI :tpn.QCNTSRVC
               :luname.VM4GATE RCHASLAI
               :modename.MODE645
               :security.NONE
```

Then issue SET COMDIR RELOAD USER.

Setting QCNTSRVC as a TPN on a DB2 Application Requester

Update the SYSIBM.SYSLOCATIONS table to specify QCNTSRVC in the LINKATTR column for the row containing the RDB_NAME of the OS/400 application server. Use an UPDATE statement from SPUFI or ISQL. You may have stop and restart DB2 for this change to take effect.

Setting QCNTSRVC as a TPN on an OS/2 Application Requester

To stop the OS/400 application server job during an OS/2 to OS/400 DRDA session, put the following value in the Parameters field of the Database Connection Services Directory entry for the OS/400 operating system:

```
D8,CNTSRVC
```

D8 is the EBCDIC hex code for 'Q', thus the TPN will be QCNTSRVC.

Chapter 10. Writing Distributed Relational Database Applications

You can create and maintain programs for a distributed relational database on the AS/400 system using the SQL/400 language the same way you use it for local-processing applications. You can embed static and dynamic Structured Query Language (SQL) statements with any one or more of the following high-level languages:

- AS/400 PL/I
- C/400*
- COBOL/400
- FORTRAN/400*
- RPG/400

The process for developing distributed applications is similar to that of developing SQL applications for local processing. The difference is that the application for distributed processing must specify the name of the relational database to which it connects. This may be done when you precompile the program or within the application.

The same SQL objects are used for both local and distributed applications, except that one object, the SQL package, is used exclusively for distributed relational database support. You create the program using the Create SQL program (CRTSQL*xxx*) command. The *xxx* in this command refers to the host language C, CBL, FTN, PLI, or RPG. The SQL package may be a product of the precompile in this process. The Create SQL Package (CRTSQLPKG) command creates SQL packages for existing distributed SQL programs.

You must have the SQL/400 licensed program installed to precompile programs with SQL statements. However, you can create SQL packages from existing distributed SQL programs with only the compiled program installed on your system. The SQL/400 licensed program also allows you to use interactive SQL to access a distributed relational database. This is helpful when you are debugging programs because it allows you to test SQL statements without having to precompile and compile a program.

This chapter provides an overview of programming issues for a distributed relational database. More detailed information on these topics is in the *SQL/400* Programmer's Guide* and the *Distributed Relational Database Application Programming Guide* for IBM Systems Application Architecture (SAA) relational database management systems.

Programming Considerations for a Distributed Relational Database Application

Programming considerations for a distributed relational database application on an AS/400 system fall into two main categories: those that deal with a function that is supported on the local system and those that are a result of having to connect to other systems. This section addresses both of these categories as it discusses the following:

- Naming conventions

- Connecting to other systems
- Distributed SQL/400 statements and coexistence
- Coded character set identifiers (CCSIDs)
- Data translation
- Distributed Data Management (DDM) files and SQL/400 programs

Naming Distributed Relational Database Objects

SQL/400 objects are created and maintained as AS/400 system objects.

You can use either of two naming conventions in SQL/400 programming: system (*SYS) and SQL (*SQL). The naming convention you use affects the method for qualifying file and table names. It also affects security and the terms used on the interactive SQL displays. Distributed relational database applications can access objects on another AS/400 system using either naming convention. However, if your program accesses a relational database on a non-AS/400 system, only SQL names can be used. Select the naming convention using the NAMING parameter on the Start SQL (STRSQL) command or the OPTION parameter on one of the CRTSQL*xxx* commands.

System (*SYS) Naming Convention

When you use the system naming convention, files are qualified by library name in the form: *library/file*. Tables created using this naming convention assume the public authority of the library in which they are created. If the table name is not explicitly qualified and a default collection name is used in the DFTRDBCOL parameter of the CRTSQL*xxx* or CRTSLQPKG commands, the default collection name is used for static SQL statements. If the file name is not explicitly qualified and the default collection name is not specified, the following rules apply:

- All SQL/400 statements except certain CREATE statements cause SQL to search the library list (*LIBL) for the unqualified file.
- The CREATE statements resolve to unqualified objects as follows:
  - CREATE TABLE: The table name must be explicitly qualified.
  - CREATE VIEW: The view is created in the first collection referred to in the subselect. If a collection is not specified, the view is created in the first library referred to in the subselect.
  - CREATE INDEX: The index is created in the collection or library that contains the table on which the index is being built.

SQL (*SQL) Naming Convention

When you use the SQL naming convention, tables are qualified by the collection name in the form: *collection.table*. If the table name is not explicitly qualified and the default collection name is specified in the default relational database collection (DFTRDBCOL) parameter of the CRTSQLxxx or CRTSQLPKG commands, the default collection name is used. If the table name is not explicitly qualified and the default collection name is not specified, the following rules apply:

- For static SQL, the default qualifier is the user profile of the program owner.
- For dynamic SQL or interactive SQL, the default qualifier is the user profile of the job running the statement.

Default Collection Name

You can specify a default collection name to be used by an SQL program by supplying this name for the DFTRDBCOL parameter on the CRTSQL*xxx* command when you precompile the program. The DFTRDBCOL parameter provides the program with the collection name as the library for an unqualified file if the *SYS naming convention is used, or as the collection for an unqualified table if the *SQL naming convention is used. If you do not specify a default collection name when you precompile the program, the rules for unqualified names apply, as stated above, for each naming convention. The default relational database collection name only applies to static SQL statements.

You can also use the DFTRDBCOL parameter on the CRTSQLPKG command to change the default collection of a package. After an SQL program is compiled you can create a new SQL package to change the default collection. See "Create SQL Package (CRTSQLPKG) Command" on page 10-23 for a discussion of all the parameters of the CRTSQLPKG command.

Connecting to a Distributed Relational Database

What makes a distributed relational database application *distributed* is its ability to connect to a relational database on another system. For an application requester (AR) to connect to an application server (AS), the application must be in a connectable state. A **connectable state** is a condition of an application when it is possible for it to connect to a relational database or change connections. An application is in a connectable state at the following times:

- When a program starts and no changes are pending
- After an SQL CONNECT or COMMIT statement is successful
- After an SQL ROLLBACK statement, regardless of the result

An application is no longer in a connectable state after it has processed an operable SQL statement other than an SQL CONNECT, COMMIT, or ROLLBACK statement. When in a connectable state, an application can connect to an AS explicitly. Whether you connect to an AS implicitly or explicitly, by running a program or by using interactive SQL, you must have a user profile with the appropriate authority on the AS.

An application is in one of three states at any time:

- Connectable and connected
- Unconnectable and connected
- Connectable and unconnected

An AR is initially in the connectable and connected state. It is connected to the local relational database until the first SQL statement is processed, at which time it is connected to the remote relational database.

In the connectable and connected state an AR is connected to an AS and CONNECT statements can be run. The application enters this state when it completes a rollback or successful commit operation from the unconnectable and connected state. The application can also enter this state if a CONNECT statement (with a TO or RESET clause) is successfully run from the connectable and unconnected state or the connectable and connected state.

In the unconnectable and connected state an AR is connected to an AS, but a CONNECT statement cannot be successfully run to change ASs. The AR enters

this state from the connectable and connected state when it runs any SQL statement other than CONNECT, COMMIT or ROLLBACK. If the SQL application is not running under commitment control, the AR can not be in this state unless a DDM file is open with commitment control, or if a file on the local system is open under commitment control.

In the connectable and unconnected state an AR is not connected to an AS. The only SQL statement that can be run is a CONNECT statement. The AR enters this state when an SQL statement is unsuccessful because of a failure that causes a ROLLBACK operation at the AS and the loss of the connection. The application can also enter this state if a CONNECT statement (with a TO or RESET clause) is run unsuccessfully.

Implicit CONNECT

An AR automatically connects to an AS when it encounters the first SQL statement that can be run in the first active SQL program on the program stack. The following conditions must be met for this automatic connection to be made:

- The application must be in a connectable state.
- The relational database name specified by the RDB parameter of the CRTSQL*xxx* command is a remote system described in the relational database directory. If a relational database name is not specified, the application uses the local relational database.
- The SQL statement is not a CONNECT statement with a TO or RESET clause (this condition would make the connection explicit).
- A program that was compiled with CLOSQLCSR(*ENDJOB) has not been run on the current AS since the last connect.

When the implicit CONNECT is successful, the name of the AS is placed in the Current Server special register.

An application can be implicitly disconnected from a relational database when any one of the following conditions is met:

- The AR job ends normally or abnormally.
- An AR detects the first active SQL program for the process has ended and if the process is in a connectable state.
- A unit of work is either committed or rolled back and there are no SQL programs on the call stack.
- If a program compiled with CLOSQLCSR(*ENDJOB) was run, the last two conditions listed above do not implicitly disconnect the application.

The following example program is not distributed (no connection is required). It is a program run at a Spiffy Corporation regional office to gather local repair information into a report.

```
CRTSQLxxx PGM(SPIFFY/FIXTOTAL) COMMIT(*CHG) RDB(*NONE)

    PROC: FIXTOTAL;
       .
       .
       .
      SELECT * INTO :SERVICE        A
               FROM REPAIRTOT;
    EXEC SQL
    COMMIT;
       .
       .
       .
    END FIXTOTAL;
```

A Statement run on the local relational database

Another program, such as the following example, could gather the same information from Spiffy dealerships in the Kansas City region. This is an example of a distributed program that is implicitly connected and disconnected:

```
CRTSQLxxx PGM(SPIFFY/FIXES) COMMIT(*CHG) RDB(KC101)

    PROC: FIXES;
       .
       .
       .
    EXEC SQL
      SELECT * INTO :SERVICE                B
               FROM SPIFFY.REPAIR1;

    EXEC SQL   C
     COMMIT;
       .
       .
       .
    END FIXES;   D
```

B Implicit connection to AS. The statement runs on the AS.

C End of unit of work. The AR is placed in a connectable and connected state if the COMMIT is successful.

D Implicit disconnect at the end of the SQL program.

Explicit CONNECT

The CONNECT statement is used to explicitly connect an AR to an identified AS. This SQL statement can be embedded within an application program or you can issue it using interactive SQL. The CONNECT statement is used with a TO or RESET clause. A CONNECT statement with a TO clause allows you to specify connection to a particular AS relational database. The CONNECT statement with a RESET clause specifies connection to the local relational database.

When you issue (or the program issues) a CONNECT statement with a TO or RESET clause, the AS identified must be described in the relational database directory. See "Using the Relational Database Directory" on page 5-4 for more information on how to work with this directory. The AR must also be in a connectable state for the CONNECT statement to be successful.

When a CONNECT statement with a TO or RESET clause is successful, the following occurs:

- Any open cursors are closed, any prepared statements are discarded, and any held resources are released from the previous AS if the application process was placed in the connectable state through the use of COMMIT HOLD or ROLLBACK HOLD SQL statements, or if the application process is running COMMIT(*NONE).
- The application process is disconnected from its previous AS, if any, and connected to the identified AS.
- The name of the AS is placed in the Current Server special register.
- Information that identifies the type of AS is placed in the SQLERRP field of the SQL communication area (SQLCA).

If the CONNECT statement with a TO or RESET clause is unsuccessful because the AR is not in the connectable state or the *server-name* is not listed in the local relational database directory, the connection state of the AR is unchanged. If the CONNECT statement is unsuccessful for any other reason, the application remains in the connectable but unconnected state. An application in the connectable but unconnected state can only run the CONNECT statement.

It is a good practice for the first SQL statement run by an application process to be the CONNECT statement. However, when you have CONNECT statements embedded in your program you may want to dynamically change the AS name if the program connects to multiple ASs. If you are going to run the application at multiple systems, you can specify the CONNECT statement with a host variable as shown below, so that the program can be passed the relational database name.

```
CONNECT TO  : host-variable
```

Without CONNECT statements, all you need to do when you change the AS is to recompile the program with the new relational database name.

Consecutive CONNECT statements can be run successfully because CONNECT does not remove the AR from the connectable state. A CONNECT to the AS to which the AR is currently connected is run like any other CONNECT statement.

If running with commitment control, the CONNECT statement cannot run successfully when it is preceded by any SQL statement other than CONNECT, COMMIT, or ROLLBACK. To avoid an error, perform a COMMIT or ROLLBACK operation before a CONNECT statement is run. If running without commitment control, the CONNECT statement is always allowed.

The following example shows two forms of the CONNECT statement (**1** and **2**) in an application program:

```
CRTSQLxxx PGM(SPIFFY/FIXTOTAL) COMMIT(*CHG) RDB(KC105)

  PROC: FIXTOTAL;
  EXEC SQL   CONNECT TO KC105; 1
  :
  EXEC SQL
    SELECT * INTO :SERVICE
             FROM REPAIRTOT;
  :
  EXEC SQL COMMIT;
  :
  EXEC SQL CONNECT TO MPLS03 USER :USERID USING :PW;   2
  :
  EXEC SQL SELECT ...
  :
  EXEC SQL COMMIT;
  :
  END FIXTOTAL;
```

The example (2) shows the CONNECT statement extension in the OS/400 operating system (Version 2 Release 2 and later). This extension provides a way that applications can use the SECURITY=PGM form of SNA LU 6.2 conversation security. The user ID and password stored in the host USERID and PW variables are transmitted in the ALLOCATE verb SECURITY parameter. In this example the ALLOCATE verb flows when the connection is established to MPLS03. You must specify the user ID and password with host variables when the CONNECT statement is embedded in a program.

The following example shows both CONNECT statement forms in interactive SQL. Note that the password must be enclosed in single quotes.

```
Type SQL statement, press Enter.
     Current connection is to relational database (RDB) KC105.
     CONNECT TO KC000_____________________________________________________
  :
     COMMIT_______________________________________________________________
===> CONNECT TO MPLS03 USER JOE USING 'X47K'_______________________________
     _____________________________________________________________________
     _____________________________________________________________________
```

When using the Start SQL Interactive Session (STRSQL) command, the Start SQL/400 Query Manager (STRQM) command, and the Start Query Management Query (STRQMQRY) command to a remote relational database, an SQL package is needed to run SQL statements. Normally, the system creates this package. However, if the system fails to create the package, or if you are not authorized to the package, you may be left in a connected and unconnectable state. To return to a connectable state, you must enter a control language (CL) COMMIT or ROLLBACK command. You cannot enter an SQL COMMIT or ROLLBACK statement because these statements require the package to perform their processing.

SQL Specific to Distributed Relational Database

During the precompile process of a distributed SQL/400 application, the OS/400 program may build SQL packages to be run on an AS. After it is compiled, a distributed SQL program and package must be compatible with the systems that are being used as ARs and ASs. For example, you cannot compile a distributed SQL program on an AS/400 system at the Version 2 Release 1 Modification 1 level and run that program on another AS/400 system at the Version 2 Release 1 level. "Preparing Distributed Relational Database Programs" on page 10-14 gives you more information about the changes to the precompile process and the addition of SQL packages.

This section gives an overview of the SQL/400 statements that are used with distributed relational database support and some things for you to consider about coexistence with other systems. For more detail on these subjects, see the *SQL/400* Reference* and the *SQL/400* Programmer's Guide*.

Distributed Relational Database Statements

The following statements included with the SQL/400 language specifically support a distributed relational database:

- CONNECT
- DROP PACKAGE
- GRANT EXECUTE ON PACKAGE
- REVOKE EXECUTE ON PACKAGE

"Explicit CONNECT" on page 10-5 describes using the CONNECT statement to connect an AR to an AS. Using the SQL GRANT EXECUTE ON PACKAGE and REVOKE EXECUTE ON PACKAGE statements to grant or revoke user authority to SQL packages is described in "Authority to Distributed Relational Database Objects" on page 4-14. The SQL DROP PACKAGE statement, as it is used to drop an SQL package, is discussed in "Working With SQL Packages" on page 10-22.

SQL/400 Coexistence

When you write and maintain programs for a distributed relational database using the SQL/400 language, you need to consider the other systems in the distributed relational database network. The program you are writing or maintaining may have to be compatible with the following:

- Other AS/400 systems
- Previous AS/400 releases
- Systems that are not AS/400 systems

Remember that the SQL statements in a distributed SQL program run on the AS. Even though the program runs on the AR, the SQL statements are in the SQL package to be run on the AS. Those statements must be supported by the AS and be compatible with the collections, tables, and views that exist on the AS. Also, the users who run the program on the AR must be authorized to the SQL package and other SQL objects on the AS.

Releases of the AS/400 system before Version 2 Release 1 Modification 1 do not support distributed relational database. If you are writing applications for an earlier version of the AS/400 system, you need to use DDM support that is at the level required for the target system. You can convert an SQL program from a previous release to a distributed SQL program by creating the program again using the

CRTSQL*xxx* command and specifying the relational database name (RDB parameter) for an AS. This compiles the program again using the distributed relational database support in the OS/400 program and the SQL/400 licensed program, and creates the SQL package needed on the AS.

You can write SQL/400 programs that run on ASs that are not AS/400 systems and these other platforms may support more or less SQL functions. Statements that are not supported on the AS/400 AR can be used and compiled on the AS/400 system when the AS supports the function. SQL programs written to run on an AS/400 AS only provide the level of support described in this guide. See the support documentation for the other systems to determine the level of function they provide.

Ending Units of Work

You should be careful about ending RUOW programs with an uncommitted unit of work. A program having uncommitted requests in a unit of work is in an *unconnectable state*. When a program ends in an unconnectable state, the conversation with the AS remains active, and the AS has no knowledge that the program ended.

This behavior differs from that of other systems because in the OS/400 operating system, COMMITs and ROLLBACKs can be used as commands from the command line or in a CL program. However, the preceding scenario can lead to unexpected results in the next SQL program run, unless you plan for the situation. For example, if you run interactive SQL next (STRSQL command), the interactive session starts up in the state of being connected to the previous AS with uncommitted work. As another example, if following the preceding scenario, you start a second RUOW program that does an implicit connect, an attempt is made to find and run a package for it on the AS that was last used. This may not be the AS that you intended. To avoid these surprises always commit or rollback the last unit of work before ending any application program.

Coded Character Set Identifier (CCSID)

Support for the national language of any country requires the proper handling of a minimum set of characters. A cross-system support for the management of character information is provided with the Systems Application Architecture (SAA) Character Data Representation Architecture (CDRA). CDRA defines the coded character set identifier (CCSID) values to identify the code points used to represent characters, and to convert these codes (character data), as needed to preserve their meanings.

The use of an architecture such as CDRA and associated conversion protocols is important in the following situations:

- More than one national language version is installed on the AS/400 system.
- Multiple AS/400 systems are sharing data between systems in different countries with different primary national language versions.
- AS/400 systems and non-AS/400 systems are sharing data between systems in different countries with different primary national language versions.

Tagging is the primary means to assign meaning to coded graphic characters. The tag may be in a data structure that is associated with the data object (explicit

tagging), or it may be inherited from objects such as the job or the system itself (implicit tagging).

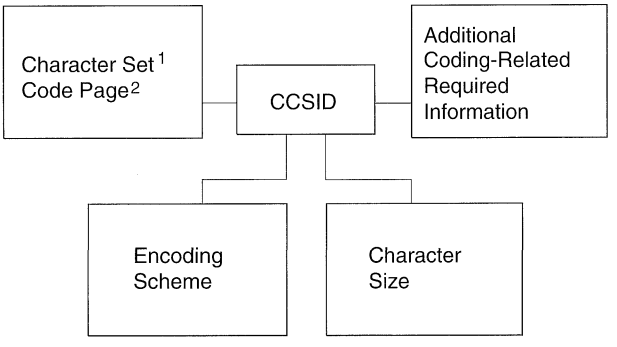
The OS/400 database manager tags character columns with CCSIDs. A **CCSID** is a 16-bit number identifying a specific set of encoding scheme identifiers, character set identifiers, code page identifiers, and additional coding-related information that uniquely identifies the coded graphic character representation used. When running applications, data is not converted when it is sent to another system; it is sent as tagged along with its CCSID. The receiving job automatically converts the data to its own CCSID if it is different from the way the data is tagged.

The CDRA has defined the following range of values for CCSIDs.

00000	Use next hierarchical CCSID
00001 through 28671	IBM-registered CCSIDs
28672 through 65533	Reserved
65534	Refer to lower hierarchical CCSID
65535	No conversion done

See the *National Language Support Planning Guide* for a list of the OS/400 CCSIDs and the *Character Data Representation Architecture - Level 1, Registry* for a complete list of the CDRA CCSIDs. For more information on handling CCSIDs, see the *SQL/400* Reference* and the *SQL/400* Programmer's Guide*.

The following illustration shows the parts of a CCSID.



1 Can be more than one character set.
2 Can be more than one code page.

RV2W743-1

Figure 10-1. Coded Character Set Identifier (CCSID)

AS/400 Support

The default CCSID for a job on the AS/400 system is specified using the Change Job (CHGJOB) command. If a CCSID is not specified in this way, the job CCSID is obtained from the CCSID attribute of the user profile. If a CCSID is not specified on the user profile, the system gets it from the QCCSID system value. This QCCSID value is initially set to 65535. If your AS/400 system is in a distributed relational database with unlike systems, it may not be able to use CCSID 65535. See Appendix B, "Cross-Platform Distributed Relational Database Notes" on page B-1 for things to consider when operating in an unlike environment.

All control information that flows between the AR and AS is in CCSID 500 (a DRDA standard). This is information such as collection names, table names, and some

descriptive text. Using variant characters for control information causes these names to be converted, which can affect performance. Package names are also sent in CCSID 500. Using variant characters in a package name causes the package name to be converted. This means the package is not found at run time.

After a job has been initiated, you can change the job CCSID by using the CHGJOB command. To do this:

1. Enter the Work with Job (WRKJOB) command to get the Work with Jobs display.
2. Select option 2 (Display job definition attributes).

   This locates the current CCSID value so you can reset the job to its original CCSID value later.
3. Enter the CHGJOB command with the new CCSID value.

The new CCSID value is reflected in the job immediately. However, if the job CCSID you change is an AR job, the new CCSID does not affect the work being done until the next CONNECT.

Warning: If you change the CCSID of an AS job, the results cannot be predicted.

Source files are tagged with the job CCSID if a CCSID is not explicitly specified on the Create Source Physical File (CRTSRCPF) or Create Physical File (CRTPF) command for source files. Externally described database files and tables are tagged with the job CCSID if a CCSID is not explicitly specified in data description specification (DDS), in interactive data definition utility (IDDU), or in the CREATE TABLE SQL statement. Program described files are tagged with CCSID 65535. Views are tagged with the CCSID of its corresponding table tag or column-level tags. If a view is defined over several tables, it is tagged at the column level and assumes the tags of the underlying columns. Views cannot be explicitly tagged with a CCSID. The system automatically converts data between the job and the table if the CCSIDs are not equal and neither of the CCSIDs is equal to 65535. (No conversion is ever done on data tagged with CCSID 65535.)

When you change the CCSID of a tagged table, it cannot be tagged at the column level or have views defined on it. To change the CCSID of a tagged table, use the Change Physical File (CHGPF) command. To change a table with column-level tagging, you must create it again and copy the data to a new table using FMT(*MAP) on the Copy File (CPYF) command. When a table has one or more views defined, you must do the following to change the table:

1. Save the view and table along with their access paths.
2. Delete the views.
3. Change the table.
4. Restore the views and their access paths over the created table.

When you migrate files to the AS/400 system, they are tagged with CCSID 65535. This includes files that are on the system when you install a new release or those that are not tagged and are restored onto the system.

All data that is sent between an AR and an AS is sent not converted. In addition, the CCSID is also sent. The receiving job automatically converts the data to its own CCSID if it is different from the way the data is tagged. For example, consider the following application that is run on a dealership system, KC105.

```
CRTSQLxxx PGM(PARTS1) COMMIT(*CHG)  RDB(KC000)

    PROC: PARTS1;
       .
       .
    EXEC SQL
      SELECT * INTO :PARTAVAIL
               FROM INVENTORY
               WHERE ITEM = :PARTNO;
       .
       .
    END PARTS1;
```

In the above example, the local system (KC105) has the QCCSID system value set at CCSID 37. The remote regional center (KC000) uses CCSID 937 and all its tables are tagged with CCSID 937. CCSID processing takes place as follows:

- The KC105 system sends an input host variable (:PARTNO) in CCSID 37. (The DECLARE VARIABLE SQL statement can be used if the CCSID of the job is not appropriate for the host variable.)
- The KC000 system converts :PARTNO to CCSID 937, selects the required data, and sends the data back to KC105 in CCSID 937.
- When KC105 gets the data, it converts it to CCSID 37 and places it in :PARTAVAIL for local use.

Other Data Conversion

Sometimes, when you are doing processing on a remote system, your program may need to convert the data from one system so that it can be used on the other. DRDA support on the AS/400 system converts the data automatically between other systems that use DRDA support. When an AS/400 AR connects to an AS, it sends information that identifies its type. Likewise, the AS sends back information to the AS/400 system that identifies its processor type (for example, S/390* host or AS/400 system). The two systems then automatically convert the data between them as defined for this connection. This means that you do not need to program for architectural differences between systems.

Data conversion between IBM systems with DRDA support includes data types such as:

- Floating point representations
- Zoned decimal representations
- Byte reversal
- Mixed data types
- AS/400 specific data types such as:
  - DBCS-only
  - DBCS-either
  - Integer with precision and scale

DDM Files and SQL

You can use AS/400 DDM support to help you do some distributed relational database tasks within a program that also uses SQL distributed relational database support. It may be faster, for example, for you to use DDM and the Copy File (CPYF) command to get a large number of records rather than an SQL FETCH statement. Also, DDM can be used to get external file descriptions of the remote system data brought in during compile for use with the distributed relational database application. To do this you need to use DDM as described in Chapter 3, "Communications for an AS/400 Distributed Relational Database" and set up a DDM file as discussed in Chapter 5, "Setting Up an AS/400 Distributed Relational Database."

Commitment control operations performed for distributed relational database can also apply to DDM file operations. All resources you access under commitment control must be accessed from the same job. This means if you want to use commitment control for both your distributed relational database and DDM file operations, the DDM file you access must use the same connection parameters (remote location, network identifier, mode, and so on) as the relational database directory uses to access the AS.

If commitment control is already active for a distributed relational database application when you try to open a DDM file under commitment control with connection parameters that do not match those in the relational database directory, the DDM file open fails. Likewise, if DDM files are open under commitment control when you try to connect to a distributed relational database with connection parameters that do not match those in the DDM file, the CONNECT fails.

The following example shows how you can add a relational database directory entry and create a DDM file so that the same job can be used on the AS and target system.

Relational Database Directory:

```
ADDRDBDIRE    RDB(KC000) +
              RMTLOCNAME(KC000)
              TEXT('Kansas City regional database')
```

DDM File:

```
CRTDDMF  FILE(SPIFFY/UPDATE)
         RMTFILE(SPIFFY/INVENTORY)
         RMTLOCNAME(KC000)
         TEXT('DDM file to update local orders')
```

The following is a sample program that uses both the relational database directory entry and the DDM file in the same job on the remote system:

```
CRTSQLxxx PGM(PARTS1) COMMIT(*CHG)  RDB(KC000)

    PROC :PARTS1;
    OPEN  SPIFFY/UPDATE;
       .
       .
       .
    CLOSE SPIFFY/UPDATE;
       .
       .
       .
    EXEC SQL
      SELECT * INTO :PARTAVAIL
               FROM INVENTORY
               WHERE ITEM = :PARTNO;
    EXEC SQL
      COMMIT;
       .
       .
       .
    END PARTS1;
```

See the *DDM Guide* for more information on how to use AS/400 DDM support.

Preparing Distributed Relational Database Programs

When you write a program using the SQL/400 language, you normally embed the SQL statements in a host program. The **host program** is the program that contains the SQL statements, written in one of the **host languages**: the AS/400 PL/I, C/400, COBOL/400, FORTRAN/400, or RPG/400 programming language. In a host program you use variables referred to as **host variables**. These are variables used in SQL statements that are identifiable to the host program. In RPG, this is called a field name; in FORTRAN, PL/I, and C, this is known as a variable; in COBOL, this is called a data item.

You can code your distributed SQL/400 programs in a way similar to the coding for an SQL/400 program that is not distributed. You use the host language to embed the SQL statements with the host variables. Also, like an SQL/400 program that is not distributed, a distributed SQL/400 program is prepared using the following processes:

- Precompiling
- Compiling
- Binding the application
- Testing and debugging

However, a distributed SQL/400 program also requires that an SQL package is created on the AS to access data.

This section discusses these steps in the process, outlining the differences for a distributed SQL/400 program.

Precompiling Programs with SQL Statements

You must precompile and compile an application program containing embedded SQL statements before you can run it. Precompiling such programs is done by an SQL precompiler. The SQL precompiler scans each statement of the application program source and does the following:

- Looks for SQL statements and for the definition of host variable names
- Verifies that each SQL statement is valid and free of syntax errors
- Validates the SQL statements using the description in the database
- Prepares each SQL statement for compilation in the host language
- Produces information about each precompiled SQL statement

Application programming statements and embedded SQL statements are the primary input to the SQL precompiler. The SQL precompiler assumes that the host language statements are syntactically correct. If the host language statements are not syntactically correct, the precompiler may not correctly identify SQL statements and host variable declarations.

The SQL precompile process produces a listing and a temporary source file member. It can also produce the SQL package depending on what is specified for the OPTION and RDB parameters of the precompiler command. See "Compiling an Application Program" on page 10-19 for more information about this parameter.

Listing

The output listing is sent to the printer file specified by the PRTFILE parameter of the CRTSQL*xxx* command. The following items are written to the printer file:

- Precompiler options

  This is a list of all the options specified with the CRTSQL*xxx* command and the date the source member was last changed.

- Precompiler source

  This output is produced if the *SOURCE option is used. It shows each precompiler source statement with its record number assigned by the precompiler, the sequence number (SEQNBR) you see when using the source entry utility (SEU), and the date the record was last changed.

- Precompiler cross-reference

  This output is produced if *XREF was specified in the OPTION parameter. It shows the name of the host variable or SQL entity (such as tables and columns), the record number where the name is defined, what the name is defined, and the record numbers where the name occurs.

- Precompiler diagnostic list

  This output supplies diagnostic messages, showing the precompiler record numbers of statements in error.

Temporary Source File Member

Source statements processed by the precompiler are written to QSQLTEMP in the QTEMP library. In your precompiler-changed source code, SQL statements have been converted to comments and calls to the SQL interface module, QSQROUTE. The name of the temporary source file member is the same as the name specified in the PGM parameter of CRTSQL*xxx*. This member cannot be changed before being used as input to the compiler.

QSQLTEMP can be moved to a permanent library after the precompile, if you want to compile at a later time. If you change the records of the temporary source file member, the compile attempted later will fail.

SQL Package Creation

An object called an SQL package can be created as part of the precompile process when the CRTSQLxxx command is compiled. See "Compiling an Application Program" on page 10-19 and "Binding an Application" on page 10-19 for information on situations that affect package creation as part of these processes. See "Working With SQL Packages" on page 10-22 for more information on the SQL package and commands that you can use to work with a package.

Precompiler Commands

The SQL/400 program has six precompiler commands, one for each of the host languages.

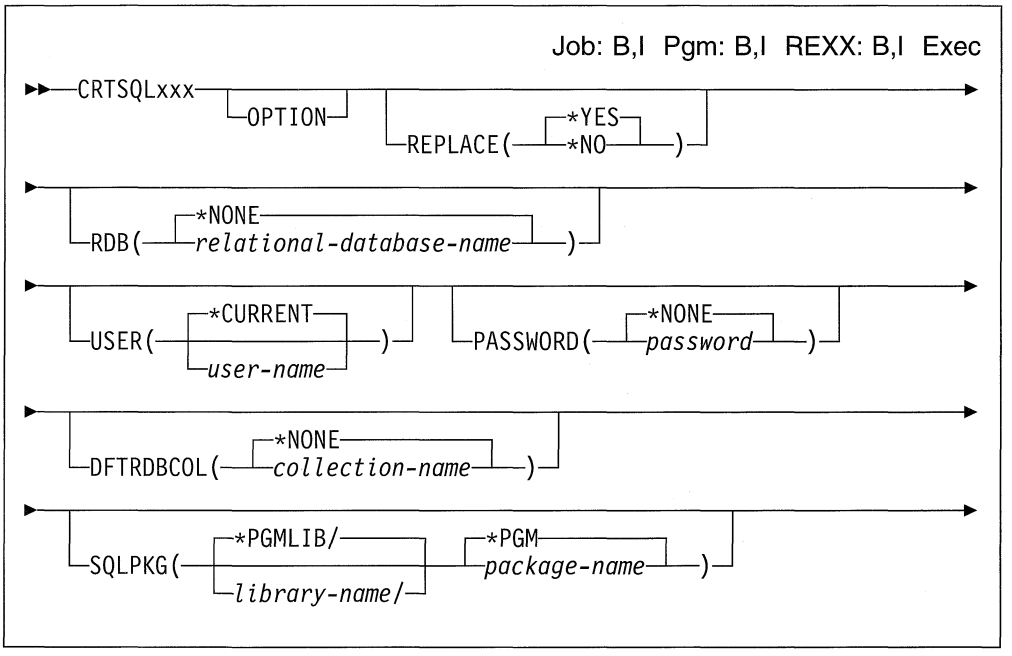
Host Language	Command
AS/400 PL/I	CRTSQLPLI
C/400 language	CRTSQLC
ILE C/400 language	CRTSQLCI
COBOL/400 language	CRTSQLCBL
FORTRAN/400 language	CRTSQLFTN
RPG III (part of RPG/400 language)	CRTSQLRPG

A separate command for each language exists so each language can have parameters that apply only to that language. For example, the options *APOST and *QUOTE are unique to COBOL. They are not included in the commands for the other languages.

The precompiler is controlled by parameters specified when it is called by one of the SQL precompiler commands. The parameters specify how the input is processed and how the output is presented.

You can precompile a program without specifying anything more than the name of the member containing the program source statements as the PGM parameter of the CRTSQL*xxx* command. SQL assigns default values for all precompiler parameters (which may, however, be overridden by any that you explicitly specify).

The following shows the syntax for parameters common to all the CRTSQL*xxx* commands for distributed relational database support only. To see the syntax and explanation for all the parameters of the CRTSQL*xxx* commands, see the *SQL/400* Programmer's Guide*.



OPTION

Specifies options for creating the program including language-specific parameters. See the *SQL/400* Programmer's Guide* for a complete description of these options for each of the host languages. Two options on this parameter specify the naming convention used:

***SYS:** Specifies that the AS/400 system naming convention will be used, (library-name/file-name).

***SQL:** Specifies that the SQL naming convention will be used (collection-name.table-name).

Note: *SYS also specifies that the AS/400 security convention will be used. *SQL specifies that the SQL security convention will be used. The AS/400 security convention requires the user running the program have authority to objects referred to by the SQL statements in the program. The SQL security convention requires that the owner of the program have authority to the objects referred to in the static SQL statements in the program. The SQL security convention is the same as the AS/400 security convention for dynamic SQL and interactive SQL.

REPLACE

Specifies if an SQL program or package is created when there is an existing SQL program or SQL package of the same name in the same library. The value of this parameter is passed to the CRT*xxx*PGM command.

***YES**: An SQL program or package is created, and any existing program or package of the same name in the specified library is moved to QRPLOBJ. The *YES value is passed to the command that creates the SQL package.

***NO**: An SQL program or package is not created if an SQL program or package of the same name already exists in the specified library.

RDB

Specifies the name of a relational database where the SQL package object is to be created.

***NONE**: An SQL package is not created. The program object is created, but it is not a distributed program. The CRTSQLPKG command cannot be used on the program created.

relational-database-name: Specifies the name of the relational database where the new SQL package object is to be created. If you specify the name of the local relational database, the program created is still a distributed SQL program. The SQL statements access the local database.

USER

Specifies the user name sent to the remote system when starting the conversation.

***CURRENT:** The user name associated with the current job is used.

user-name: Specifies the user name to be used for the remote job.

PASSWORD

Specifies the password to be used on the remote system. Even if the password and user ID were specified on the precompile command, they must be entered here again.

Once a connection to a distributed relational database is established for a user ID, the password, if specified, may not need to be validated on subsequent connections to the same relational database with the same user ID. The validation of the password depends on if the conversation is still active. For more details, see "Controlling DDM Conversations" on page 6-15.

***NONE:** No password is sent. If a user name is specified on the USER parameter, the value is not valid.

password: Specifies the password of the user name specified on the USER parameter.

DFTRDBCOL

Specifies the name of the collection identifier to be used for the unqualified names of tables, views, indexes, and SQL packages. This parameter applies only to static SQL statements.

***NONE:** The naming convention specified on the OPTION parameter is used.

collection: Specifies the name of the collection identifier to be used instead of the naming convention specified on the OPTION parameter.

SQLPKG

Specifies the name and library of the SQL package to be created on the remote relational database specified on the RDB parameter.

***PGMLIB:** Specifies that the SQL package is to be put in a library that has the same name as the library containing the program.

library-name: Specifies the name of the library where the SQL package is to be placed. If the remote system is not an AS/400 system, a maximum of 8 characters can be specified.

***PGM:** Specifies that the package name is the same as the program name.

package-name: Specifies the name of the SQL package created. If the remote system is not an AS/400 system, a maximum of 8 characters can be specified.

The following example creates a COBOL program named INVENT and stores it in a library named SPIFFY. The SQL naming convention is selected, and every row

selected from a specified table is locked until the end of the unit of recovery. An SQL package with the same name as the program is created on the remote relational database named KC000.

```
CRTSQLCBL PGM(SPIFFY/INVENT) OPTION(*SRC *XREF *SQL)
          COMMIT(*ALL) RDB(KC000)
```

Compiling an Application Program

The SQL/400 program automatically calls the host language compiler after the successful completion of a precompile, unless the *NOGEN precompiler option is specified. The CRT*xxx*PGM command is run specifying the program name, source file name, precompiler created source member name, text, and user profile. Other parameters are also passed to the compiler, depending on the host language. For more information on these parameters, see the *SQL/400* Programmer's Guide*.

You can interrupt the call to the host language compiler by specifying *NOGEN under the OPTION parameter of the precompiler command. The precompiler creates the source member in the QTEMP/QSQLTEMP file, using the program name specified in the CRTSQL*xxx* command as the name of the member. Specify this file and member on the compile command.

If you separate the precompile and compile steps, there are precautions that you need to consider:

- You must not change the source member in QTEMP/QSQLTEMP before issuing the CRT*xxx*PGM command or the compile fails.
- If the source program refers to externally described files, the files referred to must not be changed between the precompile and compile steps. Otherwise, unpredictable results may occur because changes to the field definitions are not reflected in the temporary source member. Externally described files use COPY DDS in COBOL, %INCLUDE in PL/I, #pragma mapic and #include in C, and externally defined files or data structures in RPG.

  Also, any externally described file must be on the AR.
- On separate precompile and compile steps the Create SQL Package (CRTSQLPKG) command is used to create the SQL package for the distributed program.

Binding an Application

Before you can run your application program, a relationship between the program and any referred-to tables and views must be established. This process is called **binding**. The result of binding is an access plan. The **access plan** is a control structure that describes the actions necessary to satisfy each SQL request. An access plan contains information about the program and about the data the program intends to use. For distributed relational database work, the access plan is stored in the SQL package and managed by the system along with the SQL package. See "Working With SQL Packages" on page 10-22 for more information about SQL packages.

SQL automatically attempts to bind and create access plans when a successful compile has occurred. If, at run time, the database manager detects that an access plan is not valid or that changes have occurred to the database that may

improve performance (for example, the addition of indexes), a new access plan is automatically created. If the AS is not an AS/400 system, then a bind must be done again using the CRTSQLPKG command. Binding does three things:

- Revalidates the SQL statements using the description in the database.

  During the bind process, the SQL statements are checked for valid table, view, and column names. If a referred to table or view does not exist at the time of the precompile or compile, the validation is done at run time. If the table or view does not exist at run time, a negative SQLCODE is returned.

- Selects the access paths needed to access the data your program wants to process.

  In selecting an access path, indexes, table sizes, and other factors are considered when SQL builds an access plan. The bind process considers all indexes available to access the data and decides which ones (if any) to use when selecting a path to the data.

- Attempts to build access plans.

  If all the SQL statements are valid, the bind process builds and stores access plans in the program.

If the characteristics of a table or view your program accesses have changed, the access plan may no longer be valid. When you attempt to use an access plan that is not valid, the system automatically attempts to rebuild the access plan. If the access plan cannot be rebuilt, a negative SQLCODE is returned. In this case, you might have to change the program's SQL statements and reissue the CRTSQL*xxx* command to correct the situation.

For example, if a program contains an SQL statement that refers to COLUMNA in TABLEA and the user deletes and recreates TABLEA so that COLUMNA no longer exists, when you call the program, the automatic rebind is unsuccessful because COLUMNA no longer exists. You must change the program source and reissue the CRTSQL*xxx* command.

Testing and Debugging

Testing and debugging distributed SQL programs is similar to to testing and debugging local SQL programs, but certain aspects of the process are different.

More than one system will eventually be required for testing. If applications are coded so that the relational database names can easily be changed by recompiling the program, changing the input parameters to the program, or making minor modifications to the program source, most testing can be accomplished using a single system.

After the program has been tested against local data, the program is then made available for final testing on the distributed relational database network. Consider testing the application locally on the AS so that only the program will need to be moved when the testing moves into a distributed environment.

Debugging a distributed SQL program uses the same techniques as debugging a local SQL program. You use the Start Debug (STRDBG) command to start the debugger and to put the application in debug mode. You can add breakpoints, trace statements, and display the contents of variables.

However, to debug a distributed SQL program, you must specify the value of *YES for the UPDPROD parameter. This is because OS/400 distributed relational database support uses files in library QSYS and QSYS is a production library. This allows data in production libraries to be changed on the AR. Issuing the STRDBG command on the AR only puts the AR job into debug mode, so your ability to manipulate data on the AS is not changed.

While in debug mode on the AR, informational messages are entered in the job log for each SQL statement run. These messages give information about the result of each SQL statement. A list of SQL return codes and a list of error messages for distributed relational database are provided in Chapter 9, "Handling Distributed Relational Database Problems."

Informational messages about how the system maximizes processing efficiency of SQL statements also are issued as a result of being in debug mode. Since any maximization occurs at the AS, these types of messages will not appear in the AR job log. To get this information, the AS job must be put in to debug mode.

If both the AR and AS are AS/400 systems, you can use the Submit Remote Command (SBMRMTCMD) command to start the debug mode in an AS job. Create a DDM file as described in "Setting Up DDM Files" on page 5-10. The communications information in the DDM file must match the information in the relational database directory entry for the relational database being accessed. Then issue the command:

```
SBMRMTCMD CMD('STRDBG UPDPROD(*YES)') DDMFILE(ddmfile name)
```

The SBMRMTCMD command starts the AS job if it does not already exist and starts the debug mode in that job. Use the methods described in "Monitoring Relational Database Activity" on page 6-1 to examine the AS job log to find the job.

The following method for putting the AS job into debug mode works with any AR and an AS/400 AS.

- Sign on to the AS and find the AS job.
- Issue the Start Service Job (STRSRVJOB) command from the your interactive job (the job you are using to find the AS job) as shown:

  ```
  STRSRVJOB (job-number/user-ID/job-name)
  ```

  The job name for the STRSRVJOB command is the name of the AS job. Issuing this command lets you issue certain commands from your interactive job that affect the AS job. One of these commands is the Start Debug (STRDBG) command.
- Issue the STRDBG command using a value of *YES for the UPDPROD parameter in the interactive job. This puts the AS job into debug mode to produce debug messages on the AS job log.

To end this debug session, either end your interactive job by signing off or use the End Debug (ENDDBG) command followed by the End Service Job (ENDSRVJOB) command.

Since the AS job must be put into debug before the SQL statements are run, the application may need to be changed to allow you time to set up debug on the AS. The AS job starts as a result of the application connecting to the AS. Your applica-

tion could be coded to enter a wait state after connecting to the AS until debug is started on the AS.

Program References

When a program is created, the OS/400 licensed program stores information about all collections, tables, views, SQL packages, and indexes referred to in SQL statements in an SQL program.

You can use the Display Program References (DSPPGMREF) command to display all object references in the program. If the SQL naming convention is used, the library name is stored in one of three ways:

- If the SQL name is fully qualified, the collection name is stored as the name qualifier.
- If the SQL name is not fully qualified, and the DFTRDBCOL parameter is not specified, the authorization ID of the statement is stored as the name qualifier.
- If the SQL name is not fully qualified, and the DFTRDBCOL parameter is specified, the collection name specified on the DFTRDBCOL parameter is stored as the name qualifier.

If the system naming convention is used, the library name is stored in one of three ways:

- If the object name is fully qualified, the library name is stored as the name qualifier.
- If the object is not fully qualified, and the DFTRDBCOL parameter is not specified, *LIBL is stored.
- If the SQL name is not fully qualified, and the DFTRDBCOL parameter is specified, the collection name specified on the DFTRDBCOL parameter is stored as the name qualifier.

Working With SQL Packages

An SQL package is an SQL object used specifically by distributed relational database applications. It contains control structures for each SQL statement that accesses data on an AS. These control structures are used by the AS at run time when the application program requests data using the SQL statement.

You must use a control language (CL) command to create an SQL package because there is no SQL statement for SQL package creation. You can create an SQL package in two ways:

- Using the CRTSQL*xxx* command with a relational database name specified in the RDB parameter. See "Precompiling Programs with SQL Statements" on page 10-15 for more information on this command.
- Using the CRTSQLPKG command.

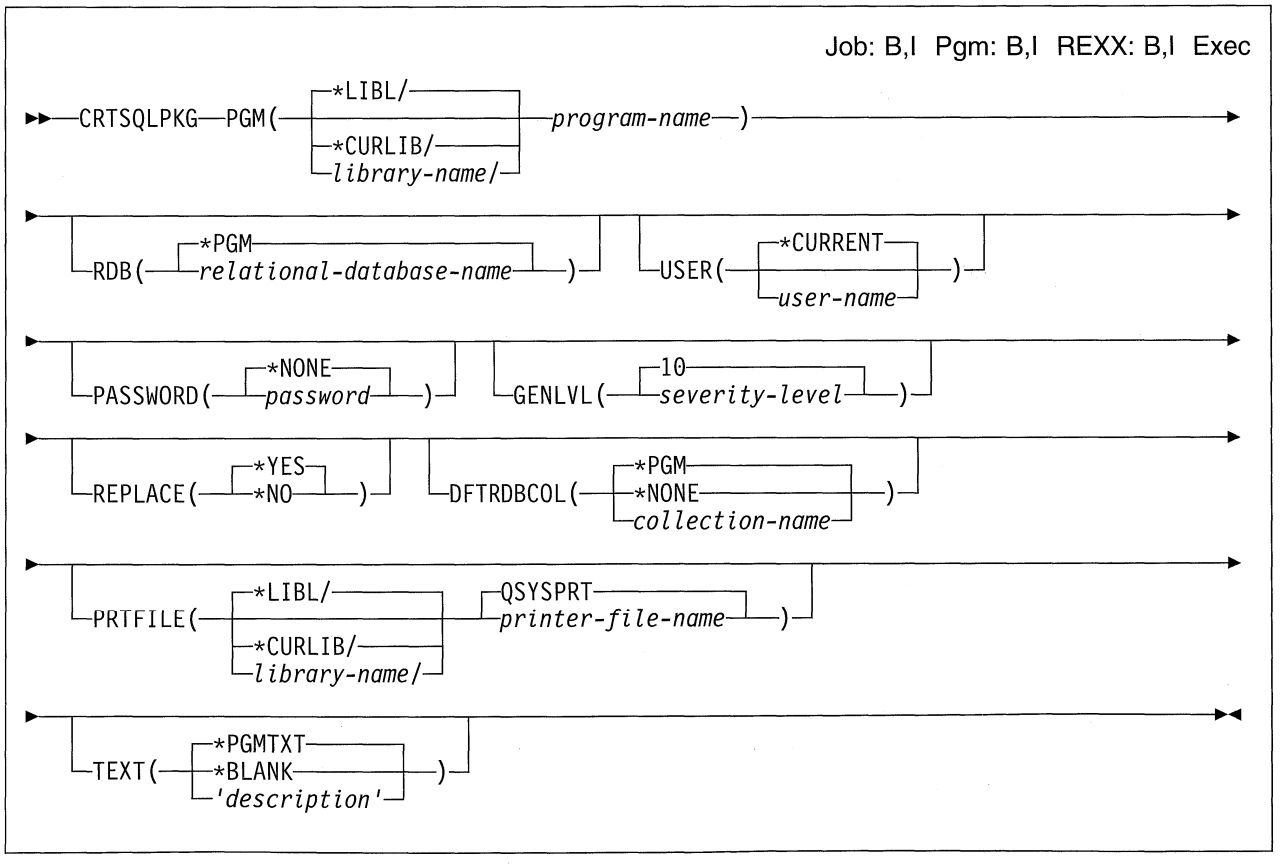
SQL Package Management

After an SQL package is created, you can manage it the same way you manage other objects on the AS/400 system, with some restrictions. You can save and restore it, send it to other systems, and grant and revoke a user's authority to the package. You can also delete it by entering the Delete SQL Package (DLTSQLPKG) command or the DROP PACKAGE SQL statement.

When a distributed SQL program is created, the name of the SQL package and an internal consistency token are saved in the program. These are used at run time to find the SQL package and verify that the SQL package is correct for this program. Because the name of the SQL package is critical for running distributed SQL programs, an SQL package cannot be moved, renamed, duplicated, or restored to a different library.

Create SQL Package (CRTSQLPKG) Command

You do not need the SQL/400 licensed program to create an SQL package on an AS. You can enter the CRTSQLPKG command to create an SQL package from a compiled distributed relational database program. You can also use this command to replace an SQL package that was created previously. A new SQL package is created on the relational database defined by the RDB parameter. The new SQL package has the same name and is placed in the same library as specified on the PKG parameter of the CRTSQL*xxx* command.



PGM

Specifies the qualified name of the program for which the SQL package is being created.

***LIBL:** Specifies that the library list is used to locate the program.

***CURLIB:** Specifies that the current library is able to find the program. If a current library entry does not exist in the library list, the QGPL library is used.

library-name: Specifies the library where the program is located.

program-name: Specifies the name of the distributed program for which the SQL package is being created.

RDB

Specifies the relational database name that identifies the remote database where the SQL package is being created.

***PGM:** Specifies that the relational database name to be used is the same as the value specified on the RDB parameter of the CRTSQL*xxx* command used when the program was created.

relational-database-name: Specifies the name of the relational database where the SQL package is to be created.

USER

Specifies the user name sent to the remote system when starting the conversation.

***CURRENT:** The user name associated with the current job is used.

user-name: Specifies the user name to be used for the remote job.

PASSWORD

Specifies the password to be used on the remote system.

***NONE:** No password is sent. If a user name is specified on the USER parameter, the value is not valid.

password: Specifies the password of the user name specified on the USER parameter.

GENLVL

Controls the generation of the SQL package. If error messages are returned with a severity greater than the GENLVL value, the SQL package is not created.

10: If a severity level value is not specified, the default severity level is 10.

severity-level: Specify a number from 0 through 40. Some suggested values are listed below:

10 warnings
20 general error messages
30 serious error messages
40 system detected error messages

Note: There are some errors that cannot be controlled by GENLVL. When those errors occur, the SQL package is not created.

REPLACE

Specifies whether or not to replace an existing SQL package of the same name with a newly created SQL package.

***YES:** Specifies that if the SQL package already exists, it will be replaced with the new SQL package.

***NO:** Specifies that the create SQL package operation will end if an SQL package already exists.

DFTRDBCOL

Identifies the default collection name to be used for unqualified names of tables, views, indexes and SQL packages with static SQL statements.

***PGM:** Specifies that the collection name to be used is the same as the DFTRDBCOL parameter value used when the program was created.

***NONE:** Specifies that unqualified names for tables, indexes, views, and SQL packages will use the search conventions defined for the *SQL and *SYS options in the SQL precompiler commands.

collection-name: Specify the name of the collection name that is to be used for unqualified tables, views, indexes and SQL packages.

PRTFILE

Specifies the qualified name of the printer device file to which the precompiler listing is directed. The file should have a minimum length of 132 characters. If a file with a record length of less than 132 characters is specified, information is lost.

***LIBL:** Specifies the library list used to locate the printer file.

***CURLIB:** Specifies that the current library for the job is used to locate the printer file. If no library entry exists in the library list, QGPL is used.

library-name: Specify the library where the printer file is located.

QSYSPRT: If a file name is not specified, the precompiler listing is directed to the IBM-supplied printer file QSYSPRT.

printer-file-name: Specify the name of the printer device file to which the precompiler listing is directed.

TEXT

Specifies text that briefly describes the program and its function.

***PGMTXT:** Specifies that the text is taken from the program.

***BLANK:** Specifies no text.

'description': Specify no more than 50 characters of text enclosed in apostrophes (').

The following sample command creates an SQL package from the distributed SQL program INVENT on relational database KC000.

```
CRTSQLPKG  INVENT  RDB(KC000)   TEXT('Inventory Check')
```

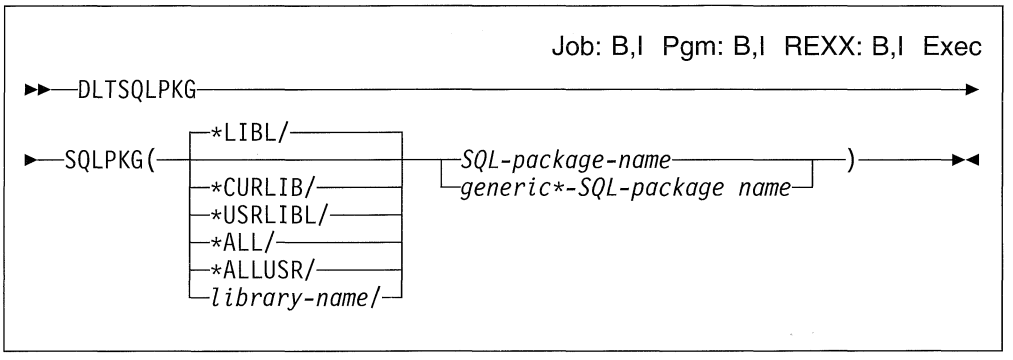
The new SQL package is created with the same options that were specified on the CRTSQL*xxx* command.

If errors are encountered while creating the SQL package, the SQL statement being processed when the error occurred and the message text for the error are written to the file identified by the PRTFILE parameter. A listing is not generated if no errors were found during the create SQL package process.

If the CRTSQL*xxx* command failed to create an SQL package (for example, the communications line failed during the precompile) but the program was created, the SQL package can be created without running the CRTSQL*xxx* command again.

Delete SQL Package (DLTSQLPKG) Command

You can use the Delete SQL Package (DLTSQLPKG) command to delete one or more SQL packages. You must enter the DLTSQLPKG command on the AS/400 system where the SQL package being deleted is located.



SQLPKG

Specifies the qualified name of the SQL package being deleted. A specific or generic SQL package name can be specified.

The possible library values are:

***LIBL:** All libraries in the user and system portions of the job's library list are searched.

***CURLIB:** The current library is searched. If no library is specified as the current library for the job, the QGPL library is used.

***USRLIBL:** Only the libraries listed in the user portion of the library list are searched.

***ALL:** All libraries in the system, including QSYS, are searched.

***ALLUSR:** All nonsystem libraries, including all user-defined libraries and the QGPL library, not just those in the job's library list are searched. Libraries whose names start with the letter Q, other than QGPL, are not searched.

library-name: Specifies the name of the library to be searched.

SQL-package-name: Specifies the name of the SQL package being deleted.

generic-SQL-package-name:* Specifies the generic name of the SQL package to be deleted. A generic name is a character string of one or more characters followed by an asterisk (*); for example, ABC*. If a generic name is specified, all SQL packages with names that begin with the generic name, and for which the user has authority, are deleted. If an asterisk is not included with the generic (prefix) name, the system assumes it to be the complete SQL package name.

You must have *OBJEXIST authority for the SQL package and at least *READ authority for the collection where it is located.

If you have the SQL/400 licensed program installed, use interactive SQL to first connect then drop the package on the AS, using the SQL DROP PACKAGE statement. If you do not have interactive SQL, run an SQL program that connects and drops the SQL package.

The following command deletes the SQL package PARTS1 in the SPIFFY collection:

```
DLTSQLPKG  SQLPKG(SPIFFY/PARTS1)
```

To delete an SQL package on a remote AS/400 system, use the Submit Remote Command (SBMRMTCMD) command to run the DLTSQLPKG command on the remote system. See "Submit Remote Command (SBMRMTCMD) Command" on page 6-8 for how to use the SBMRMTCMD command. You can also use display station pass-through to sign on the remote system to delete the SQL package. If the remote system is not an AS/400 system, pass through to that system using a remote work station program and then submit the delete SQL package command local to that system.

SQL DROP PACKAGE Statement

The DROP PACKAGE statement includes the PACKAGE parameter for distributed relational database. You can issue the DROP PACKAGE statement embedded in a program or using interactive SQL. When you issue a DROP PACKAGE statement, the SQL package and its description are deleted from the AS. This has the same result as a Delete SQL Package (DLTSQLPKG) command entered on a local system. No other objects dependent on the SQL package are deleted as a result of this statement.

You must have the following privileges on the SQL package to successfully delete it:

- The system authority *READ on the referenced collection
- The system authority *OBJEXIST on the SQL package

The following example shows how the DROP PACKAGE statement is issued:

```
DROP PACKAGE  SPIFFY.PARTS1
```

A program cannot issue a DROP PACKAGE statement for the SQL package it is currently using.

Appendix A. Application Programming Examples

This appendix contains an example application for distributed relational database use, written in RPG/400, COBOL/400 and C/400 programming languages. This example shows how to use a distributed relational database for functional specification tasks.

Business Requirement

The application for the distributed relational database in this example is parts stock management in an automobile dealer or distributor network.

This program checks the level of stock for each part in the local part stock table. If this is below the re-order point, the program then checks on the central tables to see whether there are any existing orders outstanding and what quantity has been shipped against each order.

If the net quantity (local stock, plus orders, minus shipments) is still below the re-order point, an order is placed for the part by inserting rows in the appropriate tables on the central system. A report is printed on the local system.

Technical Notes

Commitment control

This program uses the concept of Local and Remote Logical Units of Work (LUW). It is necessary to close the current LUW on one system (COMMIT) before beginning a new unit of work on another system.

Cursor repositioning

When a LUW is committed and the application connects to another database, all cursors are closed. This application requires the cursor reading the part stock file to be re-opened at the next part number. To achieve this, the cursor is defined to begin where the part number is greater than the current value of part number, and to be ordered by part number.

Note: This technique will not work if there are duplicate rows for the same part number.

Creating a Collection and Tables

```
5738PW1 V2R1M1  920327                SEU SOURCE LISTING                          03/29/92 17:16:50           PAGE    1
SOURCE FILE . . . . . .  DRDA/QLBLSRC
MEMBER  . . . . . . . .  CRTDB
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+... 0
  100       IDENTIFICATION DIVISION.                                                                    03/29/92
  200       PROGRAM-ID. CRTDB.                                                                          03/29/92
  300       ENVIRONMENT DIVISION.                                                                       03/29/92
  400       DATA DIVISION.                                                                              03/29/92
  500       WORKING-STORAGE SECTION.                                                                    03/29/92
  600            EXEC SQL INCLUDE SQLCA END-EXEC.                                                       03/29/92
  700       PROCEDURE DIVISION.                                                                         03/29/92
  800       MAIN.                                                                                       03/29/92
  900      *     -------------------------------------------------------------                          03/29/92
 1000      *  LOCATION TABLE                                                                           03/29/92
 1100      *  -------------------------------------------------------------*/--                        03/29/92
 1200            EXEC SQL                                                                              03/29/92
 1300             CREATE COLLECTION DRDA                                                               03/29/92
 1400            END-EXEC.                                                                             03/29/92
 1500           EXEC SQL                                                                               03/29/92
 1600             CREATE TABLE DRDA/PART_STOCK                                                         03/29/92
 1700                   (PART_NUM    CHAR(5)      NOT NULL,
 1800                    PART_UM     CHAR(2)      NOT NULL,
 1900                    PART_QUANT  INTEGER      NOT NULL WITH DEFAULT,                              03/29/92
 2000                    PART_ROP    INTEGER      NOT NULL,                                           03/29/92
 2100                    PART_EOQ    INTEGER      NOT NULL,                                           03/29/92
 2200                    PART_BIN    CHAR(6)      NOT NULL WITH DEFAULT                               03/29/92
 2300                  ) END-EXEC.                                                                    03/29/92
 2400             EXEC SQL                                                                           03/29/92
 2500             CREATE UNIQUE INDEX DRDA/PART_STOCI                                                03/29/92
 2600                    ON DRDA/PART_STOCK                                                          03/29/92
 2700                    (PART_NUM ASC) END-EXEC.                                                    03/29/92
 2800             EXEC SQL                                                                           03/29/92
 2900              CREATE TABLE DRDA/PART_ORDER                                                      03/29/92
 3000                   (ORDER_NUM   SMALLINT     NOT NULL,
 3100                    ORIGIN_LOC  CHAR(4)      NOT NULL,
 3200                    ORDER_TYPE  CHAR(1)      NOT NULL,
 3300                    ORDER_STAT  CHAR(1)      NOT NULL,
 3400                    NUM_ALLOC   SMALLINT     NOT NULL WITH DEFAULT,
 3500                    URG_REASON  CHAR(1)      NOT NULL WITH DEFAULT,
 3600                    CREAT_TIME  TIMESTAMP    NOT NULL,
 3700                    ALLOC_TIME  TIMESTAMP,
 3800                    CLOSE_TIME  TIMESTAMP,
 3900                    REV_REASON  CHAR(1)                                                         03/29/92
 4000                  ) END-EXEC.                                                                   03/29/92
 4100             EXEC SQL                                                                           03/29/92
 4200               CREATE UNIQUE INDEX DRDA/PART_ORDEI                                              03/29/92
 4300                    ON DRDA/PART_ORDER                                                          03/29/92
 4400                    (ORDER_NUM ASC) END-EXEC.                                                   03/29/92
 4500             EXEC SQL                                                                           03/29/92
 4600              CREATE TABLE DRDA/PART_ORDLN                                                      03/29/92
 4700                   (ORDER_NUM   SMALLINT     NOT NULL,
 4800                    ORDER_LINE  SMALLINT     NOT NULL,
 4900                    PART_NUM    CHAR(5)      NOT NULL,
 5000                    QUANT_REQ   INTEGER NOT NULL,                                               03/29/92
 5100                    LINE_STAT   CHAR(1)      NOT NULL                                           03/29/92
 5200                  ) END-EXEC.                                                                   03/29/92
 5300             EXEC SQL                                                                           03/29/92
 5400              CREATE UNIQUE INDEX PART_ORDLI                                                    03/29/92
 5500                    ON DRDA/PART_ORDLN                                                          03/29/92
 5600                    (ORDER_NUM ASC,                                                             03/29/92
 5700                     ORDER_LINE ASC) END-EXEC.                                                  03/29/92
 5800            EXEC SQL                                                                            03/29/92
 5900              CREATE TABLE DRDA/SHIPMENTLN                                                      03/29/92
 6000                   (SHIP_NUM    SMALLINT     NOT NULL,
 6100                    SHIP_LINE   SMALLINT     NOT NULL,
 6200                    ORDER_LOC   CHAR(4)      NOT NULL,
 6300                    ORDER_NUM   SMALLINT     NOT NULL,
 6400                    ORDER_LINE  SMALLINT     NOT NULL,
 6500                    PART_NUM    CHAR(5)      NOT NULL,
 6600                    QUANT_SHIP  INTEGER NOT NULL,                                               03/29/92
 6700                    QUANT_RECV  INTEGER NOT NULL WITH DEFAULT                                   03/29/92
 6800                  ) END-EXEC.                                                                   03/29/92
```

Figure A-1 (Part 1 of 2). Creating a Collection and Tables

```
5738PW1 V2R1M1  920327                SEU SOURCE LISTING                      03/29/92  17:16:50           PAGE    2
SOURCE FILE . . . . . .  DRDA/QLBLSRC
MEMBER  . . . . . . . .  CRTDB
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+... 0
 6900               EXEC SQL                                                                          03/29/92
 7000                CREATE UNIQUE INDEX SHIPMENTLI                                                   03/29/92
 7100                     ON DRDA/SHIPMENTLN                                                          03/29/92
 7200                     (SHIP_NUM ASC,                                                              03/29/92
 7300                      SHIP_LINE ASC) END-EXEC.                                                   03/29/92
 7400               EXEC SQL                                                                          03/29/92
 7500                     COMMIT         END-EXEC.                                                   03/29/92
 7600           STOP RUN.                                                                           03/29/92
                          * * * *  E N D  O F  S O U R C E  * * * *
```

Figure A-1 (Part 2 of 2). Creating a Collection and Tables

Inserting Data into the Tables

```
5738PW1 V2R1M1  920327                SEU SOURCE LISTING                        03/29/92 17:16:54           PAGE    1
SOURCE FILE . . . . . .  DRDA/QLBLSRC
MEMBER  . . . . . . . .  INSDB
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+... 0
  100       IDENTIFICATION DIVISION.                                                                    03/29/92
  200       PROGRAM-ID. INSDB.                                                                          03/29/92
  300       ENVIRONMENT DIVISION.                                                                       03/29/92
  400       DATA DIVISION.                                                                              03/29/92
  500       WORKING-STORAGE SECTION.                                                                    03/29/92
  600            EXEC SQL INCLUDE SQLCA END-EXEC.                                                       03/29/92
  700       PROCEDURE DIVISION.                                                                         03/29/92
  800       MAIN.                                                                                       03/29/92
  900                                                                                                   03/29/92
 1000                                                                                                   03/29/92
 1100      *-------------------------------------------------------------                              03/29/92
 1200      *  PART_STOCK TABLE                                                                         03/29/92
 1300      *-------------------------------------------------------------*/--                          03/29/92
 1400                                                                                                   03/29/92
 1500                                                                                                   03/29/92
 1600           EXEC SQL                                                                               03/29/92
 1700            INSERT INTO PART_STOCK                                                                03/29/92
 1800            VALUES                                                                                03/29/92
 1900            ('14020','EA',038,050,100,' ') END-EXEC.                                              03/29/92
 2000           EXEC SQL                                                                               03/29/92
 2100            INSERT INTO PART_STOCK                                                                03/29/92
 2200            VALUES                                                                                03/29/92
 2300            ('14030','EA',043,050,050,' ') END-EXEC.                                              03/29/92
 2400           EXEC SQL                                                                               03/29/92
 2500            INSERT INTO PART_STOCK                                                                03/29/92
 2600            VALUES                                                                                03/29/92
 2700            ('14040','EA',030,020,030,' ') END-EXEC.                                              03/29/92
 2800           EXEC SQL                                                                               03/29/92
 2900            INSERT INTO PART_STOCK                                                                03/29/92
 3000            VALUES                                                                                03/29/92
 3100            ('14050','EA',010,005,015,' ') END-EXEC.                                              03/29/92
 3200           EXEC SQL                                                                               03/29/92
 3300             INSERT INTO PART_STOCK                                                               03/29/92
 3400             VALUES                                                                               03/29/92
 3500             ('14060','EA',110,045,090,' ') END-EXEC.                                             03/29/92
 3600           EXEC SQL                                                                               03/29/92
 3700             INSERT INTO PART_STOCK                                                               03/29/92
 3800                VALUES                                                                            03/29/92
 3900             ('14070','EA',130,080,160,' ') END-EXEC.                                             03/29/92
 4000              EXEC SQL                                                                            03/29/92
 4100             INSERT INTO PART_STOCK                                                               03/29/92
 4200             VALUES                                                                               03/29/92
 4300             ('18020','EA',013,025,050,' ') END-EXEC.                                             03/29/92
 4400           EXEC SQL                                                                               03/29/92
 4500             INSERT INTO PART_STOCK                                                               03/29/92
 4600             VALUES                                                                               03/29/92
 4700             ('18030','EA',015,005,010,' ') END-EXEC.                                             03/29/92
 4800           EXEC SQL                                                                               03/29/92
 4900             INSERT INTO PART_STOCK                                                               03/29/92
 5000             VALUES                                                                               03/29/92
 5100             ('21010','EA',029,030,050,' ') END-EXEC.                                             03/29/92
 5200           EXEC SQL                                                                               03/29/92
 5300             INSERT INTO PART_STOCK                                                               03/29/92
 5400             VALUES                                                                               03/29/92
 5500             ('24010','EA',025,020,040,' ') END-EXEC.                                             03/29/92
 5600           EXEC SQL                                                                               03/29/92
 5700             INSERT INTO PART_STOCK                                                               03/29/92
 5800             VALUES                                                                               03/29/92
 5900             ('24080','EA',054,050,050,' ') END-EXEC.                                             03/29/92
 6000           EXEC SQL                                                                               03/29/92
 6100             INSERT INTO PART_STOCK                                                               03/29/92
 6200             VALUES                                                                               03/29/92
 6300             ('24090','EA',030,025,050,' ') END-EXEC.                                             03/29/92
 6400           EXEC SQL                                                                               03/29/92
 6500             INSERT INTO PART_STOCK                                                               03/29/92
 6600             VALUES                                                                               03/29/92
 6700             ('24100','EA',020,015,030,' ') END-EXEC.                                             03/29/92
```

Figure A-2 (Part 1 of 4). Inserting Data into the Tables

```
5738PW1 V2R1M1  920327                SEU SOURCE LISTING                      03/29/92  17:16:54           PAGE    2
SOURCE FILE . . . . . .  DRDA/QLBLSRC
MEMBER  . . . . . .  INSDB
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+... 0
 6800           EXEC SQL                                                                                  03/29/92
 6900             INSERT INTO PART_STOCK                                                                  03/29/92
 7000             VALUES                                                                                  03/29/92
 7100             ('24110','EA',052,050,080,' ') END-EXEC.                                                03/29/92
 7200           EXEC SQL                                                                                  03/29/92
 7300             INSERT INTO PART_STOCK                                                                  03/29/92
 7400             VALUES                                                                                  03/29/92
 7500             ('25010','EA',511,300,600,' ') END-EXEC.                                                03/29/92
 7600           EXEC SQL                                                                                  03/29/92
 7700             INSERT INTO PART_STOCK                                                                  03/29/92
 7800             VALUES                                                                                  03/29/92
 7900             ('36010','EA',013,005,010,' ') END-EXEC.                                                03/29/92
 8000           EXEC SQL                                                                                  03/29/92
 8100             INSERT INTO PART_STOCK                                                                  03/29/92
 8200             VALUES                                                                                  03/29/92
 8300             ('36020','EA',110,030,060,' ') END-EXEC.                                                03/29/92
 8400           EXEC SQL                                                                                  03/29/92
 8500             INSERT INTO PART_STOCK                                                                  03/29/92
 8600             VALUES                                                                                  03/29/92
 8700             ('37010','EA',415,100,200,' ') END-EXEC.                                                03/29/92
 8800           EXEC SQL                                                                                  03/29/92
 8900             INSERT INTO PART_STOCK                                                                  03/29/92
 9000             VALUES                                                                                  03/29/92
 9100             ('37020','EA',010,020,040,' ') END-EXEC.                                                03/29/92
 9200           EXEC SQL                                                                                  03/29/92
 9300             INSERT INTO PART_STOCK                                                                  03/29/92
 9400             VALUES                                                                                  03/29/92
 9500             ('37030','EA',154,055,060,' ') END-EXEC.                                                03/29/92
 9600           EXEC SQL                                                                                  03/29/92
 9700             INSERT INTO PART_STOCK                                                                  03/29/92
 9800             VALUES                                                                                  03/29/92
 9900             ('37040','EA',223,120,120,' ') END-EXEC.                                                03/29/92
10000           EXEC SQL                                                                                  03/29/92
10100             INSERT INTO PART_STOCK                                                                  03/29/92
10200             VALUES                                                                                  03/29/92
10300             ('43010','EA',110,020,040,' ') END-EXEC.                                                03/29/92
10400           EXEC SQL                                                                                  03/29/92
10500             INSERT INTO PART_STOCK                                                                  03/29/92
10600             VALUES                                                                                  03/29/92
10700             ('43020','EA',067,050,050,' ') END-EXEC.                                                03/29/92
10800           EXEC SQL                                                                                  03/29/92
10900             INSERT INTO PART_STOCK                                                                  03/29/92
11000             VALUES                                                                                  03/29/92
11100             ('48010','EA',032,030,060,' ') END-EXEC.                                                03/29/92
11200                                                                                                     03/29/92
11300      *-------------------------------------------------------------- --                            03/29/92
11400      *  PART_ORDER TABLE                                                                           03/29/92
11500      *--------------------------------------------------------------*/--                           03/29/92
11600                                                                                                     03/29/92
11700                                                                                                     03/29/92
11800                                                                                                     03/29/92
11900           EXEC SQL                                                                                  03/29/92
12000             INSERT INTO PART_ORDER                                                                  03/29/92
12100             VALUES                                                                                  03/29/92
12200          (1,'DB2B','U','O',0,' ','1991-03-12-17.00.00',NULL,NULL,NULL)                              03/29/92
12300             END-EXEC.                                                                               03/29/92
12400           EXEC SQL                                                                                  03/29/92
12500             INSERT INTO PART_ORDER                                                                  03/29/92
12600             VALUES                                                                                  03/29/92
12700              (2,'SQLA','U','O',0,' ','1991-03-12-17.01.00',                                         03/29/92
12800              NULL,NULL,NULL)                                                                        03/29/92
12900             END-EXEC.                                                                               03/29/92
13000           EXEC SQL                                                                                  03/29/92
13100             INSERT INTO PART_ORDER                                                                  03/29/92
13200             VALUES                                                                                  03/29/92
13300              (3,'SQLA','U','O',0,' ','1991-03-12-17.02.00',                                         03/29/92
13400              NULL,NULL,NULL)                                                                        03/29/92
13500             END-EXEC.                                                                               03/29/92
```

Figure A-2 (Part 2 of 4). Inserting Data into the Tables

```
5738PW1 V2R1M1  920327                SEU SOURCE LISTING                      03/29/92  17:16:54           PAGE    3
SOURCE FILE . . . . . .  DRDA/QLBLSRC
MEMBER  . . . . . .  INSDB
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+... 0
13600           EXEC SQL                                                                                  03/29/92
13700             INSERT INTO PART_ORDER                                                                  03/29/92
13800             VALUES                                                                                  03/29/92
13900              (4,'SQLA','U','O',0,' ','1991-03-12-17.03.00',                                         03/29/92
14000               NULL,NULL,NULL)                                                                       03/29/92
14100             END-EXEC.                                                                               03/29/92
14200           EXEC SQL                                                                                  03/29/92
14300             INSERT INTO PART_ORDER                                                                  03/29/92
14400             VALUES                                                                                  03/29/92
14500              (5,'DB2B','U','O',0,' ','1991-03-12-17.04.00',                                         03/29/92
14600               NULL,NULL,NULL)                                                                       03/29/92
14700             END-EXEC.                                                                               03/29/92
14800                                                                                                     03/29/92
14900      *-------------------------------------------------------------                                 03/29/92
15000      *  PART_ORDLN TABLE                                                                           03/29/92
15100      *-------------------------------------------------------------*/--                            03/29/92
15200                                                                                                     03/29/92
15300                                                                                                     03/29/92
15400           EXEC SQL                                                                                  03/29/92
15500             INSERT INTO PART_ORDLN                                                                  03/29/92
15600             VALUES                                                                                  03/29/92
15700             (1,1,'24110',005,'O') END-EXEC.                                                         03/29/92
15800           EXEC SQL                                                                                  03/29/92
15900             INSERT INTO PART_ORDLN                                                                  03/29/92
16000             VALUES                                                                                  03/29/92
16100             (1,2,'24100',021,'O') END-EXEC.                                                         03/29/92
16200           EXEC SQL                                                                                  03/29/92
16300             INSERT INTO PART_ORDLN                                                                  03/29/92
16400             VALUES                                                                                  03/29/92
16500             (1,3,'24090',018,'O') END-EXEC.                                                         03/29/92
16600           EXEC SQL                                                                                  03/29/92
16700             INSERT INTO PART_ORDLN                                                                  03/29/92
16800             VALUES                                                                                  03/29/92
16900             (2,1,'14070',004,'O') END-EXEC.                                                         03/29/92
17000           EXEC SQL                                                                                  03/29/92
17100             INSERT INTO PART_ORDLN                                                                  03/29/92
17200             VALUES                                                                                  03/29/92
17300             (2,2,'37040',043,'O') END-EXEC.                                                         03/29/92
17301           EXEC SQL                                                                                  03/29/92
17302             INSERT INTO PART_ORDLN                                                                  03/29/92
17303             VALUES                                                                                  03/29/92
17304             (2,3,'14030',015,'O') END-EXEC.                                                         03/29/92
17305           EXEC SQL                                                                                  03/29/92
17306             INSERT INTO PART_ORDLN                                                                  03/29/92
17307             VALUES                                                                                  03/29/92
17308             (3,2,'14030',025,'O') END-EXEC.                                                         03/29/92
17400           EXEC SQL                                                                                  03/29/92
17500             INSERT INTO PART_ORDLN                                                                  03/29/92
17600             VALUES                                                                                  03/29/92
17700             (3,1,'43010',003,'O') END-EXEC.                                                         03/29/92
17800           EXEC SQL                                                                                  03/29/92
17900             INSERT INTO PART_ORDLN                                                                  03/29/92
18000             VALUES                                                                                  03/29/92
18100             (4,1,'36010',013,'O') END-EXEC.                                                         03/29/92
18200           EXEC SQL                                                                                  03/29/92
18300             INSERT INTO PART_ORDLN                                                                  03/29/92
18400             VALUES                                                                                  03/29/92
18500             (5,1,'18030',005,'O') END-EXEC.                                                         03/29/92
18600                                                                                                     03/29/92
18700                                                                                                     03/29/92
18800      *-------------------------------------------------------------                                 03/29/92
18900      * SHIPMENTLN TABLE                                                                            03/29/92
19000      *-------------------------------------------------------------*/--                            03/29/92
19100                                                                                                     03/29/92
19200                                                                                                     03/29/92
19300           EXEC SQL                                                                                  03/29/92
19400             INSERT INTO SHIPMENTLN                                                                  03/29/92
19500             VALUES                                                                                  03/29/92
19600             (1,1,'DB2B',1,1,'24110',5,5) END-EXEC.                                                  03/29/92
19700           EXEC SQL                                                                                  03/29/92
19800             INSERT INTO SHIPMENTLN                                                                  03/29/92
19900             VALUES                                                                                  03/29/92
20000             (1,2,'DB2B',1,2,'24100',10,1) END-EXEC.                                                 03/29/92
20100           EXEC SQL                                                                                  03/29/92
20200             INSERT INTO SHIPMENTLN                                                                  03/29/92
20300             VALUES                                                                                  03/29/92
20400             (2,1,'SQLA',2,1,'14070',4,4) END-EXEC.                                                  03/29/92
```

Figure A-2 (Part 3 of 4). Inserting Data into the Tables

```
5738PW1 V2R1M1  920327                SEU SOURCE LISTING                      03/29/92  17:16:54          PAGE    5
SOURCE FILE . . . . . .  DRDA/QLBLSRC
MEMBER  . . . . . .  INSDB
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+... 0
20500           EXEC SQL                                                                                 03/29/92
20600             INSERT INTO SHIPMENTLN                                                                 03/29/92
20700             VALUES                                                                                 03/29/92
20800             (2,2,'SQLA',2,2,'37040',45,25) END-EXEC.                                               03/29/92
20801           EXEC SQL                                                                                 03/29/92
20802             INSERT INTO SHIPMENTLN                                                                 03/29/92
20803             VALUES                                                                                 03/29/92
20804             (2,3,'SQLA',2,3,'14030', 5,5) END-EXEC.                                                03/29/92
20805           EXEC SQL                                                                                 03/29/92
20806             INSERT INTO SHIPMENTLN                                                                 03/29/92
20807             VALUES                                                                                 03/29/92
20808             (3,1,'SQLA',2,3,'14030', 5,5) END-EXEC.                                                03/29/92
20900                                                                                                    03/29/92
21000           EXEC SQL COMMIT END-EXEC.                                                                03/29/92
21100           STOP RUN.                                                                                03/29/92
                          * * * *  E N D  O F  S O U R C E  * * * *
```

Figure A-2 (Part 4 of 4). Inserting Data into the Tables

RPG Program Example

```
5738PW1 V2R1M1  920327                SEU SOURCE LISTING                        03/29/92 17:12:48           PAGE    1
SOURCE FILE . . . . . .  DRDA/QRPGSRC
MEMBER  . . . . . . . .  DDBPT6RG
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+... 0
  100       *****************************************************************                              03/29/92
  200       *                                                               *                              03/29/92
  300       *   DESCRIPTIVE NAME = D-DB SAMPLE APPLICATION                  *                              03/29/92
  400       *                      REORDER POINT PROCESSING                 *                              03/29/92
  500       *                      AS/400                                   *                              03/29/92
  600       *                                                               *                              03/29/92
  700       *   FUNCTION =  THIS MODULE PROCESS THE PART_STOCK TABLE AND    *                              03/29/92
  800       *               FOR EACH PART BELOW THE ROP (REORDER POINT)      *                              03/29/92
  900       *               CREATES A SUPPLY ORDER AND PRINTS A REPORT.     *                              03/29/92
 1000       *                                                               *                              03/29/92
 1100       *                                                               *                              03/29/92
 1200       *      INPUT = PARAMETERS EXPLICITLY PASSED TO THIS FUNCTION:   *                              03/29/92
 1300       *                                                               *                              03/29/92
 1400       *              LOCADB         LOCAL DB NAME                     *                              03/29/92
 1500       *              REMODB         REMOTE DB NAME                    *                              03/29/92
 1600       *                                                               *                              03/29/92
 1700       *   TABLES =  PART-STOCK       - LOCAL                           *                              03/29/92
 1800       *             PART_ORDER       - REMOTE                          *                              03/29/92
 1900       *             PART_ORDLN       - REMOTE                          *                              03/29/92
 2000       *             SHIPMENTLN       - REMOTE                          *                              03/29/92
 2100       *                                                               *                              03/29/92
 2200       *   INDICATORS =   *IN89       - '0' ORDER HEADER  NOT  DONE     *                              03/29/92
 2300       *                                '1' ORDER HEADER  IS   DONE     *                              03/29/92
 2400       *                  *IN99       - '1' ABNORMAL END (SQLCOD<0)     *                              03/29/92
 2500       *                                                               *                              03/29/92
 2600       *   TO BE COMPILED WITH COMMIT(*CHG) RDB(remotedbname)          *                              03/29/92
 2700       *                                                               *                              03/29/92
 2800       *   INVOKE BY : CALL DDBPT6RG PARM(localdbname remotedbname)    *                              03/29/92
 2900       *                                                               *                              03/29/92
 3000       *   CURSORS WILL BE CLOSED IMPLICITLY (BY CONNECT) BECAUSE      *                              03/29/92
 3100       *   THERE IS NO REASON TO DO IT EXPLICITLY                      *                              03/29/92
 3200       *                                                               *                              03/29/92
 3300       *****************************************************************                              03/29/92
 3400       *                                                                                              03/29/92
 3500      FQPRINT  O   F      33     OF     PRINTER                                                       03/29/92
 3600      F*                                                                                             03/29/92
 3700      I*                                                                                             03/29/92
 3800      IMISC        DS                                                                                03/29/92
 3900      I                                    B   1   20SHORTB                                         03/29/92
 4000      I                                    B   3   60LONGB                                          03/29/92
 4100      I                                    B   7   80INDNUL                                        03/29/92
 4200      I                                        9  13 PRTTBL                                         03/29/92
 4300      I                                       14  29 LOCTBL                                         03/29/92
 4400      I I            'SQLA'                   30  33 LOC                                            03/29/92
 4500      I*                                                                                             03/29/92
 4600      I*                                                                                             03/29/92
 4700      C*                                                                                             03/29/92
 4800      C           *LIKE     DEFN SHORTB    NXTORD          NEW ORDER NR                             03/29/92
 4900      C           *LIKE     DEFN SHORTB    NXTORL          ORDER LINE NR                            03/29/92
 5000      C           *LIKE     DEFN SHORTB    RTCOD1          RTCOD NEXT_PART                          03/29/92
 5100      C           *LIKE     DEFN SHORTB    RTCOD2          RTCOD NEXT_ORD_                          03/29/92
 5200      C           *LIKE     DEFN SHORTB    CURORD          ORDER NUMBER                             03/29/92
 5300      C           *LIKE     DEFN SHORTB    CURORL          ORDER LINE                               03/29/92
 5400      C           *LIKE     DEFN LONGB     QUANTI          FOR COUNTING                             03/29/92
 5500      C           *LIKE     DEFN LONGB     QTYSTC          QTY ON STOCK                             03/29/92
 5600      C           *LIKE     DEFN LONGB     QTYORD          REORDER QTY                              03/29/92
 5700      C           *LIKE     DEFN LONGB     QTYROP          REORDER POINT                            03/29/92
 5800      C           *LIKE     DEFN LONGB     QTYREQ          QTY ORDERED                              03/29/92
 5900      C           *LIKE     DEFN LONGB     QTYREC          QTY RECEIVED                             03/29/92
 6000      C*                                                                                             03/29/92
 6100      C*                                                                                             03/29/92
 6200      C*************************************************************                                 03/29/92
 6300      C*     PARAMETERS                                            *                                 03/29/92
 6400      C*************************************************************                                 03/29/92
 6500      C*                                                                                             03/29/92
 6600      C           *ENTRY    PLIST                                                                    03/29/92
 6700      C                     PARM           LOCADB 18       LOCAL DATABASE                           03/29/92
 6800      C                     PARM           REMODB 18       REMOTE DATABASE                          03/29/92
 6900      C*                                                                                             03/29/92
 7000      C*                                                                                             03/29/92
```

Figure A-3 (Part 1 of 7). RPG Program Example

```
5738PW1 V2R1M1  920327                SEU SOURCE LISTING                      03/29/92  17:12:48           PAGE    2
SOURCE FILE . . . . . .  DRDA/QRPGSRC
MEMBER  . . . . . .  DDBPT6RG
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+... 0
 7100      C*************************************************************                                  03/29/92
 7200      C*    SQL CURSOR DECLARATIONS                                *                                  03/29/92
 7300      C*************************************************************                                  03/29/92
 7400      C*                                                                                             03/29/92
 7500      C* NEXT PART WHICH STOCK QUANTITY IS UNDER REORDER POINTS QTY                                    03/29/92
 7600      C/EXEC SQL                                                                                     03/29/92
 7700      C+     DECLARE NEXT_PART CURSOR FOR                                                            03/29/92
 7800      C+         SELECT PART_NUM,                                                                    03/29/92
 7900      C+                PART_QUANT,                                                                  03/29/92
 8000      C+                PART_ROP,                                                                    03/29/92
 8100      C+                PART_EOQ                                                                     03/29/92
 8200      C+         FROM   PART_STOCK                                                                   03/29/92
 8300      C+         WHERE  PART_ROP > PART_QUANT                                                        03/29/92
 8400      C+           AND  PART_NUM > :PRTTBL                                                           03/29/92
 8500      C+         ORDER BY PART_NUM ASC                                                               03/29/92
 8600      C/END-EXEC                                                                                     03/29/92
 8700      C*                                                                                             03/29/92
 8800      C* ORDERS WHICH ARE ALREADY MADE FOR CURRENT PART                                              03/29/92
 8900      C/EXEC SQL                                                                                     03/29/92
 9000      C+     DECLARE NEXT_ORDER_LINE CURSOR FOR                                                      03/29/92
 9100      C+         SELECT A.ORDER_NUM,                                                                 03/29/92
 9200      C+                ORDER_LINE,                                                                  03/29/92
 9300      C+                QUANT_REQ                                                                    03/29/92
 9400      C+         FROM   PART_ORDLN A,                                                                03/29/92
 9500      C+                PART_ORDER B                                                                 03/29/92
 9600      C+         WHERE  PART_NUM  = :PRTTBL                                                          03/29/92
 9700      C+         AND    LINE_STAT  <> 'C'                                                            03/29/92
 9800      C+         AND    A.ORDER_NUM = B.ORDER_NUM                                                    03/29/92
 9900      C+         AND    ORDER_TYPE  = 'R'                                                            03/29/92
10000      C/END-EXEC                                                                                     03/29/92
10100      C*                                                                                             03/29/92
10200      C*************************************************************                                  03/29/92
10300      C*     SQL RETURN CODE HANDLING                               *                                  03/29/92
10400      C*************************************************************                                  03/29/92
10500      C/EXEC SQL                                                                                     03/29/92
10600      C+     WHENEVER SQLERROR GO TO DBERRO                                                          03/29/92
10700      C/END-EXEC                                                                                     03/29/92
10800      C/EXEC SQL                                                                                     03/29/92
10900      C+     WHENEVER SQLWARNING CONTINUE                                                            03/29/92
11000      C/END-EXEC                                                                                     03/29/92
11100      C*                                                                                             03/29/92
11200      C*                                                                                             03/29/92
11300      C*************************************************************                                  03/29/92
11400      C*    PROCESS  - MAIN PROGRAM LOGIC                          *                                  03/29/92
11500      C*    MAIN PROCEDURE WORKS   WITH LOCAL DATABASE             *                                  03/29/92
11600      C*************************************************************                                  03/29/92
11700      C*                                                                                             03/29/92
11800      C*CLEAN UP TO PERMIT RE-RUNNING OF TEST DATA                                                   03/29/92
11900      C                     EXSR CLEANU                                                              03/29/92
12000      C*                                                                                             03/29/92
12100      C*                                                                                             03/29/92
12200      C           RTCOD1    DOUEQ100                                                               03/29/92
12300      C*                                                                                             03/29/92
12400      C/EXEC SQL                                                                                     03/29/92
12500      C+          CONNECT   TO  :LOCADB                                                            03/29/92
12600      C/END-EXEC                                                                                     03/29/92
12700      C/EXEC SQL                                                                                     03/29/92
12800      C+          OPEN      NEXT_PART                                                              03/29/92
12900      C/END-EXEC                                                                                     03/29/92
13000      C/EXEC SQL                                                                                     03/29/92
13100      C+          FETCH     NEXT_PART                                                              03/29/92
13200      C+           INTO     :PRTTBL,                                                               03/29/92
13300      C+                    :QTYSTC,                                                               03/29/92
13400      C+                    :QTYROP,                                                               03/29/92
13500      C+                    :QTYORD                                                                03/29/92
13600      C/END-EXEC                                                                                     03/29/92
```

Figure A-3 (Part 2 of 7). RPG Program Example

```
5738PW1 V2R1M1  920327                SEU SOURCE LISTING                      03/29/92  17:12:48           PAGE    3
SOURCE FILE . . . . . .  DRDA/QRPGSRC
MEMBER  . . . . . . . .  DDBPT6RG
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+... 0
13700     C                     MOVE SQLCOD    RTCOD1                                                      03/29/92
13800     C/EXEC SQL                                                                                       03/29/92
13900     C+          COMMIT                                                                               03/29/92
14000     C/END-EXEC                                                                                       03/29/92
14100     C           RTCOD1    IFNE 100                                                                   03/29/92
14200     C                     EXSR CHECKO                                                                03/29/92
14300     C                     ENDIF                                                                      03/29/92
14400     C*                                                                                               03/29/92
14500     C                     ENDDO                                                                      03/29/92
14600     C*                                                                                               03/29/92
14700     C                     GOTO SETLR                                                                 03/29/92
14800     C*                                                                                               03/29/92
14900     C*                                                                                               03/29/92
15000     C*****************************************************************                              03/29/92
15100     C*    SQL RETURN CODE HANDLING ON ERROR SITUATIONS               *                              03/29/92
15200     C*****************************************************************                              03/29/92
15300     C*                                                                                               03/29/92
15400     C           DBERRO    TAG                                                                        03/29/92
15500     C*         *-------------*                                                                       03/29/92
15600     C                     EXCPTERRLIN                                                                03/29/92
15700     C                     MOVE *ON       *IN99                                                       03/29/92
15800     C/EXEC SQL                                                                                       03/29/92
15900     C+          WHENEVER  SQLERROR  CONTINUE                                                       03/29/92
16000     C/END-EXEC                                                                                       03/29/92
16100     C/EXEC SQL                                                                                       03/29/92
16200     C+          ROLLBACK                                                                           03/29/92
16300     C/END-EXEC                                                                                       03/29/92
16400     C/EXEC SQL                                                                                       03/29/92
16500     C+          WHENEVER  SQLERROR  GO TO  DBERRO                                                  03/29/92
16600     C/END-EXEC                                                                                       03/29/92
16700     C*                                                                                               03/29/92
16800     C*                                                                                               03/29/92
16900     C           SETLR     TAG                                                                      03/29/92
17000     C*         *-------------*                                                                       03/29/92
17100     C/EXEC SQL                                                                                       03/29/92
17200     C+          CONNECT   RESET                                                                    03/29/92
17300     C/END-EXEC                                                                                       03/29/92
17400     C                     MOVE *ON       *INLR                                                       03/29/92
17500     C*                                                                                               03/29/92
17600     C*****************************************************************                              03/29/92
17700     C*    THE END OF THE PROGRAM                                     *                              03/29/92
17800     C*****************************************************************                              03/29/92
17900     C*                                                                                               03/29/92
18000     C*                                                                                               03/29/92
18100     C*****************************************************************                              03/29/92
18200     C* SUBROUTINES TO WORK WITH REMOTE DATABASES                     *                              03/29/92
18300     C*****************************************************************                              03/29/92
18400     C*                                                                                               03/29/92
18500     C*                                                                                               03/29/92
18600     C           CHECKO    BEGSR                                                                    03/29/92
18700     C*         *--------------*                                                                      03/29/92
18800     C*****************************************************************                              03/29/92
18900     C* CHECKS WHAT IS CURRENT ORDER AND SHIPMENT STATUS FOR THE PART *                              03/29/92
19000     C* IF ORDERED AND SHIPPED IS LESS THAN REORDER POINT OF PART,    *                              03/29/92
19100     C* PERFORMS A SUBROUTINE WHICH MAKES AN ORDER.                   *                              03/29/92
19200     C*****************************************************************                              03/29/92
19300     C*                                                                                               03/29/92
19400     C                     MOVE 0         RTCOD2                                                      03/29/92
19500     C                     MOVE 0         QTYREQ                                                      03/29/92
19600     C                     MOVE 0         QTYREC                                                      03/29/92
19700     C*                                                                                               03/29/92
19800     C/EXEC SQL                                                                                       03/29/92
19900     C+          CONNECT   TO   :REMODB                                                             03/29/92
20000     C/END-EXEC                                                                                       03/29/92
20100     C/EXEC SQL                                                                                       03/29/92
20200     C+          OPEN      NEXT_ORDER_LINE                                                          03/29/92
20300     C/END-EXEC                                                                                       03/29/92
```

Figure A-3 (Part 3 of 7). RPG Program Example

```
5738PW1 V2R1M1  920327                SEU SOURCE LISTING                    03/29/92  17:12:48          PAGE    4
SOURCE FILE . . . . . .  DRDA/QRPGSRC
MEMBER  . . . . . .  DDBPT6RG
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+... 0
20400      C*                                                                                          03/29/92
20500      C           RTCOD2    DOWNE100                                                          03/29/92
20600      C*                                                                                          03/29/92
20700      C/EXEC SQL                                                                                  03/29/92
20800      C+          FETCH     NEXT_ORDER_LINE                                                   03/29/92
20900      C+           INTO     :CURORD,                                                          03/29/92
21000      C+                    :CURORL,                                                          03/29/92
21100      C+                    :QUANTI                                                           03/29/92
21200      C/END-EXEC                                                                                  03/29/92
21300      C*                                                                                          03/29/92
21400      C           SQLCOD    IFEQ 100                                                          03/29/92
21500      C                     MOVE 100       RTCOD2                                             03/29/92
21600      C                     ELSE                                                              03/29/92
21700      C                     ADD  QUANTI    QTYREQ                                             03/29/92
21800      C*                                                                                          03/29/92
21900      C/EXEC SQL                                                                                  03/29/92
22000      C+          SELECT    SUM(QUANT_RECV)                                                   03/29/92
22100      C+            INTO    :QUANTI:INDNUL
22200      C+            FROM    SHIPMENTLN                                                        03/29/92
22300      C+            WHERE   ORDER_LOC  = :LOC                                                 03/29/92
22400      C+              AND   ORDER_NUM  = :CURORD                                              03/29/92
22500      C+              AND   ORDER_LINE = :CURORL                                              03/29/92
22600      C/END-EXEC                                                                                  03/29/92
22700      C*                                                                                          03/29/92
22800      C           INDNUL    IFGE 0                                                            03/29/92
22900      C                     ADD  QUANTI    QTYREC                                             03/29/92
23000      C                     ENDIF                                                             03/29/92
23100      C*                                                                                          03/29/92
23200      C                     ENDIF                                                             03/29/92
23300      C                     ENDDO                                                             03/29/92
23400      C*                                                                                          03/29/92
23500      C/EXEC SQL                                                                                  03/29/92
23600      C+     CLOSE NEXT_ORDER_LINE                                                              03/29/92
23700      C/END-EXEC                                                                                  03/29/92
23800      C*                                                                                          03/29/92
23900      C           QTYSTC    ADD  QTYREQ    QUANTI                                             03/29/92
24000      C                     SUB  QUANTI    QTYREC                                             03/29/92
24100      C*                                                                                          03/29/92
24200      C           QTYROP    IFGT QUANTI                                                       03/29/92
24300      C                     EXSR ORDERP                                                       03/29/92
24400      C                     ENDIF                                                             03/29/92
24500      C*                                                                                          03/29/92
24600      C/EXEC SQL                                                                                  03/29/92
24700      C+          COMMIT                                                                    03/29/92
24800      C/END-EXEC                                                                                  03/29/92
24900      C*                                                                                          03/29/92
25000      C                     ENDSR                          CHECKO                             03/29/92
25100      C*                                                                                          03/29/92
25200      C*                                                                                          03/29/92
25300      C           ORDERP    BEGSR                                                             03/29/92
25400      C*         *---------------*                                                              03/29/92
25500      C*************************************************************                          03/29/92
25600      C* MAKES AN ORDER. IF FIRST TIME, PERFORMS THE SUBROUTINE, WHICH *                       03/29/92
25700      C* SEARCHES FOR NEW ORDER NUMBER AND MAKES THE ORDER HEADER.     *                       03/29/92
25800      C* AFTER THAT MAKES ORDER LINES USING REORDER QUANTITY FOR THE   *                       03/29/92
25900      C* PART. FOR EVERY ORDERED PART WRITES A LINE ON REPORT.         *                       03/29/92
26000      C*************************************************************                          03/29/92
26100      C*                                                                                          03/29/92
26200      C           *IN89     IFEQ *OFF                      FIRST ORDER ?                      03/29/92
26300      C                     EXSR STRORD                                                       03/29/92
26400      C                     MOVE *ON       *IN89           ORD.HEAD.DONE                      03/29/92
26500      C                     EXCPTHEADER                    WRITE HEADERS                      03/29/92
26600      C                     ENDIF                                                             03/29/92
```

Figure A-3 (Part 4 of 7). RPG Program Example

```
5738PW1 V2R1M1  920327                SEU SOURCE LISTING                      03/29/92  17:12:48           PAGE    5
SOURCE FILE . . . . . .  DRDA/QRPGSRC
MEMBER  . . . . . .  DDBPT6RG
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+... 0
26700     C*                                                                                          03/29/92
26800     C                     ADD  1         NXTORL           NEXT ORD.LIN                        03/29/92
26900     C/EXEC SQL                                                                                  03/29/92
27000     C+          INSERT                                                                          03/29/92
27100     C+            INTO    PART_ORDLN                                                            03/29/92
27200     C+                   (ORDER_NUM,                                                            03/29/92
27300     C+                    ORDER_LINE,                                                           03/29/92
27400     C+                    PART_NUM,                                                             03/29/92
27500     C+                    QUANT_REQ,                                                            03/29/92
27600     C+                    LINE_STAT)                                                            03/29/92
27700     C+          VALUES   (:NXTORD,                                                              03/29/92
27800     C+                    :NXTORL,                                                              03/29/92
27900     C+                    :PRTTBL,                                                              03/29/92
28000     C+                    :QTYORD,                                                              03/29/92
28100     C+                    'O')                                                                  03/29/92
28200     C/END-EXEC                                                                                  03/29/92
28300     C*                                                                                          03/29/92
28400     C           *INOF     IFEQ *ON                                                              03/29/92
28500     C                     EXCPTHEADER                                                           03/29/92
28600     C                     END                                                                   03/29/92
28700     C                     EXCPTDETAIL                                                           03/29/92
28800     C*                                                                                          03/29/92
28900     C                     ENDSR                            ORDERP                               03/29/92
29000     C*                                                                                          03/29/92
29100     C*                                                                                          03/29/92
29200     C           STRORD    BEGSR                                                                 03/29/92
29300     C*         *---------------*                                                                03/29/92
29400     C*************************************************************                             03/29/92
29500     C* SEARCHES FOR NEXT ORDER NUMBER AND MAKES AN ORDER HEADER      *                         03/29/92
29600     C* USING THAT NUMBER. WRITES ALSO HEADERS ON REPORT.             *                         03/29/92
29700     C*************************************************************                             03/29/92
29800     C*                                                                                          03/29/92
29900     C/EXEC SQL                                                                                  03/29/92
30000     C+          SELECT    (MAX(ORDER_NUM) + 1)                                                  03/29/92
30100     C+            INTO    :NXTORD                                                               03/29/92
30200     C+            FROM    PART_ORDER                                                            03/29/92
30300     C/END-EXEC                                                                                  03/29/92
30400     C/EXEC SQL                                                                                  03/29/92
30500     C+          INSERT                                                                          03/29/92
30600     C+            INTO    PART_ORDER                                                            03/29/92
30700     C+                   (ORDER_NUM,                                                            03/29/92
30800     C+                    ORIGIN_LOC,                                                           03/29/92
30900     C+                    ORDER_TYPE,                                                           03/29/92
31000     C+                    ORDER_STAT,                                                           03/29/92
31100     C+                    CREAT_TIME)                                                           03/29/92
31200     C+          VALUES   (:NXTORD,                                                              03/29/92
31300     C+                    :LOC,                                                                 03/29/92
31400     C+                    'R',                                                                  03/29/92
31500     C+                    'O',                                                                  03/29/92
31600     C+                    CURRENT TIMESTAMP)                                                    03/29/92
31700     C/END-EXEC                                                                                  03/29/92
31800     C                     ENDSR                            STRORD                               03/29/92
31900     C*                                                                                          03/29/92
32000     C*                                                                                          03/29/92
32100     C           CLEANU    BEGSR                                                                 03/29/92
32200     C*         *---------------*                                                                03/29/92
32300     C*************************************************************                             03/29/92
32400     C* THIS SUBROUTINE IS ONLY REQUIRED IN A TEST ENVIRONMENT                                   03/29/92
32500     C* TO RESET THE DATA TO PERMIT RE-RUNNING OF THE TEST                                       03/29/92
32600     C*************************************************************                             03/29/92
32700     C*                                                                                          03/29/92
32800     C/EXEC SQL                                                                                  03/29/92
32900     C+          CONNECT   TO   :REMODB                                                          03/29/92
33000     C/END-EXEC                                                                                  03/29/92
33100     C/EXEC SQL                                                                                  03/29/92
33200     C+          DELETE                                                                          03/29/92
33300     C+            FROM    PART_ORDLN                                                            03/29/92
33400     C+           WHERE    ORDER_NUM IN                                                          03/29/92
33500     C+                   (SELECT    ORDER_NUM                                                  03/29/92
33600     C+                      FROM    PART_ORDER                                                 03/29/92
33700     C+                      WHERE   ORDER_TYPE = 'R')                                          03/29/92
33800     C/END-EXEC                                                                                  03/29/92
```

Figure A-3 (Part 5 of 7). RPG Program Example

```
5738PW1 V2R1M1  920327                SEU SOURCE LISTING                      03/29/92  17:12:48           PAGE    7
SOURCE FILE . . . . . .  DRDA/QRPGSRC
MEMBER  . . . . . .  DDBPT6RG
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+... 0
33900      C/EXEC SQL                                                                                    03/29/92
34000      C+          DELETE                                                                            03/29/92
34100      C+            FROM    PART_ORDER                                                              03/29/92
34200      C+           WHERE    ORDER_TYPE = 'R'                                                        03/29/92
34300      C/END-EXEC                                                                                    03/29/92
34400      C/EXEC SQL                                                                                    03/29/92
34500      C+          COMMIT                                                                            03/29/92
34600      C/END-EXEC                                                                                    03/29/92
34700      C*                                                                                            03/29/92
34800      C                     ENDSR                         CLEANU                                    03/29/92
34900      C*                                                                                            03/29/92
35000      C*                                                                                            03/29/92
35100      C*************************************************************                               03/29/92
35200      O* OUTPUTLINES FOR THE REPORT                                 *                               03/29/92
35300      O*************************************************************                               03/29/92
35400      O*                                                                                            03/29/92
35500      OQPRINT  E    2           HEADER                                                              03/29/92
35600      O                                 +  0 '***** ROP PROCESSING'                                 03/29/92
35700      O                                 +  1 'REPORT *****'                                         03/29/92
35800      O*                                                                                            03/29/92
35900      OQPRINT  E 2              HEADER                                                              03/29/92
36000      O                                 +  0 '   ORDER NUMBER = '                                   03/29/92
36100      O                         NXTORDZ +  0                                                        03/29/92
36200      O*                                                                                            03/29/92
36300      OQPRINT  E 1              HEADER                                                              03/29/92
36400      O                                 +  0 '-------------------------'                            03/29/92
36500      O                                 +  0 '---------'                                            03/29/92
36600      O*                                                                                            03/29/92
36700      OQPRINT  E 1              HEADER                                                              03/29/92
36800      O                                 +  0 '   LINE     '                                         03/29/92
36900      O                                 +  0 'PART          '                                       03/29/92
37000      O                                 +  0 'QTY    '                                              03/29/92
37100      O*                                                                                            03/29/92
37200      OQPRINT  E 1              HEADER                                                              03/29/92
37300      O                                 +  0 '  NUMBER   '                                         03/29/92
37400      O                                 +  0 'NUMBER      '                                         03/29/92
37500      O                                 +  0 'REQUESTED '                                           03/29/92
37600      O*                                                                                            03/29/92
37700      OQPRINT  E 11             HEADER                                                              03/29/92
37800      O                                 +  0 '-------------------------'                            03/29/92
37900      O                                 +  0 '---------'                                            03/29/92
38000      O*                                                                                            03/29/92
38100      OQPRINT  EF1              DETAIL                                                              03/29/92
38200      O                         NXTORLZ +  4                                                        03/29/92
38300      O                         PRTTBL  +  4                                                        03/29/92
38400      O                         QTYORD1 +  4                                                        03/29/92
38500      O*                                                                                            03/29/92
38600      OQPRINT  T 2      LRN99                                                                       03/29/92
38700      O                                 +  0 '-------------------------'                            03/29/92
38800      O                                 +  0 '---------'                                            03/29/92
38900      OQPRINT  T 1      LRN99                                                                       03/29/92
39000      O                                 +  0 'NUMBER OF LINES '                                     03/29/92
39100      O                                 +  0 'CREATED = '                                           03/29/92
39200      O                         NXTORLZ +  0                                                        03/29/92
39300      O*                                                                                            03/29/92
39400      OQPRINT  T 1      LRN99                                                                       03/29/92
39500      O                                 +  0 '-------------------------'                            03/29/92
39600      O                                 +  0 '---------'                                            03/29/92
39700      O*                                                                                            03/29/92
39800      OQPRINT  T 2      LRN99                                                                       03/29/92
39900      O                                 +  0 '*********'                                            03/29/92
40000      O                                 +  0 ' END OF PROGRAM '                                     03/29/92
40100      O                                 +  0 '********'                                             03/29/92
40200      O*                                                                                            03/29/92
```

Figure A-3 (Part 6 of 7). RPG Program Example

```
5738PW1 V2R1M1  920327               SEU SOURCE LISTING                      03/29/92  17:12:48          PAGE    8
SOURCE FILE . . . . . .  DRDA/QRPGSRC
MEMBER  . . . . . . . .  DDBPT6RG
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+... 0
40300      OQPRINT  E 2             ERRLIN                                                                  03/29/92
40400      O                                +  0 '** ERROR **'                                              03/29/92
40500      O                                +  0 '** ERROR **'                                              03/29/92
40600      O                                +  0 '** ERROR **'                                              03/29/92
40700      OQPRINT  E 1             ERRLIN                                                                  03/29/92
40800      O                                +  0 '* SQLCOD:'                                                03/29/92
40900      O                        SQLCODM +  0                                                            03/29/92
41000      O                                  33 '*'                                                        03/29/92
41100      OQPRINT  E 1             ERRLIN                                                                  03/29/92
41200      O                                +  0 '* SQLSTATE:'                                              03/29/92
41300      O                        SQLSTT  +  2                                                            03/29/92
41400      O                                  33 '*'                                                        03/29/92
41500      OQPRINT  E 1             ERRLIN                                                                  03/29/92
41600      O                                +  0 '** ERROR **'                                              03/29/92
41700      O                                +  0 '** ERROR **'                                              03/29/92
41800      O                                +  0 '** ERROR **'                                              03/29/92
```

Figure A-3 (Part 7 of 7). RPG Program Example

COBOL Program Example

```
5738PW1 V2R1M1  920327                SEU SOURCE LISTING                      03/29/92  17:12:35           PAGE    1
SOURCE FILE . . . . . .  DRDA/QLBLSRC
MEMBER  . . . . . . . .  DDBPT6CB
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+... 0
  100       IDENTIFICATION DIVISION.
  200      *-----------------------
  300       PROGRAM-ID. DDBPT6CB.                                                                      03/29/92
  400      ****************************************************************                            03/29/92
  500      *   MODULE NAME = DDBPT6CB                                                                 03/29/92
  600      *
  700      *   DESCRIPTIVE NAME = D-DB SAMPLE APPLICATION
  800      *                      REORDER POINT PROCESSING
  900      *                      AS/400                                                              03/29/92
 1000      *                      COBOL
 1100      *
 1200      *   FUNCTION =  THIS MODULE PROCESS THE PART_STOCK TABLE AND
 1300      *               FOR EACH PART BELOW THE ROP (REORDER POINT)
 1400      *               CHECKS THE EXISTING ORDERS AND SHIPMENTS,                                   03/29/92
 1500      *               CREATES A SUPPLY ORDER AND PRINTS A REPORT.                                 03/29/92
 1600      *
 1700      *      DEPENDENCIES = NONE                                                                  03/29/92
 1800      *
 1900      *      INPUT = PARAMETERS EXPLICITLY PASSED TO THIS FUNCTION:
 2000      *
 2100      *              LOCAL-DB       LOCAL DB NAME                                                03/29/92
 2200      *              REMOTE-DB      REMOTE DB NAME                                               03/29/92
 2300      *
 2400      *   TABLES =  PART-STOCK       - LOCAL                                                     03/29/92
 2500      *             PART_ORDER       - REMOTE                                                    03/29/92
 2600      *             PART_ORDLN       - REMOTE                                                    03/29/92
 2700      *             SHIPMENTLN       - REMOTE                                                    03/29/92
 2800      *                                                                                          03/29/92
 2900      *   CRTSQLCBL  SPECIAL PARAMETERS                                                          03/29/92
 3000      *    PGM(DDBPT6CB) RDB(remotedbname) OPTION(*APOST *APOSTSQL)                              03/29/92
 3100      *                                                                                          03/29/92
 3200      *   INVOKE BY : CALL DDBPT6CB PARM(localdbname remotedbname)                               03/29/92
 3300      *                                                                                          03/29/92
 3400      ****************************************************************                          03/29/92
 3500       ENVIRONMENT DIVISION.
 3600      *--------------------
 3700       INPUT-OUTPUT SECTION.
 3800       FILE-CONTROL.
 3900           SELECT RELAT  ASSIGN TO PRINTER-QPRINT.                                                03/29/92
 4000       DATA DIVISION.
 4100      *--------------
 4200       FILE SECTION.
 4300      *-------------                                                                              03/29/92
 4400       FD  RELAT
 4500           RECORD CONTAINS 33 CHARACTERS
 4600           LABEL RECORDS ARE OMITTED
 4700           DATA RECORD IS REPREC.
 4800       01  REPREC          PIC  X(33).
 4900       WORKING-STORAGE SECTION.
 5000      *-----------------------                                                                    03/29/92
 5100      *    PRINT LINE  DEFINITIONS                                                               03/29/92
 5200       01  LINE0           PIC  X(33) VALUE  SPACES.
 5300       01  LINE1           PIC  X(33) VALUE
 5400           '***** ROP PROCESSING REPORT *****'.
 5500       01  LINE2.
 5600         05  FILLER        PIC  X(18) VALUE  '   ORDER NUMBER = '.
 5700         05  MASK0         PIC  ZZZ9.
 5800         05  FILLER        PIC  X(11) VALUE  SPACES.
 5900       01  LINE3           PIC  X(33) VALUE
 6000           '---------------------------------'.
 6100       01  LINE4           PIC  X(33) VALUE
 6200           '   LINE     PART        QTY     '.
 6300       01  LINE5           PIC  X(33) VALUE
 6400           '  NUMBER   NUMBER     REQUESTED  '.
```

Figure A-4 (Part 1 of 6). COBOL Program Example

```
5738PW1 V2R1M1  920327                SEU SOURCE LISTING                      03/29/92  17:12:35           PAGE    2
SOURCE FILE . . . . . .  DRDA/QLBLSRC
MEMBER  . . . . . . . .  DDBPT6CB
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+... 0
 6500        01  LINE6.
 6600          05  FILLER        PIC  XXXX  VALUE SPACES.
 6700          05  MASK1         PIC  ZZZ9.
 6800          05  FILLER        PIC  XXXX  VALUE SPACES.
 6900          05  PART-TABLE    PIC  XXXXX.
 7000          05  FILLER        PIC  XXXX  VALUE SPACES.
 7100          05  MASK2         PIC  Z,ZZZ,ZZZ.ZZ.
 7200        01  LINE7.
 7300          05  FILLER        PIC  X(26) VALUE
 7400            'NUMBER OF LINES CREATED = '.
 7500          05  MASK3         PIC  ZZZ9.
 7600          05  FILLER        PIC  XXX   VALUE  SPACES.
 7700        01  LINE8           PIC  X(33) VALUE
 7800            '********* END OF PROGRAM ********'.
 7900       *    MISCELLANEOUS DEFINITIONS                                                    03/29/92
 8000        01  WHAT-TIME       PIC  X     VALUE '1'.
 8100            88  FIRST-TIME            VALUE '1'.
 8200        01  CONTL           PIC  S9999 COMP-4 VALUE ZEROS.                             03/29/92
 8300        01  CONTD           PIC  S9999 COMP-4 VALUE ZEROS.                             03/29/92
 8400        01  RTCODE1         PIC  S9999 COMP-4 VALUE ZEROS.                             03/29/92
 8500        01  RTCODE2         PIC  S9999 COMP-4.                                         03/29/92
 8600        01  NEXT-NUM        PIC  S9999 COMP-4.                                         03/29/92
 8700        01  IND-NULL        PIC  S9999 COMP-4.                                         03/29/92
 8800        01  LOC-TABLE       PIC  X(16).
 8900        01  ORD-TABLE       PIC  S9999 COMP-4.                                         03/29/92
 9000        01  ORL-TABLE       PIC  S9999 COMP-4.                                         03/29/92
 9100        01  QUANT-TABLE     PIC  S9(9) COMP-4.                                         03/29/92
 9200        01  QTY-TABLE       PIC  S9(9) COMP-4.                                         03/29/92
 9300        01  ROP-TABLE       PIC  S9(9) COMP-4.                                         03/29/92
 9400        01  EOQ-TABLE       PIC  S9(9) COMP-4.                                         03/29/92
 9500        01  QTY-REQ         PIC  S9(9) COMP-4.                                         03/29/92
 9600        01  QTY-REC         PIC  S9(9) COMP-4.                                         03/29/92
 9700       * CONSTANT  FOR LOCATION NUMBER                                                 03/29/92
 9800        01  XPARM.                                                                     03/29/92
 9900            05   LOC        PIC  X(4)  VALUE 'SQLA'.                                  03/29/92
10000       * DEFINITIONS FOR ERROR MESSAGE HANDLING                                       03/29/92
10100        01  ERROR-MESSAGE.                                                            03/29/92
10200            05   MSG-ID.                                                             03/29/92
10300            10   MSG-ID-1     PIC  X(2)                                              03/29/92
10400                 VALUE 'SQ'.                                                         03/29/92
10500            10   MSG-ID-2     PIC 99999.                                             03/29/92
10600       ******************************                                                03/29/92
10700       *    SQLCA INCLUDE           *                                                03/29/92
10800       ******************************                                                03/29/92
10900            EXEC SQL INCLUDE SQLCA    END-EXEC.
11000                                                                                      03/29/92
11100        LINKAGE SECTION.                                                             03/29/92
11200       *----------------                                                             03/29/92
11300        01  LOCAL-DB        PIC  X(18).                                              03/29/92
11400        01  REMOTE-DB       PIC  X(18).                                              03/29/92
11500                                                                                      03/29/92
11600        PROCEDURE DIVISION USING LOCAL-DB REMOTE-DB.                                 03/29/92
11700       *------------------                                                           03/29/92
11800       ******************************                                                03/29/92
11900       *    SQL CURSOR DECLARATION *                                                 03/29/92
12000       ******************************                                                03/29/92
12100       * RE-POSITIONABLE CURSOR : POSITION AFTER LAST PART_NUM                       03/29/92
12200            EXEC SQL DECLARE NEXT_PART CURSOR FOR
12300                 SELECT PART_NUM,
12400                        PART_QUANT,
12500                        PART_ROP,
12600                        PART_EOQ
12700                 FROM   PART_STOCK
12800                 WHERE  PART_ROP > PART_QUANT
12900                   AND  PART_NUM > :PART-TABLE                                        03/29/92
13000                 ORDER BY PART_NUM ASC                                                03/29/92
13100            END-EXEC.
```

Figure A-4 (Part 2 of 6). COBOL Program Example

```
5738PW1 V2R1M1  920327                SEU SOURCE LISTING                      03/29/92  17:12:35           PAGE    3
SOURCE FILE . . . . . .  DRDA/QLBLSRC
MEMBER  . . . . . .  DDBPT6CB
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+... 0
13200      * CURSOR FOR ORDER LINES                                                                    03/29/92
13300           EXEC SQL DECLARE NEXT_ORDER_LINE CURSOR FOR
13400                SELECT A.ORDER_NUM,
13500                       ORDER_LINE,
13600                       QUANT_REQ
13700                FROM   PART_ORDLN A,                                                              03/29/92
13800                       PART_ORDER B
13900                WHERE  PART_NUM  = :PART-TABLE
14000                AND    LINE_STAT  <> 'C'                                                          03/29/92
14100                AND    A.ORDER_NUM = B.ORDER_NUM
14200                AND    ORDER_TYPE  = 'R'
14300           END-EXEC.
14400      ******************************                                                             03/29/92
14500      *    SQL RETURN CODE HANDLING*                                                              03/29/92
14600      ******************************                                                             03/29/92
14700           EXEC SQL WHENEVER SQLERROR GO TO DB-ERROR END-EXEC.
14800           EXEC SQL WHENEVER SQLWARNING CONTINUE  END-EXEC.                                       03/29/92
14900                                                                                                  03/29/92
15000       MAIN-PROGRAM-PROC.                                                                         03/29/92
15100      *------------------                                                                         03/29/92
15200           PERFORM START-UP THRU START-UP-EXIT.                                                   03/29/92
15300           PERFORM MAIN-PROC THRU MAIN-EXIT UNTIL RTCODE1 = 100.                                  03/29/92
15400       END-OF-PROGRAM.                                                                            03/29/92
15500      *---------------                                                                            03/29/92
15600      ****                                                                                       03/29/92
15700           EXEC SQL CONNECT RESET END-EXEC.                                                       03/29/92
15800      ****
15900           CLOSE RELAT.
16000           GOBACK.
16100       MAIN-PROGRAM-EXIT. EXIT.                                                                   03/29/92
16200      *------------------                                                                         03/29/92
16300                                                                                                  03/29/92
16400       START-UP.                                                                                  03/29/92
16500      *----------                                                                                 03/29/92
16600           OPEN OUTPUT RELAT.                                                                     03/29/92
16700      ****                                                                                       03/29/92
16800           EXEC SQL COMMIT END-EXEC.                                                              03/29/92
16900      ****                                                                                       03/29/92
17000           PERFORM CLEAN-UP THRU CLEAN-UP-EXIT.                                                   03/29/92
17100      ********************************                                                           03/29/92
17200      *    CONNECT TO LOCAL DATABASE *                                                            03/29/92
17300      ********************************                                                           03/29/92
17400      ****                                                                                       03/29/92
17500           EXEC SQL CONNECT TO :LOCAL-DB END-EXEC.                                                03/29/92
17600      ****                                                                                       03/29/92
17700       START-UP-EXIT. EXIT.                                                                       03/29/92
17800      *------------                                                                               03/29/92
17900           EJECT
18000       MAIN-PROC.
18100      *---------
18200           EXEC SQL OPEN NEXT_PART END-EXEC.                                                      03/29/92
18300           EXEC SQL
18400                FETCH NEXT_PART
18500                INTO  :PART-TABLE,
18600                      :QUANT-TABLE,
18700                      :ROP-TABLE,
18800                      :EOQ-TABLE
18900           END-EXEC.
19000           IF SQLCODE = 100
19100              MOVE 100 TO RTCODE1                                                                 03/29/92
19200              PERFORM TRAILER-PROC THRU TRAILER-EXIT                                              03/29/92
19300           ELSE
19400              MOVE 0 TO RTCODE2
19500              MOVE 0 TO QTY-REQ
19600              MOVE 0 TO QTY-REC
19700      * --- IMPLICIT "CLOSE" CAUSED BY COMMIT ---                                                 03/29/92
19800      ****                                                                                       03/29/92
19900              EXEC SQL COMMIT END-EXEC                                                            03/29/92
20000      ****                                                                                       03/29/92
```

Figure A-4 (Part 3 of 6). COBOL Program Example

```
5738PW1 V2R1M1  920327                SEU SOURCE LISTING                      03/29/92  17:12:35           PAGE    4
SOURCE FILE . . . . . .  DRDA/QLBLSRC
MEMBER  . . . . . . . .  DDBPT6CB
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+... 0
20100       *********************************                                                          03/29/92
20200       *    CONNECT TO REMOTE DATABASE *                                                          03/29/92
20300       *********************************                                                          03/29/92
20400       ****                                                                                       03/29/92
20500               EXEC SQL CONNECT TO :REMOTE-DB END-EXEC                                            03/29/92
20600       ****                                                                                       03/29/92
20700               EXEC SQL OPEN NEXT_ORDER_LINE END-EXEC                                             03/29/92
20800               PERFORM UNTIL RTCODE2 = 100
20900                  EXEC SQL                                                                        03/29/92
21000                       FETCH NEXT_ORDER_LINE
21100                       INTO  :ORD-TABLE,
21200                             :ORL-TABLE,
21300                             :QTY-TABLE
21400                  END-EXEC
21500                  IF SQLCODE = 100
21600                     MOVE 100 TO RTCODE2
21700                     EXEC SQL CLOSE NEXT_ORDER_LINE END-EXEC
21800                  ELSE
21900                     ADD QTY-TABLE TO QTY-REQ
22000                     EXEC SQL
22100                          SELECT SUM(QUANT_RECV)                                                  03/29/92
22200                          INTO   :QTY-TABLE:IND-NULL
22300                          FROM   SHIPMENTLN                                                       03/29/92
22400                          WHERE  ORDER_LOC  = :LOC
22500                          AND    ORDER_NUM  = :ORD-TABLE
22600                          AND    ORDER_LINE = :ORL-TABLE
22700                     END-EXEC
22800                     IF IND-NULL NOT < 0
22900                        ADD QTY-TABLE TO QTY-REC
23000                     END-IF
23100                  END-IF
23200               END-PERFORM
23300               IF ROP-TABLE > QUANT-TABLE + QTY-REQ - QTY-REC
23400                  PERFORM ORDER-PROC THRU ORDER-EXIT
23500               END-IF
23600            END-IF.
23700       ****                                                                                       03/29/92
23800            EXEC SQL COMMIT END-EXEC.                                                             03/29/92
23900       ****                                                                                       03/29/92
24000       *********************************                                                          03/29/92
24100       *    RECONNECT TO LOCAL DATABASE *                                                         03/29/92
24200       *********************************                                                          03/29/92
24300       ****                                                                                       03/29/92
24400            EXEC SQL CONNECT TO :LOCAL-DB END-EXEC.                                               03/29/92
24500       ****                                                                                       03/29/92
24600        MAIN-EXIT. EXIT.
24700       *---------------
24800        ORDER-PROC.
24900       *----------
25000            IF FIRST-TIME
25100               MOVE '2' TO WHAT-TIME
25200               PERFORM CREATE-ORDER-PROC THRU CREATE-ORDER-EXIT.                                  03/29/92
25300            ADD 1 TO CONTL.
25400            EXEC SQL
25500                 INSERT
25600                 INTO    PART_ORDLN                                                             03/29/92
25700                        (ORDER_NUM,
25800                         ORDER_LINE,
25900                         PART_NUM,
26000                         QUANT_REQ,
26100                         LINE_STAT)
26200                 VALUES (:NEXT-NUM,
26300                         :CONTL,
26400                         :PART-TABLE,
26500                         :EOQ-TABLE,
26600                         'O')
26700            END-EXEC.
26800            PERFORM DETAIL-PROC THRU DETAIL-EXIT.
26900        ORDER-EXIT. EXIT.
27000       *----------------
```

Figure A-4 (Part 4 of 6). COBOL Program Example

```
5738PW1 V2R1M1  920327                SEU SOURCE LISTING                      03/29/92  17:12:35           PAGE    5
SOURCE FILE . . . . . .  DRDA/QLBLSRC
MEMBER  . . . . . . . .  DDBPT6CB
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+... 0
27100                                                                                                  03/29/92
27200       CREATE-ORDER-PROC.                                                                         03/29/92
27300      *------------------                                                                         03/29/92
27400      *GET NEXT ORDER NUMBER                                                                      03/29/92
27500           EXEC SQL                                                                              03/29/92
27600                SELECT (MAX(ORDER_NUM) + 1)                                                      03/29/92
27700                INTO   :NEXT-NUM:IND-NULL                                          03/29/92
27800                FROM   PART_ORDER                                                                03/29/92
27900           END-EXEC.                                                                             03/29/92
28000           IF IND-NULL < 0                                                                       03/29/92
28100             MOVE 1 TO NEXT-NUM.                                                                 03/29/92
28200           EXEC SQL                                                                              03/29/92
28300                INSERT                                                                           03/29/92
28400                INTO    PART_ORDER                                                               03/29/92
28500                       (ORDER_NUM,                                                               03/29/92
28600                        ORIGIN_LOC,                                                              03/29/92
28700                        ORDER_TYPE,                                                              03/29/92
28800                        ORDER_STAT,                                                              03/29/92
28900                        CREAT_TIME)                                                              03/29/92
29000                VALUES (:NEXT-NUM,                                                               03/29/92
29100                        :LOC, 'R', 'O',                                                          03/29/92
29200                        CURRENT TIMESTAMP)                                                       03/29/92
29300              END-EXEC.                                                                          03/29/92
29400           MOVE NEXT-NUM TO MASK0.                                                               03/29/92
29500           PERFORM HEADER-PROC THRU HEADER-EXIT.                                                 03/29/92
29600       CREATE-ORDER-EXIT. EXIT.                                                                  03/29/92
29700      *------------------                                                                         03/29/92
29800                                                                                                  03/29/92
29900       DB-ERROR.                                                                                  03/29/92
30000      *--------                                                                                   03/29/92
30100           PERFORM ERROR-MSG-PROC THRU ERROR-MSG-EXIT.                                           03/29/92
30200      ************************                                                                   03/29/92
30300      *    ROLLBACK THE LUW *                                                                     03/29/92
30400      ************************                                                                   03/29/92
30500           EXEC SQL WHENEVER SQLERROR CONTINUE END-EXEC.                                         03/29/92
30600      ****                                                                                       03/29/92
30700           EXEC SQL ROLLBACK WORK END-EXEC.                                                      03/29/92
30800      ****                                                                                       03/29/92
30900           PERFORM END-OF-PROGRAM THRU MAIN-PROGRAM-EXIT.                                        03/29/92
31000      * -- NEXT LINE INCLUDED TO RESET THE "GO TO" DEFAULT --                                     03/29/92
31100           EXEC SQL WHENEVER SQLERROR GO TO DB-ERROR END-EXEC.                                   03/29/92
31200                                                                                                  03/29/92
31300       ERROR-MSG-PROC.                                                                            03/29/92
31400      *-----------                                                                                03/29/92
31500           MOVE  SQLCODE   TO  MSG-ID-2.                                                         03/29/92
31600           DISPLAY 'SQL STATE =' SQLSTATE ' SQLCODE =' MSG-ID-2.                                 03/29/92
31700      * -- ADD HERE ANY ADDITIONAL ERROR MESSAGE HANDLING --                                      03/29/92
31800       ERROR-MSG-EXIT. EXIT.                                                                      03/29/92
31900      *----------------                                                                           03/29/92
32000                                                                                                  03/29/92
32100      *******************                                                                        03/29/92
32200      * REPORT PRINTING *                                                                        03/29/92
32300      *******************                                                                        03/29/92
32400       HEADER-PROC.                                                                               03/29/92
32500      *-----------                                                                                03/29/92
32600           WRITE REPREC FROM LINE1 AFTER ADVANCING PAGE.
32700           WRITE REPREC FROM LINE2 AFTER ADVANCING 3 LINES.
32800           WRITE REPREC FROM LINE3 AFTER ADVANCING 2 LINES.
32900           WRITE REPREC FROM LINE4 AFTER ADVANCING 1 LINES.
33000           WRITE REPREC FROM LINE5 AFTER ADVANCING 1 LINES.
33100           WRITE REPREC FROM LINE3 AFTER ADVANCING 1 LINES.
33200           WRITE REPREC FROM LINE0 AFTER ADVANCING 1 LINES.
33300       HEADER-EXIT. EXIT.
33400      *-----------------
```

Figure A-4 (Part 5 of 6). COBOL Program Example

```
5738PW1 V2R1M1  920327                SEU SOURCE LISTING                      03/29/92  17:12:35          PAGE    7
SOURCE FILE . . . . . .  DRDA/QLBLSRC
MEMBER  . . . . . .  DDBPT6CB
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+... 0
33500        DETAIL-PROC.
33600       *-----------
33700            ADD 1 TO CONTD.
33800            IF CONTD > 50
33900               MOVE 1 TO CONTD
34000               PERFORM HEADER-PROC THRU HEADER-EXIT
34100            END-IF
34200            MOVE CONTL     TO MASK1.
34300            MOVE EOQ-TABLE TO MASK2.
34400            WRITE REPREC FROM LINE6 AFTER ADVANCING 1 LINES.
34500        DETAIL-EXIT. EXIT.
34600       *-----------------
34700        TRAILER-PROC.
34800       *------------
34900            MOVE CONTL TO MASK3.
35000            WRITE REPREC FROM LINE3 AFTER ADVANCING 2 LINES.
35100            WRITE REPREC FROM LINE7 AFTER ADVANCING 2 LINES.
35200            WRITE REPREC FROM LINE3 AFTER ADVANCING 2 LINES.
35300            WRITE REPREC FROM LINE8 AFTER ADVANCING 1 LINES.
35400        TRAILER-EXIT. EXIT.
35500       *------------------
35600       ************************************************************                03/29/92
35700       * THIS PARAGRAPH IS ONLY REQUIRED IN A TEST ENVIRONMENT*                    03/29/92
35800       * TO RESET THE DATA TO PERMIT RE-RUNNING OF THE TEST   *                    03/29/92
35900       ************************************************************                03/29/92
36000        CLEAN-UP.                                                                  03/29/92
36100       *---------                                                                  03/29/92
36200       *********************************                                          03/29/92
36300       *    CONNECT TO REMOTE DATABASE *                                          03/29/92
36400       *********************************                                          03/29/92
36500       ****                                                                       03/29/92
36600            EXEC SQL CONNECT TO :REMOTE-DB END-EXEC.                              03/29/92
36700       ****                                                                       03/29/92
36800       *---------------------DELETE ORDER ROWS FOR RERUNABILITY                    03/29/92
36900               EXEC SQL                                                           03/29/92
37000                    DELETE                                                        03/29/92
37100                    FROM    PART_ORDLN                                            03/29/92
37200                    WHERE   ORDER_NUM IN                                          03/29/92
37300                               (SELECT  ORDER_NUM                                 03/29/92
37400                                FROM    PART_ORDER                                03/29/92
37500                                WHERE   ORDER_TYPE = 'R')                         03/29/92
37600               END-EXEC.                                                          03/29/92
37700               EXEC SQL                                                           03/29/92
37800                    DELETE                                                        03/29/92
37900                    FROM    PART_ORDER                                            03/29/92
38000                    WHERE   ORDER_TYPE = 'R'                                      03/29/92
38100               END-EXEC.                                                          03/29/92
38200       ****                                                                       03/29/92
38300            EXEC SQL COMMIT END-EXEC.                                             03/29/92
38400       ****                                                                       03/29/92
38500        CLEAN-UP-EXIT. EXIT.                                                      03/29/92
38600       *-------------                                                             03/29/92
                            * * * *  E N D  O F  S O U R C E  * * * *
```

Figure A-4 (Part 6 of 6). COBOL Program Example

C Program Example

```
5738PW1 V2R1M1  920327                SEU SOURCE LISTING                      03/29/92  17:11:13          PAGE    1
SOURCE FILE . . . . . .  DRDA/QCSRC
MEMBER  . . . . . . . .  DDBPT6C
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+... 0
  100      /*************************************************************************/                      03/29/92
  200      /*   MODULE NAME = DDBPT6C                                               */                      03/29/92
  300      /*                                                                       */                      03/29/92
  400      /*   DESCRIPTIVE NAME:  D-DB SAMPLE APPLICATION                          */                      03/29/92
  500      /*                      REORDER POINT PROCESSING                         */                      03/29/92
  600      /*                      AS/400                                           */                      03/29/92
  700      /*                      C/400                                            */                      03/29/92
  800      /*                                                                       */                      03/29/92
  900      /*   FUNCTION:   THIS MODULE PROCESS THE PART_STOCK TABLE AND            */                      03/29/92
 1000      /*               FOR EACH PART BELOW THE ROP (REORDER POINT)             */                      03/29/92
 1100      /*               CREATES A SUPPLY ORDER.                                 */                      03/29/92
 1200      /*                                                                       */                      03/29/92
 1300      /*   OUTPUT:  BATCH : SPOOLFILE                                          */                      03/29/92
 1400      /*            INTER : DISPLY                                             */                      03/29/92
 1500      /*                                                                       */                      03/29/92
 1600      /*   LOCAL TABLES:    PART_STOCK                                         */                      03/29/92
 1700      /*                                                                       */                      03/29/92
 1800      /*   REMOTE TABLES:   PART_ORDER, PART_ORDLN, SHIPMENTLN                 */                      03/29/92
 1900      /*                                                                       */                      03/29/92
 2000      /*   COMPILE OPTIONS:                                                    */                      03/29/92
 2100      /*   CRTSQLC    PGM(DDBPT6C) COMMIT(*CHG) RDB(rdbname)                   */                      03/29/92
 2200      /*                                                                       */                      03/29/92
 2300      /*   INVOKED BY: CALL PGM(DDBPT6C) PARM('lcldbname' 'rmtdbname')         */                      03/29/92
 2400      /*************************************************************************/                      03/29/92
 2500                                                                                                       03/29/92
 2600 #include <stdlib.h>
 2700 #include <string.h>                                                                                   03/29/92
 2800 #include <stdio.h>
 2900                                                                                                       03/29/92
 3000               EXEC SQL BEGIN DECLARE SECTION;                                                         03/29/92
 3100                                                                                                       03/29/92
 3200       char loc       [4] = "SQLA";              /* dealer's database name */
 3300       char remote_db [18] = "                  "; /* sample remote database */
 3400       char local_db [18] = "                   ";  /* sample local  database */
 3500                                                                                                       03/29/92
 3600       char part_table [5] = " ";      /* part number in table part_stock    */
 3700       long  quant_table;             /* quantity in stock, tbl part_stock   */                       03/29/92
 3800       long  rop_table;             /* reorder point    , tbl part_stock     */                       03/29/92
 3900       long  eoq_table;             /* reorder quantity , tbl part_stock     */                       03/29/92
 4000                                                                                                       03/29/92
 4100       short next_num;              /* next order nbr,table part_order       */                       03/29/92
 4200                                                                                                       03/29/92
 4300       short  ord_table;            /* order nbr.       , tbl order_line     */                       03/29/92
 4400       short  orl_table;            /* order line       , tbl order_line     */                       03/29/92
 4500       long qty_table;              /* ordered quantity , tbl order_line     */                       03/29/92
 4600       long line_count = 0;         /* total number of order lines           */                       03/29/92
 4700       short ind_null;              /* null indicator for qty_table          */                       03/29/92
 4800       short  contl = 0;            /* continuation line, tbl order_line     */                       03/29/92
 4900                                                                                                       03/29/92
 5000               EXEC SQL END   DECLARE SECTION;                                                         03/29/92
 5100               EXEC SQL INCLUDE SQLCA;                                                                 03/29/92
 5200               EXEC SQL WHENEVER SQLERROR go to error_tag;                                             03/29/92
 5300               EXEC SQL WHENEVER SQLWARNING CONTINUE;                                                  03/29/92
 5400                                                                                                       03/29/92
 5500 /*************************************************************************/                          03/29/92
 5600 /* Other Variables                                                       */                          03/29/92
 5700 /*************************************************************************/                          03/29/92
 5800                                                                                                       03/29/92
 5900       char first_time, what_time;                                                                     03/29/92
 6000       long  qty_rec = 0, qty_req = 0;                                                                 03/29/92
 6100                                                                                                       03/29/92
```

Figure A-5 (Part 1 of 4). C Program Example

```
5738PW1 V2R1M1  920327                SEU SOURCE LISTING                      03/29/92  17:11:13           PAGE    2
SOURCE FILE . . . . . .  DRDA/QCSRC
MEMBER  . . . . . .  DDBPT6C
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+... 0
 6200 /*****************************************************************************/                          03/29/92
 6300 /* Function Declaration                                                      */                          03/29/92
 6400 /*****************************************************************************/                          03/29/92
 6500                                                                                                           03/29/92
 6600 declare_cursor () {                                                                                       03/29/92
 6700                                                                                                           03/29/92
 6800 /* SQL Cursor declaration and reposition for local UW                        */                          03/29/92
 6900                                                                                                           03/29/92
 7000               EXEC SQL DECLARE NEXT_PART CURSOR FOR                                                       03/29/92
 7100                        SELECT PART_NUM, PART_QUANT, PART_ROP, PART_EOQ                                    03/29/92
 7200                        FROM   PART_STOCK                                                                  03/29/92
 7300                        WHERE  PART_ROP > PART_QUANT                                                       03/29/92
 7400                        AND    PART_NUM > :part_table                                                      03/29/92
 7500                        ORDER BY PART_NUM;                                                                 03/29/92
 7600 /* SQL Cursor declaration and connect for RUW                               */                          03/29/92
 7700                                                                                                           03/29/92
 7800               EXEC SQL DECLARE NEXT_OLINE CURSOR FOR                                                      03/29/92
 7900                        SELECT A.ORDER_NUM, ORDER_LINE, QUANT_REQ                                          03/29/92
 8000                        FROM   PART_ORDLN A,                                                               03/29/92
 8100                               PART_ORDER B                                                                03/29/92
 8200                        WHERE  PART_NUM  = :part_table                                                     03/29/92
 8300                        AND    LINE_STAT  <> 'C'                                                           03/29/92
 8400                        AND    A.ORDER_NUM = B.ORDER_NUM                                                   03/29/92
 8500                        AND    ORDER_TYPE  = 'R';                                                          03/29/92
 8600                                                                                                           03/29/92
 8700  /* upline exit function in connectable state                               */                          03/29/92
 8800                                                                                                           03/29/92
 8900               EXEC SQL COMMIT;                                                                            03/29/92
 9000 goto function_exit;                                                                                       03/29/92
 9100 error_tag:                                                                                                03/29/92
 9200       error_function();                                                                                   03/29/92
 9300 function_exit: ;                                                                                          03/29/92
 9400  }  /* function declare_cursor                                              */                          03/29/92
 9500                                                                                                           03/29/92
 9600 delete_for_rerun () {                                                                                     03/29/92
 9700                                                                                                           03/29/92
 9800 /* Clean up for rerunability in test environment                           */                          03/29/92
 9900               EXEC SQL CONNECT TO :remote_db;                                                             03/29/92
10000               EXEC SQL  DELETE                                                                            03/29/92
10100                         FROM    PART_ORDLN                                                               03/29/92
10200                         WHERE   ORDER_NUM IN                                                             03/29/92
10300                                 (SELECT  ORDER_NUM                                                       03/29/92
10400                                  FROM    PART_ORDER                                                      03/29/92
10500                                  WHERE   ORDER_TYPE = 'R');                                              03/29/92
10600               EXEC SQL  DELETE                                                                            03/29/92
10700                         FROM    PART_ORDER                                                               03/29/92
10800                         WHERE   ORDER_TYPE = 'R';                                                        03/29/92
10900  /* upline exit function in connectable state                               */                          03/29/92
11000               EXEC SQL COMMIT;                                                                            03/29/92
11100               EXEC SQL CONNECT TO :local_db;                                                              03/29/92
11200 goto function_exit;                                                                                      03/29/92
11300 error_tag:                                                                                               03/29/92
11400       error_function();                                                                                  03/29/92
11500 function_exit: ;                                                                                         03/29/92
11600  }  /* function delete_for_rerun                                           */                          03/29/92
11700                                                                                                          03/29/92
11800 calculate_order_quantity () {                                                                            03/29/92
11900                                                                                                          03/29/92
12000 /* available qty = Stock qty + qty in order - qty received                  */                          03/29/92
12100                                                                                                          03/29/92
12200       EXEC SQL    OPEN NEXT_PART;                                                                      03/29/92
12300       EXEC SQL    FETCH NEXT_PART                                                                      03/29/92
12400                   INTO  :part_table, :quant_table, :rop_table, :eoq_table;            03/29/92
12500                                                                                                          03/29/92
12600            if (sqlca.sqlcode == 100) {                                                                   03/29/92
12700               printf("-------------------------------\n");                                             03/29/92
12800               printf("NUMBER OF LINES CREATED = %d\n",line_count);                                      03/29/92
12900               printf("-------------------------------\n");                                             03/29/92
13000               printf("*****  END OF PROGRAM  *********\n");                                            03/29/92
13100               rop_table = 0;            /* no (more) orders to process      */                        03/29/92
13200            }                                                                                            03/29/92
```

Figure A-5 (Part 2 of 4). C Program Example

```
5738PW1 V2R1M1  920327                SEU SOURCE LISTING                      03/29/92  17:11:13           PAGE    3
SOURCE FILE . . . . . .  DRDA/QCSRC
MEMBER  . . . . . .  DDBPT6C
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+... 0
13300           else  {      qty_rec = 0;                                                                  03/29/92
13400                        qty_req = 0;                                                                  03/29/92
13500 /*                                                                          */                       03/29/92
13600                  EXEC SQL    COMMIT;                                                                03/29/92
13700                  EXEC SQL CONNECT TO :remote_db;                                                    03/29/92
13800                  EXEC SQL OPEN NEXT_OLINE;                                                          03/29/92
13900                  do {                                                                               03/29/92
14000                       EXEC SQL  FETCH NEXT_OLINE                                                    03/29/92
14100                                 INTO  :ord_table, :orl_table, :qty_table;                  03/29/92
14200                       qty_rec = qty_rec + qty_table;                                                03/29/92
14300                     } while(sqlca.sqlcode != 100);                                                  03/29/92
14400                     EXEC SQL CLOSE NEXT_OLINE;                                                      03/29/92
14500                     EXEC SQL  SELECT SUM(QUANT_RECV)                                                03/29/92
14600                               INTO   :qty_table:ind_null                                  03/29/92
14700                               FROM   SHIPMENTLN                                                     03/29/92
14800                               WHERE  ORDER_LOC  = :loc                                              03/29/92
14900                               AND    ORDER_NUM  = :ord_table                                        03/29/92
15000                               AND    ORDER_LINE = :orl_table;                                       03/29/92
15100                     if (ind_null != 0)                                                              03/29/92
15200                         qty_rec = qty_rec + qty_table;                                              03/29/92
15300                  } /* end of else branch                                           */               03/29/92
15400 goto function_exit;                                                                                 03/29/92
15500 error_tag:                                                                                          03/29/92
15600        error_function();                                                                            03/29/92
15700 function_exit: ;                                                                                    03/29/92
15800 } /* end of calculate_order_quantity                                             */                 03/29/92
15900                                                                                                     03/29/92
16000 process_order () {                                                                                  03/29/92
16100                                                                                                     03/29/92
16200 /* insert order and order_line in remote database                               */                   03/29/92
16300                                                                                                     03/29/92
16400     if (contl == 0) {                                                                               03/29/92
16500                                                                                                     03/29/92
16600            EXEC SQL  SELECT (MAX(ORDER_NUM) + 1)                                                    03/29/92
16700                      INTO   :next_num                                                               03/29/92
16800                      FROM   PART_ORDER;                                                             03/29/92
16900            EXEC SQL INSERT INTO    PART_ORDER                                                       03/29/92
17000                     (ORDER_NUM, ORIGIN_LOC, ORDER_TYPE, ORDER_STAT, CREAT_TIME)                     03/29/92
17100                      VALUES (:next_num, :loc, 'R', 'O', CURRENT TIMESTAMP);                 03/29/92
17200            printf("*****  ROP PROCESSING  *********\n");                                           03/29/92
17300            printf("ORDER NUMBER = %d \n\n",next_num);                                             03/29/92
17400            printf("--------------------------------\n");                                         03/29/92
17500            printf("   LINE     PART       QTY      \n");                                          03/29/92
17600            printf("   NBR      NBR        REQUESTED\n");                                         03/29/92
17700            printf("--------------------------------\n");                                         03/29/92
17800            contl = contl + 1;                                                                     03/29/92
17900        }  /* if contl == 0                                                        */               03/29/92
18000                                                                                                     03/29/92
18100     EXEC SQL INSERT INTO PART_ORDLN                                                               03/29/92
18200               (ORDER_NUM, ORDER_LINE, PART_NUM, QUANT_REQ, LINE_STAT)                             03/29/92
18300               VALUES (:next_num, :contl, :part_table, :eoq_table, 'O');              03/29/92
18400     line_count = line_count + 1;                                                                    03/29/92
18500     printf("   %d        %.5s      %d\n",                                                          03/29/92
18600              line_count,part_table,eoq_table);                                                     03/29/92
18700     contl = contl + 1;                                                                             03/29/92
18800                                                                                                     03/29/92
18900  /* upline exit function in connectable state                                   */                   03/29/92
19000            EXEC SQL COMMIT;                                                                       03/29/92
19100 /* RECONNECT TO LOCAL DATABASE                                                  */                 03/29/92
19200            EXEC SQL CONNECT TO :local_db;                                                         03/29/92
19300                                                                                                     03/29/92
19400 goto function_exit;                                                                                 03/29/92
19500 error_tag:                                                                                          03/29/92
19600        error_function();                                                                            03/29/92
19700 function_exit: ;                                                                                    03/29/92
19800 }  /* end of function process_order                                            */                   03/29/92
19900                                                                                                     03/29/92
```

Figure A-5 (Part 3 of 4). C Program Example

```
5738PW1 V2R1M1  920327                SEU SOURCE LISTING                    03/29/92  17:11:13          PAGE    4
SOURCE FILE . . . . . .  DRDA/QCSRC
MEMBER  . . . . . .  DDBPT6C
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+... 0
20000 error_function () {                                                                              03/29/92
20100 /*                                                                                 */            03/29/92
20200           printf("************************\n");                                                  03/29/92
20300           printf("*      SQL ERROR       *\n");                                                  03/29/92
20400           printf("************************\n");                                                  03/29/92
20500           printf("SQLCODE    = %d\n",sqlca.sqlcode);                                             03/29/92
20600           printf("SQLSTATE    = %5s",sqlca.sqlstate);                                            03/29/92
20700           printf("\n************************\n");                                                03/29/92
20800           EXEC SQL WHENEVER SQLERROR CONTINUE;                                                   03/29/92
20900           EXEC SQL ROLLBACK;                                                                     03/29/92
21000           EXEC SQL CONNECT RESET;                                                                03/29/92
21100 exit (999);                                                                                      03/29/92
21200 }                                                                                                03/29/92
21300                                                                                                  03/29/92
21400 main(int argc, char *argv[])  {
21500   memcpy(local_db,argv[1],strlen(argv[1]));
21600   memcpy(remote_db,argv[2],strlen(argv[2]));
21700 /* clean up                                                                       */            03/29/92
21800              declare_cursor();                                                                   03/29/92
21900              delete_for_rerun();                                                                 03/29/92
22000                                                                                                  03/29/92
22100 /* main-line,  state is connectable                                              */            03/29/92
22200                                                                                                  03/29/92
22300     do {                                                                                        03/29/92
22400                calculate_order_quantity ();                                                      03/29/92
22500                if (rop_table > quant_table + qty_req - qty_rec) {                                03/29/92
22600                    process_order();                                                              03/29/92
22700                    quant_table = qty_req = qty_rec = 0;                                          03/29/92
22800                }                                                                                 03/29/92
22900        } while (sqlca.sqlcode == 0);                                                             03/29/92
23000 /* RECONNECT TO APPLICATION SERVER                                               */            03/29/92
23100           EXEC SQL CONNECT RESET;                                                                03/29/92
23200 exit(0);                                                                                         03/29/92
23300                                                                                                  03/29/92
23400                                                                                                  03/29/92
23500 } /* end of main                                                                 */            03/29/92
                              * * * *  E N D  O F  S O U R C E  * * * *
```

Figure A-5 (Part 4 of 4). C Program Example

Program Output Example

```
*****  ROP PROCESSING  *********
ORDER NUMBER = 6
--------------------------------
   LINE      PART       QTY
   NBR       NBR        REQUESTED
--------------------------------
   1         14020      100
   2         14030      50
   3         18020      50
   4         21010      50
   5         37020      40
--------------------------------
NUMBER OF LINES CREATED = 5
--------------------------------
*****  END OF PROGRAM  *********
```

Figure A-6. Program Output Example

Appendix B. Cross-Platform Distributed Relational Database Notes

This manual describes AS/400 support for distributed relational databases in a network of AS/400 systems (a *like* environment). Many distributed relational database implementations exist in a network of different DRDA-supporting platforms. This appendix provides a list of tips and techniques you may need to consider when using the AS/400 system in an *unlike* environment.

This appendix describes some conditions you need to consider when working with another specific IBM product. It is not intended to be a comprehensive list. Many problems or conditions like the ones described here depend significantly on your application. You can get more information on the differences between the various IBM platforms from the *Distributed Data Application Programming Guide*.

CCSID Considerations

When you work with a distributed relational database in an unlike environment, coded character set identifiers (CCSIDs) need to be set up and used properly. The AS/400 system is shipped with a default value that may need to be changed to work in an unlike environment. Also, the AS/400 system supports some CCSIDs for DBCS that are not supported by the DB2 and SQL/DS database managers. This section discusses these two conditions and provides you with a way to work around them.

AS/400 System Value QCCSID

The AS/400 system is shipped with a QCCSID value set to 65535. Data tagged with this CCSID is not to be converted by the receiving system. You may not be able to connect to an unlike system when your AS/400 AR is using this CCSID. Also, you may not be able to use source files that are tagged with this CCSID to create applications on unlike systems.

As stated in "Coded Character Set Identifier (CCSID)" on page 10-9, the CCSID used at connection time is determined by the job CCSID. Source files are tagged with the job CCSID you are using when the source file is created. When a job begins, its CCSID is determined by the user profile the job is running under. The user profile can, and as a default does, use the system value QCCSID.

If you are connecting to a system that does not support a CCSID, or if you are creating a source file that will need to run on such a system, you need to change your job CCSID. You can change the job CCSID by using the Change Job (CHGJOB) command. However, this solution is only for the job you are currently working with. The next time you will have to change the job CCSID again.

A more permanent solution is to change the CCSID designated by the user profiles used in the distributed relational database. When you change the user profiles you affect only those users that need to have their data converted. If you are working with an AS/400 AS, you need to change the user profile that the AS uses.

The default CCSID value in a user profile is *SYSVAL. This references the QCCSID system value. You can change this system value, and therefore the

default value used by all user profiles, with the Change System Values (CHGSYSVAL) command. If you do this, you would want to select a CCSID that represents most (if not all) of the users on your system. For a list of CCSIDs available and the languages they represent, see the *National Language Support Planning Guide*.

If you suspect that you are working with a system that does not support a CCSID used by your job or your system, look for the following indicators in a job log or SQLCA:

Message SQ30073
SQLCODE -30073
SQLSTATE 58017
Text Distributed Data Management (DDM) parameter X'0035' not supported.

Message SQL0332
SQLCODE -332
SQLSTATE 57017
Text Total conversion between CCSID &1 and CCSID &2 not valid.

CCSID Conversion Considerations for DDCS Connections

When you connect an OS/2 AR to an AS/400 AS, columns tagged with CCSID 65535 are not converted from EBCDIC to ASCII. If the files that contain these columns do not contain any columns that have a CCSID explicitly identified, the CCSID of all character columns can be changed to another CCSID value. To change the CCSID, use the Change Physical File (CHGPF) command. If you have logical files built over the physical file, follow the directions given in the recovery section of the error message (CPD322D) that you get.

CCSID Conversion Considerations for DB2 and SQL/DS Database Managers

One of the differences between the AS/400 database and a DB2* database is that the AS/400 system supports a larger set of CCSIDs. This can lead to errors when the systems attempt to perform character conversion on the data (SQLCODE –332 and SQLSTATE 57017).

Certain fields in the AS/400 SQL catalog tables are defined to have a DBCS-open data type. This is a data type that allows both double-byte character set (DBCS) and single-byte character set (SBCS) characters. The CCSID for these field types is based on the default CCSID shipped with the system.

When these fields are selected from a DB2 or SQL/DS AR, the SELECT statement may fail because the DB2 and SQL/DS databases may not support the conversion to this CCSID.

To avoid this error, you must change the DB2 database or the SQL/DS AR to run with either:

- The same mixed-byte CCSID as the DBCS-OPEN fields in the AS/400 SQL catalog tables.
- A CCSID that the system allows conversion of data to when the data is from the mixed-byte CCSID of the DBCS-OPEN fields in the AS/400 SQL catalog

tables. This CCSID may be a single-byte CCSID if the data in the AS/400 SQL catalog tables DBCS-OPEN fields is all single-byte data.

This requires some analysis of the CCSID conversions supported on the DB2 or SQL/DS system so you can make the correct changes to your system. See the *IBM DATABASE 2 Administration Guide* for specific information on how to handle this error.

Interactive SQL and Query Management Setup on Unlike Application Servers

Interactive SQL and Query Manager/400 create packages on unlike application servers based on the user's run options (date format, commitment control, and so on) as they are needed. These packages are created in a collection called QSQL400 on the application server. The package name is QSQLabcd where 'abcd' correspond to numbers which refer to specific options that are used for that package. Values for 'abcd' correspond to options as follows :

Position	Option	Value
a	Date Format	0 = ISO, JIS date format 1 = USA date format 2 = EUR date format
b	Time Format	0 = JIS time format 1 = USA time format 2 = EUR, ISO time format
c	Commitment Control Decimal Delimiter	0 = *CS commitment control period decimal delimiter 1 = *CS commitment control comma decimal delimiter 2 = *RR commitment control period decimal delimiter 3 = *RR commitment control comma decimal delimiter
d	String Delimiter Default Character Subtype	0 = apostrophe string delimiter, single byte character subtype 1 = apostrophe string delimiter, double byte character subtype 2 = double quote string delimiter, single byte character subtype 3 = double quote string delimiter, double byte character subtype

For example, a package created from interactive SQL to an unlike application server with the following options: USA date format, USA time format, commitment control level of *CS, a period for the decimal delimiter, an apostrophe for the string delimiter, and a default character subtype of single byte would have the name 'QSQL1100'. Once a package is created with a particular set of options, all subsequent interactive SQL or Query Manager/400 users running with those same options against that application server will use that package.

Creating Interactive SQL Packages on SQL/DS

On SQL/DS, a collection name is synonymous with a user ID. To create packages to be used with interactive SQL or Query Manager/400 on an SQL/DS application server, create a user ID of QSQL400 on the OS/400 system. This user ID can be used to create all the necessary packages on the SQL/DS application server. Users can then use their own user IDs to access SQL/DS through interactive SQL or Query Manager/400 on the OS/400.

Accessing AS/400 Data through DDCS

This section answers the following frequently asked questions from users of work stations who want to access AS/400 data through a DDCS gateway:

- Do AS/400 files have to be journaled?
- When will query data be blocked for better performance?
- Is the SQL/400 product required on an AS/400 system to create collections and tables?
- How do you interpret an SQLCODE and the associated tokens reported in a DBM SQL0969N error message?
- How can host variable type in WHERE clauses affect performance? (See "Performance Effect of Host Variables in WHERE Clauses" on page 8-21 for this information.)
- What considerations must be given to CCSIDs? (See "CCSID Considerations" on page B-1.)
- Why are no rows returned when I perform a query? One potential cause of this problem is a failure to add an entry for the AS/400 in the DDCS Database Communication Services Directory.

Do AS/400 Files Have to Be Journaled?

The answer to this question is closely related to the question in "When Will Query Data Be Blocked for Better Performance?." Journaling is not required if the client application is using an isolation level of uncommitted read (UR) and if the OS/400 SQL function determines that the query data can be blocked. In that case commitment control is not enabled, which makes journaling unnecessary.

The OS/2 database manager precompiler parameter that specifies uncommitted read is /I=UR. When using the interactive OS/2 command line interface, the command DBM CHANGE SQLISL TO UR sets the isolation level to uncommitted read.

When Will Query Data Be Blocked for Better Performance?

The query data will be blocked if none of the following conditions are true:

- The cursor is updatable (see Note 1).
- The cursor is potentially updatable (see Note 2).
- The /K=NO precompile or bind option was used on SQLPREP or SQLBIND.

Basically what these rules (including the notes) say is that if you have a program that does not include dynamic statements and you bind it with the default /K precompile/bind option of UNAMBIG (single-row protocol), blocking will occur if:

- The cursor is read-only (see note 3), or,
- There is no FOR UPDATE OF clause in the SELECT, and,
- There are no UPDATE or DELETE WHERE CURRENT OF statements against the cursor in the program.

Furthermore, if you have dynamic statements in the program, but use the /K=ALL option to bind the program, data is still blocked for cursors that satisfy the other conditions (i.e., they are read-only, or they have no FOR UPDATE OF clause in the

SELECT and there are no UPDATE or DELETE WHERE CURRENT OF statements against the cursor in the program).

Notes:

1. A cursor is updatable if it is not read-only (see Note 3), and one of the following is true:
   - The select statement contained the FOR UPDATE OF clause, or
   - There exists in the program an UPDATE or DELETE WHERE CURRENT OF against the cursor.
2. A cursor is potentially updatable if it is not read-only (see Note 3), and if the program includes any dynamic statement, and the /K=UNAMBIG precompile or bind option was used on SQLPREP or SQLBIND.
3. A cursor is read-only if one or more of the following conditions is true:
   - The DECLARE CURSOR statement specified an ORDER BY clause but did not specify a FOR UPDATE OF clause.
   - The DECLARE CURSOR statement specified a FOR FETCH ONLY clause.
   - One or more of the following conditions are true for the cursor or a view or logical file referenced in the outer subselect to which the cursor refers:
     - The outer subselect contains a DISTINCT keyword, GROUP BY clause, HAVING clause, or a column function in the outer subselect.
     - The select contains a join function.
     - The select contains a UNION operator.
     - The select contains a subquery that refers to the same table as the table of the outer-most subselect.
     - The select contains a complex logical file that had to be copied to a temporary file.
     - All of the selected columns are expressions, scalar functions, or constants.
     - All of the columns of a referenced logical file are input only.

Is the SQL/400 Product Needed for Collection and Table Creation?

Working locally on an AS/400 system, it is possible to create an SQL collection without having the SQL/400 product installed. For example, to create the NULLID collection needed for part of the DDCS installation you can:

1. Create a source file member containing the line:

   ```
   CREATE COLLECTION NULLID
   ```

2. Perform a CRTQMQRY command referencing the above source file member.
3. Execute the CREATE statement using the STRQMQRY command.

It is also possible to create tables and execute other SQL statements with the above approach as well. A REXX or Control Language program can improve the usability of this approach. The following CL program is a simple example of the type of thing that can be done.

```
PGM
MONMSG   MSGID(CPF0000)
DLTQMQRY MYLIB/QMTEMP
STRSEU   MYLIB/SRC QMTEMP
CRTQMQRY MYLIB/QMTEMP MYLIB/SRC
STRQMQRY MYLIB/QMTEMP
ENDPGM
```

When the SEU program involved in the preceding series of commands displays an edit screen, enter an SQL statement and save the file. The program then attempts to process and execute the statement.

How Do You Interpret an SQLCODE and the Associated Tokens Reported in a DBM SQL0969N Error Message?

The client support used with DDCS returns message SQL0969N when reporting host SQLCODEs and tokens for which it has no equivalent code. The following is an example of message SQL0969N:

```
SQL0969N  There is no message text corresponding to SQL error
"-7008" in the Database Manager message file on this workstation.
The error was returned from module "QSQOPEN" with original
tokens "TABLE1    PRODLIB1    3".
```

Use the AS/400 DSPMSGD command to interpret the code and tokens:

```
DSPMSGD SQL7008 MSGF(QSQLMSG)
```

Select option 1 (Display message text) and the system presents the Display Formatted Message Text display. The three tokens in the message are represented by &1, &2, and &3 in the display. The reason code in the example message is 3, which points to Code 3 in the list at the bottom of the display.

```
                        Display Formatted Message Text
                                                              System:   RCHASLAI
Message ID . . . . . . . . . :   SQL7008
Message file . . . . . . . . :   QSQLMSG
  Library  . . . . . . . . . :     QSYS

Message . . . . :    &1 in &2 not valid for operation.
Cause . . . . . :    The reason code is &3.  A list of reason codes follows:
    -- Code 1 indicates that the table has no members.
    -- Code 2 indicates that the table has been saved with storage free.
    -- Code 3 indicates that the table is not journaled, the table is
  journaled to a different journal than other tables being processed under
  commitment control, or that you do not have authority to the journal.
    -- Code 4 indicates that the table is in a production library but the user
  is in debug mode with UPDPROD(*NO); therefore, production tables may not be
  updated.
    -- Code 5 indicates that a table, view, or index is being created into a
  production library but the user is in debug mode with UPDPROD(*NO);
  therefore, tables, views, or indexes may not be created.
                                                                        More...
Press Enter to Continue.

F3=Exit   F11=Display unformatted message text    F12=Cancel
```

Appendix C. Interpreting Trace Job and FFDC Data

This appendix provides additional problem-analysis information. It is useful to specialists responsible for problem determination. It is also for suppliers of software products designed to conform to the Distributed Relational Database Architecture who want to test connectivity to an AS/400 system.

This appendix contains an example of the RW component trace data from a job trace with an explanation of the trace data output. Some of this information is helpful with interpreting communications trace data. This appendix also shows an example of a first-failure data capture printout of storage, with explanations of the output.

Interpreting Data Entries for the RW Component of Trace Job

It is the RW component of the OS/400 licensed program that includes most of the DRDA support. This component produces certain types of diagnostic information when the Trace Job (TRCJOB) command is issued with TRCTYPE(*ALL) or TRCTYPE(*DATA). RW trace points are of the type that are shown in Figure C-1. RW trace points can be located easily by doing a find operation using the string '>>' as the search argument. The end of the data dumped at each trace point can be determined by looking for the '<<<...' delimiter characters. There are one or more of the '<' delimiter characters at the end of the data, enough to fill out the last line.

```
DATA   FF  6E6ED9E6D8E840D9C37A0016D0520001001022050006114900000  *>>RWQY RC:  }             *
DATA   FF  00062102241700250D0530001001F241A0C76D005000231000030A  *        }         }       *
DATA   FF  00080971E0540001D000010671F0E00000002CD0530001002624   *    \   }   0\    }        *
DATA   FF  1BFF0000000100F1F1F1411000000000000FF0000000200F2F2    *       111              22 *
DATA   FF  F241200000000000000026D0520001002022OB00061149000400   *2         }              *
DATA   FF  162110C4C2F2C5E2E8E2404040404040404040404000056D00300  *   DB2ESYS            }   *
DATA   FF  0100502408000000064F0F2F0F0F0C4E2D5E7D9C6D54000C4C2    *  &      02000DSNXRFN  DB *
DATA   FF  F2C5E2E8E2404040404040404040404040FFFFFF92000000000000 *2ESYS              k       *
DATA   FF  0000FFFFFFFF000000000000000040404040404040404040404000 *                          *
DATA   FF  0000004C4C4C4C4C4C4C4C4C4C4C4C4C4C4C4C4C4C4C4C4C4C4C4C  *   <<<<<<<<<<<<<<<<<<<<<<< *
```

Figure C-1. An Example of Job Trace RW Component Information

Note: There is an exception to the use of the '<' delimiters to determine the end of data. In certain rare circumstances where a received data stream is being dumped, the module that writes the trace data is unable to determine where the end of the data stream is. In that case, the program dumps the entire receive buffer, and as a warning that the length of the data dumped is greater than that of the data stream, it replaces the '<<<...' delimiter with a string of '(' characters.

Following the '>>' prefix is a 7-character string that identifies the trace point. The first 2 characters, 'RW', identify the component. The second 2 characters identify the RW function being performed. The 'QY' indicates the query function which corresponds to the DDM commands OPNQRY, CNTQRY, and CLSQRY. The 'EX' indicates the EXECUTE function which corresponds to the DDM commands EXCSQLSTT, EXCSQLIMM, and PRPSQLSTT.

Which program module corresponds to each of these functions depends on whether the job trace was taken at the application requester (AR) end of the distributed SQL access operation, or at the application server (AS) end. The modules performing the process and query functions at the AR are QRWSEXEC and QRWSQRY. The modules at the AS are QRWTEXEC and QRWTQRY.

The last 2 characters of the 7-byte trace point identifier indicate the nature of the dumped data or the point at which the dump is taken. For example, SN corresponds to the data stream sent from an AR or an AS, and RC corresponds to the data stream received by an AR.

Analyzing the RW Trace Data Example

The example in Figure C-1 on page C-1 shows the data stream received during a distributed SQL query function. This particular trace was run at the AR end of the connection. Therefore, the associated program module that produced the data is QRWSQRY.

The following discussion examines the elements that make up the data stream in the example. For more information on the interpretation of DRDA data steams, see the *Distributed Relational Database Architecture Reference* manual and the *Distributed Data Management Level 3.0 Architecture Reference* manual.

The trace data follows the ':' marking the end of the trace point identifier. In this example, the first 6 bytes of the data stream contain the DDM data stream structure (DSS) header. The first 2 bytes of this DSS header are a length field. The third byte, X'D0' is the registered SNA architecture identifier for all DDM data. The fourth byte is the format identifier (explained in more detail later). The fifth and sixth bytes contain the DDM request correlation identifier.

The next 2 bytes, X'0010' (decimal 16) give the length of the next DDM object, which in this case is identified by the X'2205' which follows it and is the code point for the OPNQRYRM reply message.

Following the 16-byte reply message is a 6-byte DSS header for the reply objects that follow the reply message. The first reply object is identified by the X'241A' code point. It is a QRYDSC object. The second reply object in the example is a QRYDTA structure identified by the X'241B' code point (split between two lines in the trace output). As with the OPNQRYRM code point, the preceding 2 bytes give the length of the object.

Looking more closely at the QRYDTA object, you can see a X'FF' following the X'241B' code point. This represents a null SQLCAGRP (the form of an SQLCA that flows on the wire). The null form of the SQLCAGRP indicates that it contains no error or warning information about the associated data. In this case, the associated data is the row of data from an SQL SELECT operation. It follows the null SQLCAGRP. Because rows of data as well as SQLCAGRPs are nullable, however, the first byte that follows the null SQLCAGRP is an indicator containing X'00' that indicates that the row of data is not null. The meaning of the null indicator byte is determined by the first bit. A '1' in this position indicates 'null'. However, all 8 bits are usually set on when an indicator represents a null object.

The format of the row of data is indicated by the preceding QRYDSC object. In this case, the QRYDSC indicates that the row contains a nullable SMALLINT value, a nullable CHAR(3) value, and a non-nullable double precision floating point value.

The second byte past the null SQLCAGRP is the null indicator associated with the SMALLINT field. It indicates the field is not null, and the X'0001' following it is the field data. The nullable CHAR(3) that follows is present and contains '111'. The floating point value that follows next does not have a X'00' byte following it, since it is defined to be not nullable.

A second row of data with a null SQLCAGRP follows the first, which in turn is followed by another 6-byte DSS header. The second half of the format byte (X'2') contained in that header indicates that the corresponding DSS is a REPLY. The format byte of the previous DSS (X'53') indicated that it was an OBJECT DSS. The ENDQRYRM reply message carried by the third DSS requires that it be contained in a REPLY DSS. The ENDQRYRM code point is X'220B'. This reply message contains a severity code of X'0004', and the name of the RDB that returned the query data ('DB2ESYS').

Following the third DSS in this example is a fourth and final one. The format byte of it is X'03'. The 3 indicates that it is an OBJECT DSS, and the 0 that precedes it indicates that it is the last DSS of the chain (the chaining bits are turned off).

The object in this DSS is an SQLCARD containing a non-null SQLCAGRP. The first byte following the X'2408' SQLCARD code point is the indicator telling us that the SQLCAGRP is not null. The next 4 bytes, X'00000064', represents the +100 SQLCODE which means that the query was ended by the encounter of a 'row not found' condition. The rest of the fields correspond to other fields in an SQLCA. The mapping of SQLCAGRP fields to SQLCA fields can be found in the *Distributed Relational Database Architecture Reference* manual.

Description of RW Trace Points

RWxx RC—Receive Data Stream Trace Point: This data stream contains a DDM response from an AS program. The DSS headers are present in this data stream. This is the trace point shown in the above example.

RWxx SN—Send Data Stream Trace Point: This data stream contains either a DDM request from an AR program, or a DDM response from an AS program, as they exist before they are given to the lower level CN component for addition of headers and transmission across the wire. Besides content, the main difference between the trace information for receive data streams and send data streams is that for the latter, the 6-byte DSS header information is missing. For the first DSS in a send data stream trace area, the header is omitted entirely, and for subsequent ones, 6 bytes of zeros are present which will be overlaid by the header when it is constructed later by a CN component module.

RWQY S1—Partial Send Data Stream Trace Point 1: This trace point occurs in the NEWBLOCK routine of the QRWTQRY module, when a new query block is needed in the building of QRYDTA in the like environment. In the like environment a query block need not be filled up before it is transmitted, and it is always put on the wire at this point so that the buffer space can be reused. DSS headers are absent as in other send data streams.

RWQY S2—Partial Send Data Stream Trace Point 2: This trace point occurs in the NEWBLOCK routine of the QRWTQRY module, when a new query block is needed in the building of QRYDTA in the unlike environment. In the unlike environment all query blocks except the last one must be filled up before construction of a new one can be started, and they are not transmitted until all are built.

RWQY BP—Successful Fetch Trace Point:

This trace point occurs in the FETCH routine of the QRWTQRY module, when a call to the SQFCHCRS macro results in a non-null pointer to a BPCA structure, implying that one or more records were returned in the BPCA buffer. The data dumped is the BPCA structure (not the associated buffer), which among other things indicates how many records were returned.

RWQY NB—Unsuccessful Fetch Trace Point: This trace point occurs in the FETCH routine of the QRWTQRY module, when a call to the SQFCHCRS macro results in a null pointer to a BPCA structure, implying that no records were returned in the BPCA buffer. The data dumped is the SQLSTATE in the associated SQLCA area.

First-Failure Data Capture (FFDC)

The AS/400 system provides a way for you to capture and report error information for the distributed relational database. This function is called **first-failure data capture (FFDC)**. The primary purpose of FFDC support is to provide extensive information on errors detected in the DDM components of the OS/400 system from which an Authorized Program Analysis Report (APAR) can be created.

You can also use this function to help you diagnose some system-related application problems. By means of this function, key structures and the DDM data stream are automatically dumped to the spooled file. This automatic dumping of error information on the first occurrence of an error means that you do not have to create the failure again to report it for service support. FFDC is active in both the application requester and the application server.

One thing you should keep in mind is that not all negative SQLCODEs result in dumps; only those that may indicate an APAR situation are dumped.

An FFDC Dump

The processing of alerts triggers FFDC data to be dumped. However, the FFDC data is produced even when alerts or alert logging is disabled (using the CHGNETA command). FFDC output can be disabled by setting the QSFWERRLOG system value to *NOLOG, but it is strongly recommended that you do not disable the FFDC dump process. If an FFDC dump has occurred, the informational message, "*Software problem detected in Qxxxxxxx." (where Qxxxxxxx is an OS/400 module identifier), is logged in the QSYSOPR message queue.

To see output from an FFDC dump operation, use the Work with Spooled Files (WRKSPLF) command and view QPSRVDMP. The information contained in the dump output are:

- DDM function
- Specific information on the failing DDM module
- DDM source or target main control block
- DDM internal control structures
- DDM communication control blocks
- Input and output parameter list for the failing DDM module if at the application requester
- The request and reply data stream

The first 1K-byte of data is put in the error log. However, the data put in the spooled file is always complete and easier to work with. If multiple DDM conversations have been established, the dump output may be contained in more than one spooled file because of a limit of only 32 entries per spooled file. In this case, there will be multiple "Software Problem" messages in the QSYSOPR message queue that are prefixed with an asterisk (*).

```
 Work With Error Log                  02/27/91 13:33:05  Page . . . :       1
   A                                                      B
5738SS1 V2R1M1                 AS/400 DUMP               090454/SRR/SRRS1        02/27/91 15:12:52        PAGE   1
DUMP TAKEN FOR DETECTED ERROR
                                C
.SUSPECTED-                    QRWSQRY               LIBRARY-         S
..LICENSED PROGRAM-            5738SS1    V2R1M1
..FUNCTION-                    5001
..LOAD-                        0000
..PTF-
                                D
.DETECTOR-                     QRWSQRY               LIBRARY-         S
..LICENSED PROGRAM-            5738SS1    V2R1M1
..FUNCTION-                    5001
..LOAD-                        0000
..PTF-
.SYMPTOM STRING-
|          E     F       G
|  5738 MSGCPF3E86 F/QRWSQRY RC10000002
|                          H
| .SPACE-                        01                                                                  I
  000000   F0F17EC9 D5C4E740 F0F27EC6 C3E34E40   F0F37EC5 D4E2C740 F0F47ED7 D9D4E240  *01=INDX 02=FCT+ 03=EMSG 04=PRMS *
  000020   F0F57EE2 D5C4C240 F0F67ED9 C3E5C240   F0F77EC1 D9C4C240 F0F87ED8 C4E3C140  *05=SNDB 06=RCVB 07=ARDB 08=QDTA *
  000040   F0F97EC9 D5C4C140 F1F07EE2 D8C3C140   F1F17EE6 D9C3C140 F1F27ED9 C6D4E340  *09=INDA 10=SQCA 11=WRCA 12=RFMT *
  000060   F1F37EC1 C6E34040 F1F47EE2 D4C3C240   F1F57EE3 E2D3D240 F1F67EE5 C1D9E240  *13=AFT  14=SMCB 15=TSLK 16=VARS *
  000080   4DD9C5E2 E340C9E2 40C3C3C2 6BD7C3C2   E26BE2C1 E36BD7D4 C1D76BD9 C3E5C240  *(REST IS CCB,PCBS,SAT,PMAP,RCVB *
  0000A0   D7C5D940 C3C3C25D                                                         *PER CCB)                        *
| .SPACE-                  L     02
  000000   200C1254 0102F5F8 F0F0F9                                                  *      58009                     *
| .SPACE-                        04                                                      J
  000000   D8D7C1D9 D4E20000 D67FC01D A60065A0   00000000 F0F10000 00000434 00000000  *QPARMS  O"          01          *
  000020   D9C3C8C1 E2F2F6F6 40404040 40404040   4040E2D9 D9404040 40404040 40404040  *RCHAS266          SRR           *
  000040   40404040 D7E3F140 40404040 40404040   40404040 4040700F 70DB33C0 00BB0005  *    PT1                         *
 .SPACE-                        05
  000000   00000000 0056D051 00010050 200C0044   2113D9C3 C8C1E2F2 F6F64040 40404040  *          &      RCHAS266      *
  000020   40404040 E2D9D940 40404040 40404040   40404040 4040D7E3 F1404040 40404040  *    SRR               PT1       *
  000040   40404040 40404040 700F70DB 33C000BB   00050008 21140000 7FFF0021 D0030001  *                              *
  000060   001B2412 00100010 0676D004 00000671   E4D00001 0007147A 000002            *                              *
 .SPACE-                        06
  000000   0016D052 00010010 22050006 11490000   00062102 24170052 D0530001 0022241A  *                              *
  000020   0F76D004 00002600 03020000 0A000009   71E05400 01D00001 0671F0E0 0000002A  *                              *
  000040   241BFF00 0001F0F0 F1000000 013FF000   00000000 00FF0000 02F0F0F2 00000002  *      001     0          002    *
  000060   40000000 00000000 0010D052 0001000A   220B0006 11490004 0069D003 00010063  *                              *
  0000E0   FF                                                                        *                              *
| .SPACE-                  K     07
  000000   D9C3C8C1 E2F2F6F6 40404040 40404040   4040D9C3 C8C1E2F2 F6F64040 40404040  *RCHAS266          RCHAS266      *
  000020   40404040 E2D9D940 40404040 40404040   40404040 4040D7E3 F1404040 40404040  *    SRR               PT1       *
  000040   40404040 40404040 700F70DB 33C000BB   D8E3C4E2 D8D3F4F0 F0D8E2D8 F0F2F0F1  *                QTDSQL400QSQ0201*
  000060   F1002500 00000000 25000000 000010F0   F4F5F1F7 F461E2D9 D961C4E2 F3F7F840  *1               045174/SRR/DS378 *
  000080   40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040  *                              *
        LINES  0000A0    TO    00015F  SAME AS ABOVE
  000160   40404040 40404040 40404040 4040A000   2434E2D9 D9404040 40404040 00000000  *                  SRR         *
  000180   C1D7D7D5 4BD9C3C8 C1E2F3F7 F8A7CCA7   54137200 40404000 00000000 00000000  *APPN.RCHAS378x x           *
  0001A0   00000000 00000000                                                         *                              *
 .SPACE-                        09
  000000   E2D8D3C4 C1404040 00000060 00010001   01F40002 00000400 00000040 40404040  *SQLDA            4            *
  000020   80000000 00000000 007FC01E 11000334   00000000 00000000 00000000 00000000  *                              *
  000040   00080000 00250000 00000000 00000000   00000000 00000000 00000000 00000000  *                              *
 .SPACE-                        10
  000000   E2D8D3C3 C1404040 00000088 FFFF8ABC   00041254 01020000 00000000 00000000  *SQLCA                          *
  000020   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                              *
  000040   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                              *
  000060   00000000 00000000 00000000 00000000   00000000 00000000 40404040 40404040  *                              *
  000080   404040F5 F8F0F0F9                                                         *   58009                      *
 .SPACE-                        11
  000000   E2D8D3C3 C1404048 00000088 00000000   00000000 00000000 00000000 00000000  *SQLCA                          *
  000020   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                              *
  000040   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                              *
  000060   00000000 00000000 00000000 00000000   00000000 00000000 40404040 40404040  *                              *
  000080   404040F0 F0F0F0F0                                                         *   00000                      *
```

```
.SPACE-                          13
  000000   00001BB0 00310001 F0F0F0F0 F0F0F0F0   00000000 00000000 00000000 00000000  *        00000000                *
  000020   00000470 000002C0 7023C382 57000048   80000000 00000000 007FA083 A3000820  *                                *
  000040   80000000 00000000 007FA083 E7000100   D9C3C8C1 E2F2F6F6 40404040 40404040  *                RCHAS266        *
  000060   40405CD3 D6C34040 40404040 5CD5C5E3   C1E3D940 D9C3C8C1 E2F2F6F6 5CD3D6C3  *  *LOC      *NETATR RCHAS266*LOC*
        LINES  0000A0    TO    001B9F  SAME AS ABOVE
  001BA0   00000000 00000000 00000000 00000000                                      *                                *
.SPACE-                          14
  000000   E2D4C3C2 20000100 00000010 F0F9F0F4   F5F461E2 D9D961E2 D9D9E2F1 00000000  *SMCB        090454/SRR/SRRS1    *
  000020   00000000 00000000 E5F0F2D9 F0F1D4F0   F1D9C3C8 C1E2F3F7 F8000000 00800000  *        V02R01M01RCHAS378       *
  000040   0302C3D5 E2E2D5D9 C3E5D8D3 F7F9F7F1   80000000 00000000 007FA083 E9000106  *  CNSSNRCVQL7971                *
  000060   F1000000 00710000 00000000 00000000   00000470 000002C0 7023C382 57000048  *1                               *
.SPACE-                          15
  000000   00000000 00000000 007FA083 E60019FF   00000000 00000000 00000000 00000000  *                                *
  000020   00000000 00400000                                                         *                                *
.SPACE-                          16
  000000   00000000 00000000 00000000 00000002   00000017 000000E1 00000000 00000071  *                                *
  000020   00000000 00007FFF 00000003 00170000   001B0000 FF000000 00002410 00F0F060  *                                *
  000040   E70400                                                                    *X                               *
.SPACE-                          17
  000000   E2C3C3C2 5CD3D6C3 40404040 40405CD5   C5E3C1E3 D9405CD3 D6C34040 4040D9C3  *SCCB*LOC      *NETATR *LOC    RC*
  000020   C8C1E2F2 F6F65CD3 D6C34040 404007F6   C4C24040 40405CC4 D9C4C140 40404040  *HAS266*LOC     6DB    *DRDA     *
  000040   40404040 40404040 4000001E 00110000   00000000 00000000 00000000 00000000  *                                *
  000060   00000000 00000000 00000000 00000000                                      *                                *
.SPACE-                          18
  000000   E2D7C3C2 00000000 007FA083 A3000810   00000470 000002C0 7023C382 57000048  *SPCB                            *
  000020   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                                *
  000040   00000000 00000000 00000000 00000000                                      *                                *
.SPACE-                          19
  000000   C5E7C3C2 00000076 00000003 00000079   00000009 00000082 00000010 00000092  *EXCB                            *
  000020   00000008 00000000 00000018 00200003   00030003 00030003 00030001 00030003  *                                *
  000040   00000000 00000000 00000000 00000000   00000000 0000C4C4 D4E5F0F2 D9F0F1D4  *                      DDMV02R01M*
  000060   F0F1F0F4 F5F1F7F4 61E2D9D9 61C4E2F3   F7F8D9C3 C8C1E2F2 F6F6              *01045174/SRR/DS378RCHAS266      *
.SPACE-                          20
  000000   00000030 000002B6 00000430 0000043E   00010000 00000000 00000000 00000000  *                                *
  000020   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                                *
  000040   80000000 00000000 007FA083 D2000100   00000000 0000029A 0000005C 22050000  *                                *
  000060   00060000 02B60000 00B00000 00000000   00000000 00000000 00000000 00000000  *                                *
        LINES  0000E0    TO    00017F  SAME AS ABOVE
.SPACE-                          21
  000000   0016D052 00010010 22050006 11490000   00062102 24170052 D0530001 0022241A  *                                *
  000020   0F76D004 00002600 03020000 0A000009   71E05400 01D00001 0671F0E0 0000002A  *                                *
  000040   241BFF00 0001F0F0 F1000000 013FF000   00000000 00FF0000 02F0F0F2 00000002  *      001     0          002     *
  000060   40000000 00000000 0010D052 0001000A   220B0006 11490004 0069D003 00010063  *                                *
  000080   24080000 000064F0 F2F0F0F0 D8E2D8C6   C5E3C3C8 00D9C3C8 C1E2F2F6 F6404040  *       02000QSQFETCH RCHAS266   *
.SPACE-                          22
  000000   E2C3C3C2 5CD3D6C3 40404040 40405CD5   C5E3C1E3 D9405CD3 D6C34040 4040D9C3  *SCCB*LOC      *NETATR *LOC    RC*
  000020   C8C1E2F2 F6F65CD3 D6C34040 404007F0   F0F14040 4040E77D F0F7C6F0 C6F0C6F1  *HAS266*LOC     001    X'07F0F0F1*
  000040   7D404040 40404040 40000014 00110000   00000000 00000000 00000000 00000000  *                                *
  000060   00000000 00000000 00000000 00000000   00008F00 00000700 F0F0F100 00000000  *                          001     *
.SPACE-                          23
  000000   C5E7C3C2 00000076 00000003 00000079   00000009 00000082 00000010 00000092  *EXCB                    b       k*
  000020   00000008 00000000 00000018 00200003   00030003 00030003 00030001 00030003  *                                *
  000040   00000000 00000000 00000000 00000000   00000000 0000C4C4 D4E5F0F2 D9F0F1D4  *                      DDMV02R01M*
  000060   F0F1F0F4 F5F1F7F2 61E2D9D9 61C4E2F3   F7F8D9C3 C8C1E2F2 F6F6              *01045172/SRR/DS378RCHAS266      *
.SPACE-                          24
  000000   00000030 0000005C 00000000 000000CC   00010000 00000000 00000000 00000000  *       *                        *
  000020   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *                                *
  000040   80000000 00000000 007FA083 A4000100   00000000 00000000 0000005C D2010000  *                            *K   *
.SPACE-                          25
  000000   0010D002 0001000A D2010006 11490000   E2000D11 5AE5F0F2 D9F0F1D4 F0F1000C  *                    V02R01M01   *
  000020   116DD9C3 C8C1E2F2 F6F60014 115EF0F4   F5F1F7F2 61E2D9D9 61C4E2F3 F7F80064  *  RCHAS266    045172/SRR/DS378  *
  000040   14041403 00031423 00031405 00031406   00031407 00031444 00031458 00011457  *                                *
  000060   0003140C 00031419 0003141E 00031422   0003240F 000314A0 00041432 00031433  *
END OF DUMP
                                        ***** END OF LISTING *****
```

FFDC Dump Output Description

The following information describes the data areas and types of information available in an FFDC dump output like the one in the preceding figure.

Notes:

1. Each FFDC dump output will differ in content, but the format is generally the same. An index (I) is provided to help you understand the content and location of each section of data.
2. Each section of data is identified by "SPACE-" and a number; for example: SPACE- ... 01. The sections of data present in your dump output are dependent on the operation and its progress at the time of failure.
3. Each section of data is given a name; for example SQCA. SQCA is the section name for data from the SQL/400 SQLCA. To locate the SQLCA data, find SQCA in the index (I). In the sample dump index, SQCA is shown to be in data section 10 (10=SQCA). To view the SQLCA data, go the SPACE- 10.
4. There are two basic classes of modules that can be dumped:
   - Application requester (AR) modules
   - Application server (AS) modules

   The sample dump output is typical of a dump from an AR module. AR dump outputs typically have a fixed number of data sections identified in the index. (For example, in the sample dump output SPACE- 01 through 16 are listed.) In addition, they have a variable number of other data sections. These sections are not included in the index. (For example, in the sample dump output, SPACE- 17 through 25 are not listed in the index.)

   Application server dump output is usually simpler because they consist only of a fixed number of data sections, all of which are identified in the index.
5. There are index entries for all data sections whether or not the data section actually exists in the current dump output. For example, in the sample dump output, there is no SPACE- 08. In the index, 08 equals QDTA (query data). The absence of SPACE- 08 means that no query data was returned, so none could be dumped.
6. In the sample dump output, the last entry in the index is "(REST IS CCB, PCBS, SAT, PMAP, RCVB, PER CCB)." This entry means that SPACE- 17 and upward contain one or more communications control blocks (CCB), each containing:
   - Zero, one, or more path control blocks (SPCB); there is normally just one.
   - Exchange server attributes control block (EXCB)
   - Parser map space
   - Receive buffer for the communications control block

   The data section number is incremented by one from 17 onward as each control block is dumped. For example, in the sample dump output, data sections SPACE- 17 through SPACE- 21 are for the first data control block dumped (CCB 1), while data sections SPACE- 22 through SPACE- 25 are for the second data control block dumped (CCB 2), as shown below:

   17 CCB (Eyecatcher is :'SCCB:'. For an application server module, the eyecatcher is :'TCCB:'.)

   18 PCB for CCB 1 (Eyecatcher is :'SPBC:'.)

19 SAT for CCB 1 (Eyecatcher is :'EXCB:'.)

20 PMAP for CCB 1 (No eyecatcher.)

21 RCVB for CCB 1 (No eyecatcher.)

22 CCB 2 (Eyecatcher is :'SCCB:'.)

-- (No PCB for CCB 2 because the conversation is not active.)

23 SAT for CCB 2 (Eyecatcher is :'EXCB:'.)

24 PMAP for CCB 2 (No eyecatcher.)

25 RCVB for CCB 2 (No eyecatcher.)

A Name and release information of the system on which the dump was taken.

B Name of job that created the dump output.

C Name of module in the operating system suspected of failure.

D Name of module that detected the failure.

Symptom String- contents:

E Message identifier.

F Name of module suspected of causing the FFDC dump.

G Return code (RC), identifying the point of failure.

The first digit after RC indicates the number of dump files associated with this failure. There can be multiple dump files depending on the number of conversations that were allocated. In the sample dump output, the digit is "1," indicating that this is the first (and possible the only) dump file associated with this failure.

You may have four digits (not zeros) at the rightmost end of the return code that indicate the type of error.

- The possible codes for errors detected by the AR are:

  0001 Failure occurred in connecting to the remote database

  0002 More-to-receive indicator was on when it should not have been

  0003 AR detected an unrecognized object in the data stream received from the AS

  0097 Error detected by the AR DDM communications manager

  0098 Conversation protocol error detected by the DDM component of the AR

  0099 Function check

- The possible codes for errors detected by the AS are:

  0099 Function check

  4415 Conversational protocol error

  4458 Agent permanent error

  4459 Resource limit reached

  4684 Data stream syntax not valid

  4688 Command not supported

4689 Parameter not supported

4690 Value not supported

4691 Object not supported

4692 Command check

8706 Query not open

8708 Remote database not accessed

8711 Remote database previously accessed

8713 Package bind process active

8714 FDO:CA descriptor is not valid

8717 Abnormal end of unit of work

8718 Data and/or descriptor does not match

8719 Query previously opened

8722 Open query failure

8730 Remote database not available

H SPACE- number identifying a section of data. The number is related to a data section name by the index. Data section names are defined under **I** below.

I An index and definition of SPACE- numbers (defined in **H**) to help you understand the content and location of each section of data. The order of the different data sections may vary between dump output from different modules. The meaning of the data section names are:

AFT: DDM active file table, containing all conversation information.

ARDB: Access remote database control block, containing the AR and AS connection information. The structure contains the LUWID token used to correlate the dump with any related alert data.

BDTA: Buffer processing communications area (BPCA) and associated data record from SELECT INTO statement.

BIND: SQL bind template

BPCA: BPCA structure (without data records)

DATA: Data records associated with the BPCA. It is possible that the records in this section do not reflect the total BPCA buffer contents. Already-processed records may not be included.

DOFF: Offset within query data stream (QRYDTA) where the error was detected.

EICB: Error information control block

EMSG: Error message associated with a function check or DDM communications manager error.

FCT: DDM function code point (2 bytes)

FCT+: Same as FCT, plus message tokens and the SQLSTATE logged in the SQLCA. See "DDM Error Codes" on page C-13 for more information on how to interpret FCT+.

- DDM function code point (2 bytes)
- DDM reply code point (2 bytes)
- DDM reply reason code (2 bytes)
  - Location at which the error was detected (1 byte; 01 = application requester; 02 = application server)
  - Error reason code (1 byte; see the DDM Error Codes on page C-13.)
- SQLSTATE (5 bytes)

FDOB: FDO:CA descriptor input to the parser in an execute operation.

FDTA: FDO:CA data structure consisting of:

- A 4-byte field defining the length of the FDO:CA data stream (FDODTA)
- The FDODTA

HDRS: Communications manager command header stack.

INDA: Input SQLDA containing user-defined SQLDA for insert, select, delete, update, open, and execute operations.

INDX: The index that maps the data section name to the data section SPACE- code. Not all of the entries in the index have a corresponding data section. The dump data is based on the error that occurs and the progress of the operation at the time of the error. A maximum of 32 entries can be dumped in one spooled file.

INST: SQL statement

ITKN: Interrupt token.

PKGN: Input package name, consistency token and section number.

PMAP: Parser map in an AS dump output.

PRMS: DDM module input or output parameter structure.

PSOP: Input parser options.

QDTA: Query data structure consisting of:

- A 4-byte field defining the length of the query data stream (QRYDTA)
- The QRYDTA

RCVB: Received data stream. The contents depend on the following:

- If the dump occurs on the application server, the section contains the DDM request data that was sent from the application requester.
- If the dump occurs on the application requester, the section contains the DDM reply data that was sent from the application server. If this section is not present, it is possible the received data may be found in the receive buffer in the variable part of the dump.

RDBD: Relational database directory.

RFMT: Record format structure.

RMTI: Remote location information in the commitment control block.

SMCB: DDM source master control block, containing pointers to other DDM connection control blocks and internal DDM control blocks.

SNDB: Send data stream. The contents depend on the following:

- If the dump occurs on the application requester, the buffer contains the DDM request that was sent to the application server or that was being prepared to send.

  Note the four bytes of zeros that are at the beginning of SPACE- 05 in the example. When zeros are present, they are not part of the data stream. They represent space in the buffer that is used only in the case that a DDM large object has to be sent with a DDM request. The DDM request stream is shifted left four bytes in that case.

- If the dump occurs on the application server, the buffer contains the DDM reply data that was being prepared to send to the application requester.

SQCA: Output SQLCA being returned to the user.

SQDA: SQLDA built by the FDO:CA parser.

TBNM: Input remote database table name.

TMCB: Target main control block.

TSLK: Target or source connection control block, containing pointers to the DDM active file table and other internal DDM control blocks.

VARS: Local variables for the module being dumped.

WRCA: Warning SQLCA returned only for an open operation (OPNQRYRM).

XSAT: Exchange server attributes control block.

Remainder: Multiple conversation control blocks for all the DDM conversations for the job at the time of the error. Each conversation control block contains the following:

- Path control blocks, containing information about an established conversation. There can be multiple path control blocks for one conversation control block.
- One exchange server information control block, containing information about the application requester and application server.
- One DDM parser map area, containing the locations and values for all the DDM commands, objects, and replies.
- One receive buffer, containing the requested data stream received by the application server. See also 6 on page C-8.

The data section number is incremented by one as each control block is dumped.

J The eyecatcher area. Information identifying the type of data in some of the areas that were dumped.

K The logical unit of work identifier (LUWID) for the conversation in progress at the time of the failure can be found in the access RDB control block. This data area is identified by the string 'ARDB' in the FFDC index. In this example, it is in SPACE- 07. The LUWID begins at offset 180. The network identifier (NETID) is APPC. A period separates it from the logical unit (LU)

name, RCHAS378, which follows. Following the LU name is the 6-byte LUW instance number X'A7CCA7541372'.

DDM Error Codes

These error codes are included in the FFDC dumps (**L** in the sample dump output) that identify DDM error conditions. These conditions may or may not be defined by the DDM architecture. The DDM error codes are divided into three categories:

- Command check codes
- Conversational protocol error code descriptions
- DDM syntax error code descriptions

Command Check Codes

If FCT+ (SPACE- 02) contains 1254 in bytes 3 and 4, look for one of these codes in byte 6:

01	Failure to connect to the relational database (RDB).
02	State of the DDM data stream is incorrect.
03	Unrecognized object in the data stream.
97	DDM communications manager detected an error.
98	Conversation protocol error detected by the DDM module.
99	Function check. Look for EMSG section, normally in SPACE- 03.

Conversational Protocol Error Code Descriptions

IF FCT+ (SPACE- 02) contains 1245 in bytes 3 and 4, look for one of these codes in byte 6:

01	RPYDSS received by target communications manager.
02	Multiple DSSs sent without chaining, or multiple DSS chains sent.
03	OBJDSS sent when not allowed.
04	Request correlation identifier of an RQSDSS is less than or equal to the previous RQSDSS request correlation identifier in the chain. If two RQSDSSs have the same request correlation identifier, the PRECCNVRM must be sent in RPYDSS with a request correlation identifier of minus 1.
05	Request correlation identifier of an OBJDSS does not equal the request correlation identifier of the preceding RQSDSS.
06	EXCSAT was not the first command after the connection was established.
E0	No OPNQRY (open query) reply message.
E1	RDBNAM on ENDQRYRM (end query reply message) is not valid.
E2	An OPEN got QRYDTA (query answer set data) without a QRYDSC (query answer set description).
E3	Unexpected OPNQRY RPY object.
E4	Unexpected CXXQRY RPY object.

E5 QRYDTA on OPEN, single row.

E6 RM after OPNQRYRM is not valid.

E7 No interrupt reply message.

FD Happy SQLCARD (SQLCA reply date) following error RM.

FE Null QRYDTA row after happy.

FF Expected SQLCARD missing.

DDM Syntax Error Code Descriptions

If FCT+ (SPACE- 02) contains 124C in bytes 3 and 4, look for one of these codes in byte 6:

01 DSS header length less than 6.

02 DSS header length does not match the number of bytes of data found.

03 DSS head C-byte not X'D0'.

04 DSS header F-bytes either not recognized or not supported.

05 DSS continuation specified, but not found. For example, DSS continuation is specified on the last DSS, and the SEND indicator has been returned by the SNA LU 6.2 communications program.

06 DSS chaining specified, but no DSS found. For example, DSS chaining is specified on the last DSS, and the SEND indicator has been returned by the SNA LU 6.2 communications program.

07 Object length less than 4. For example, a command parameter length is specified as 2, or a command length is specified as 3.

08 Object length does not match the number of bytes of data found. For example, an RQSDSS with a length 150 contains a command whose length is 125, or an SRVDGN (server diagnostic information) parameter specifies a length of 200, but there are only 50 bytes left in the DSS.

09 Object length greater than maximum allowed. For example, the RECCNT parameter specifies a length of 5, but this indicates that only half of the hours field is present instead of the complete hours field.

0A Object length less than minimum required. For example, the SVRCOD parameter specifies a length of 5, but the parameter is defined to have a fixed length of 6.

0B Object length not allowed. For example, the FILEXPDT parameter is specified with a length of 11, but this indicates that only half of the hours field is present instead of the complete hours field.

0C Incorrect large object extended length field (see the description of DSS). For example, an extended length field is present, but it is only 3 bytes long. It is defined as being a multiple of 2 bytes long.

0D Object code point index not supported. For example, a code point of X'8032' is encountered, but X'8' is a reserved code point index.

0E Required object not found. For example, a CLRFIL command does not have an FILNAM parameter present, or an MODREC command is not followed by a RECORD command data object.

0F	Too many command data objects sent. For example, an MODREC command is followed by two RECORD command data objects, or a DELREC command is followed by a RECORD object.
10	Mutually exclusive objects present. For example, a CRTDIRF command specifies both a DCLNAM and a FILNAM parameter.
11	Too few command data objects sent. For example, an INSRECEF command that specified RECCNT(5) is followed by only four RECORD command data objects.
12	Duplicate object present. For example, a LSTFAT command has two FILNAM parameters specified.
13	Specified request correlation identifier not valid. Use PRCCNVRM with a PRCCNVCD of X'04' or X'05' instead of this error code. This error code is being maintained for compatibility with Level 1 architecture.
14	Required value not found.
15	Reserved value not allowed. For example, an INSRECEF command specified an RECCNT(0) parameter.
16	DSS continuation less than or equal to 2. For example, the length bytes for the DSS continuation have a value of 1.
17	Objects not in required order. For example, a RECAL object contains a RECORD object followed by a RECNBR object that is not in the specified order.
18	DSS chaining bit not a binary 1, but DSSFMT bit 3 is set to a binary 1. is requested.
19	Previous DSS indicated current DSS has the same request correlation, but the request correlation identifiers are not the same.
1A	DSS chaining bit not a binary 1, but error continuation is requested.
1B	Mutually exclusive parameter values specified. For example, an OPEN command specified PRPSHD(TRUE) and FILSHR(READER).
1D	Code point not a valid command. For example, the first code point in RQSDSS either is not in the dictionary or is not a code point for a command.

Appendix D. DDM Architecture Command Support

The OS/400 licensed program supports all DDM commands and parameters that the Remote Unit of Work portion of the DRDA architecture requires. The following tables show how OS/400 supports each of these DDM codepoints and parameters.

The meanings of the columns for *commands* are:

The R column indicates how the OS/400 application requester handles the codepoint or parameter:

Y The OS/400 program flows it to application server.

N The OS/400 program does not flow it to the application server.

The S column indicates how the AS/400 application server supports the codepoint or parameter:

Y The OS/400 program recognizes and processes it.

S The OS/400 program allows the parameter, depending on its value. The supported values are defined after the table.

I The OS/400 program ignores it if received.

The meanings of the columns for *reply objects* (in the indented tables) are:

The S column indicates how the AS/400 application server handles the codepoint or parameter:

Y The OS/400 program flows it to the application requester.

N The OS/400 program does not flow it to the application requester.

The R column indicates how the AS/400 application requester supports the codepoint or parameter:

Y The OS/400 program recognizes and processes it.

I The OS/400 program ignores it.

Note that each DDM command can have associated with it:

- Parameters (instance variables)
- Command data objects
- Reply messages
- Reply data objects

In the following tables, commands and associated object types can be distinguished by the following means:

- Command and reply message names are in uppercase and contained in the top row of a table.
- Parameter names are in lowercase.
- Command object names are in uppercase and if present are in the bottom rows of a table.
- Reply data object names are in mixed case (first letter capitalized).

Figure D-1. ACCRDB Command

DDM Codepoint	Optional	R	S	AR Value
ACCRDB		Y	Y	
rdbnam (name of relational database)		Y	Y	
rdbacccl (access manager class)		Y	Y	
typdefnam (data type definition name)		Y	Y	QDTSQL400
typdefovr (data type definition override)		Y	Y	
prdid (product specific identifier)		Y	Y	QSQvvrrm
rdbalwupd (relational database to allow updates)	Y	N	Y	
prddta (product specific data)	Y	N	I	
sttstrdel (string delimiter)	Y	Y	Y	
sttdecdel (decimal delimiter)	Y	Y	Y	
crrtkn (correlation token)	Y	N	Y	
trgdftrt (target default value return)	Y	Y	Y	

Notes:

1. On the STTSTRDEL parameter, the application requester always sends DFTPKG to application server for dynamic SQL.
2. On the STTDECDEL parameter, the application requester always sends DFTPKG to application server for dynamic SQL.
3. For the PRDID parameter, the string 'vvrrm' contains the requester's version, release, and modification level.

Figure D-2. ACCRDBRM Reply for ACCRDB command

DDM Codepoint	Optional	R	S	AS Value
ACCRDBRM		Y	Y	
svrcod (severity code)		Y	Y	
prdid (product specific identifier)		Y	Y	QSQvvrrm
typdefnam (data type definition name)		Y	Y	QTDSQL400
typdefovr (data type definition override)		Y	Y	
srvdgn (server diagnostic information)	Y	I	N	
rdbinttkn (relational database interrupt token)	Y	Y	Y	
crrtkn (correlation token)	Y	Y	Y	
pkgdftcst (package default character subtype)	Y	Y	Y	
usrid (user id at the target system)	Y	Y	Y	

Note:

For the PRDID parameter, the string 'vvrrm' contains the application server's version, release, and modification level.

Figure D-3 (Page 1 of 2). BGNBND Command

DDM Codepoint	Optional	R	S	CRTSQLxxx Pre-compile Options
BGNBND		Y	Y	
rdbnam (name of relational database as in ACCRDB)	Y	Y	Y	RDB
pkgnamct (package name and consistency token) rdbnam (application server database name). rdbcolid (relational database collection identifier). pkgid (package identifier). pkgcnstkn (package consistency token).	N	Y	Y	RDB, SQLPKG, DFTRDBCOL
vrsnam (package version name)	Y	N	S	
pkgrplvrs (replaced package version name)	Y	N	S	
bndchkexs (bind existence checking)	Y	Y	Y	
pkgrplopt (package replacement option)	Y	Y	Y	REPLACE
pkgathopt (package authorization option)	Y	N	Y	
sttstrdel (statement string delimiter)	Y	Y	Y	OPTION – *QUOTE – *APOST
sttdecdel (statement decimal delimiter)	Y	Y	Y	OPTION – *SYSVAL – *PERIOD – *COMMA
sttdatfmt (date format of statement)	Y	Y	Y	DATFMT
stttimfmt (time format of statement)	Y	Y	Y	TIMFMT
pkgisolvl (package isolation level)	N	Y	Y	COMMIT
bndcrtctl (bind creation control)	Y	Y	Y	GENLVL
bndexpopt (bind explain option)	Y	N	S	
pkgownid (package owner identifier)	Y	N	S	
rdbrlsopt (relational database release option)	Y	N	S	
dftrdbcol (default relational database collection identifier)	Y	Y	S	DFTRDBCOL
title (brief description of package)	Y	Y	S	TEXT
qryblkctl (query block protocol control)	Y	N	Y	
pkgdftcst (default character subtype)	Y	Y	Y	See note 8.
pkgdftcc (package default CCSID)	Y	Y	Y	See note 8.
decprc (decimal precision)	Y	N	S	

Figure D-3 (Page 2 of 2). BGNBND Command

DDM Codepoint	Optional	R	S	CRTSQLxxx Pre-compile Options

Notes:

1. For the PKGRPLVRS parameter, the application requester never sends a value. The application server allows a null value, otherwise VALNSPRM is returned.
2. For the PKGISOLVL parameter, the application requester user specifies:
   - COMMIT(*NONE), mapped to zero. (*NONE applies to AS/400 system to AS/400 system only.)
   - COMMIT(*CHG), mapped to ISOLVLCHG.
   - COMMIT(*CS), mapped to ISOLVLCS.
   - COMMIT(*ALL), mapped to ISOLVLALL.

   The application server maps the request of ISOLVLRR to COMMIT(*CHG) with full table locking.
3. For the BNDEXPOPT parameter, only the DDM default value (EXPNON) is supported. Other values are rejected, and the application server returns a "Value not Supported" reply message.
4. For the PKGOWNID parameter, if a package owner id longer than 10 characters is received, the application server returns a "Value not Supported" reply message.
5. For the RDBRLSOPT parameter, only the DDM default value (RDBRLSCMM) is supported, otherwise the request is rejected.
6. Using the DFTRDBCOL parameter, the application requester user can specify the default collection ID. The default value of the default collection ID is the application requester end user ID.

   If the application server receives a a collection name longer than 10 characters, it returns a "Value not Supported" reply message.
7. For the TITLE parameter, a length greater than fifty characters is truncated by the application server.
8. For the PKGDFTCST and PKGDFTCC parameters, values are based on the job attributes determined by the system values QICG and QCCSID, as well as any job or user profile overrides.
9. For the DECPRC parameter, if a decimal precision of 15 is received, the application server returns a "Value not Supported" reply message.
10. For the VRSNAM parameter, if a value other than null is received, the AS sends a "Value not supported" reply message.

Figure D-4. Reply Objects for BGNBND command

DDM Codepoint	Optional	R	S
Typdefnam (data type definition name)	Y	Y	N
Typdefovr (TYPDEF override)	Y	Y	Y
Sqlcard (SQLCA reply data)	N	Y	Y

Figure D-5 (Page 1 of 2). BNDSQLSTT Command

DDM Codepoint	Optional	R	S	
BNDSQLSTT		Y	Y	Part of the package creation process.
rdbnam (name of relational database as in ACCRDB)	Y	Y	Y	

Figure D-5 (Page 2 of 2). BNDSQLSTT Command

DDM Codepoint	Optional	R	S
pkgnamcsn (package name, consistency token and section #)		Y	Y
rdbnam (application server database name). rdbcolid (relational database collection identifier). pkgid (package identifier). pkgcnstkn (package consistency token). pkgsn (package section number).			
sqlsttnbr (source application statement number)	Y	Y	Y
bndsttasm (bind statement assumptions)	Y	Y	Y
TYPDEFNAM (data type definition name)	Y	N	Y
TYPDEFOVR (TYPDEF override)	Y	Y	Y
SQLSTT (SQL statement to be bound in the AS package)		Y	Y
SQLSTTVRB (description of each variable)	Y	Y	Y
SQLPRECISION (precision of fixed decimal field, or zero). SQLSCALE (scale of fixed/zoned decimal, or zero). SQLLENGTH (length of field - not counting length field). SQLTYPE (SQL data type associated with field). SQLCCSID (0 or CCSID for the column). SQLNAME (name of program variable in statement). SQLDIAGNAME (fully qualified name of program variable).			

Figure D-6. Reply Objects for BNDSQLSTT command

DDM Codepoint	Optional	R	S
Typdefnam (data type definition name)	Y	Y	N
Typdefovr (TYPDEF override)	Y	Y	N
Sqlcard (SQLCA reply data)		Y	Y

Figure D-7. CLSQRY Command

DDM Codepoint	Optional	R	S	SQL Statement
CLSQRY		Y	Y	CLOSE
rdbnam (name of relational database as in ACCRDB)	Y	N	Y	
pkgnamcsn (package name, consistency token and section #)		Y	Y	
rdbnam (application server database name). rdbcolid (relational database collection identifier). pkgid (package identifier). pkgcnstkn (package consistency token). pkgsn (package section number).				

Figure D-8. Reply Objects for CLSQRY command

DDM Codepoint	Optional	R	S
Typdefnam (data type definition name)	Y	Y	N
Typdefovr (TYPDEF override)	Y	Y	N
Sqlcard (SQLCA reply data)		Y	Y

Figure D-9. CNTQRY Command

DDM Codepoint	Optional	R	S	SQL Statement
CNTQRY		Y	Y	FETCH
rdbnam (name of relational database as in ACCRDB)	Y	N	Y	
pkgnamcsn (package name, consistency token and section #) rdbnam (application server database name). rdbcolid (relational database collection identifier). pkgid (package identifier). pkgcnstkn (package consistency token). pkgsn (package section number).		Y	Y	
qryblksz (query block size)		Y	Y	

Figure D-10. Reply Objects for CNTQRY command

DDM Codepoint	Optional	R	S
Typdefnam (data type definition name)	Y	Y	N
Typdefovr (TYPDEF override)	Y	Y	N
Sqlcard (SQLCA reply data)		Y	Y
Qrydta (query answer set data)		Y	Y

Figure D-11. DRPPKG Command

DDM Codepoint	Optional	R	S	SQL Statement
DRPPKG		N	Y	DROP PACKAGE
rdbnam (name of relational database as in ACCRDB)	Y		Y	
pkgnam (package grouping name and identifier)			Y	
rdbnam (application server database name). rdbcolid (relational database collection identifier). pkgid (package identifier).				
vrsnam (version name)	Y		S	

DRPPKG Parameters Clarification:

- VRSNAM
  - Application server rejects this parameter if it is not the DDM default value (NULL).

Figure D-12. Reply Objects for DRPPKG command

DDM Codepoint	Optional	R	S
Typdefnam (data type definition name)	Y		N
Typdefovr (TYPDEF override)	Y		N
Sqlcard (SQLCA reply data)			Y

Figure D-13 (Page 1 of 2). DSCRDBTBL Command

DDM Codepoint	Optional	R	S	SQL Statement
DSCRDBTBL		Y	Y	DESCRIBE TABLE
rdbnam (name of relational database as in ACCRDB)	Y	N	Y	

Figure D-13 (Page 2 of 2). DSCRDBTBL Command

DDM Codepoint	Optional	R	S	SQL Statement
TYPDEFNAM (data type definition name)	Y	N	I	
TYPDEFOVR (TYPDEF override)	Y	Y	Y	
SQLOBJNAM (SQL object name)		Y	Y	

Figure D-14. Reply Objects for DSCRDBTBL command

DDM Codepoint	Optional	R	S
Typdefnam (data type definition name)	Y	Y	N
Typdefovr (TYPDEF override)	Y	Y	N
Sqlcard (SQLCA reply data)		Y	Y
Sqldard (SQLDA reply data)		Y	Y

Figure D-15. DSCSQLSTT Command

DDM Codepoint	Optional	R	S	SQL Statement
DSCSQLSTT		Y	Y	DESCRIBE
rdbnam (name of relational database as in ACCRDB)	Y	N	Y	
pkgnamcsn (package name, consistency token and section #)		Y	Y	
rdbnam (application server database name). rdbcolid (relational database collection identifier). pkgid (package identifier). pkgcnstkn (package consistency token). pkgsn (package section number).				

Figure D-16. Reply Objects for DSCSQLSTT command

DDM Codepoint	Optional	R	S
Typdefnam (data type definition name)	Y	Y	N
Typdefovr (TYPDEF override)	Y	Y	N
Sqlcard (SQLCA reply data)		Y	Y
Sqldard (SQLDA reply data)		Y	Y

Figure D-17 (Page 1 of 2). ENDBND Command

DDM Codepoint	Optional	R	S	
ENDBND		Y	Y	Part of the package creation process.
rdbnam (name of relational database as in ACCRDB)	Y	Y	Y	
pkgnamct (package name and consistency token)		Y	Y	
rdbnam (application server database name). rdbcolid (relational database collection identifier). pkgid (package identifier). pkgcnstkn (package consistency token).				

Figure D-17 (Page 2 of 2). ENDBND Command

DDM Codepoint	Optional	R	S
maxsctnbr (maximum section number)	Y	Y	Y

Figure D-18. Reply Objects for ENDBND command

DDM Codepoint	Optional	R	S
Typdefnam (data type definition name)	Y	Y	N
Typdefovr (TYPDEF override)	Y	Y	N
Sqlcard (SQLCA reply data)		Y	Y

Figure D-19. EXCSAT Command

DDM Codepoint	Optional	R	S	Requester Values
EXCSAT		Y	Y	
extnam (external name)	Y	Y	Y	Qualified name at the application requester.
mgrlvlls (manager level list)	Y	Y	Y	Always fixed.
spvnam (supervisor name)	Y	N	I	
srvclsnm (server class name)	Y	Y	Y	QAS
srvnam (server – can be ignored)	Y	Y	Y	System name.
srvrlslv (server release level – can be ignored)	Y	Y	Y	VvvRrrMmm

Note: For the SRVRLSLV parameter, vv, rr, and mm designate the version, release, and modification level.

Figure D-20 (Page 1 of 2). EXCSATRD Reply for EXCSAT command

DDM Codepoint	Optional	R	S	Server Values
EXCSATRD		Y	Y	
extnam (external name)	Y	Y	Y	Fully qualified job name at the application server.
mgrlvlls (manager level list)	Y	Y	Y	From list from the application requester.
srvclsnm (server class name)	Y	Y	Y	QAS
srvnam (server name)	Y	Y	Y	System name.
srvrlslv (server release level)	Y	Y	Y	VvvRrrMmm

Figure D-20 (Page 2 of 2). EXCSATRD Reply for EXCSAT command

DDM Codepoint	Optional	R	S	Server Values

Note: For the SRVRLSLV parameter, vv, rr, and mm designate the version, release, and modification level.

Figure D-21. EXCSQLIMM Command

DDM Codepoint	Optional	R	S	SQL Statement
EXCSQLIMM		Y	Y	EXECUTE IMMEDIATE
rdbnam (name of relational database as in ACCRDB)	Y	N	Y	
pkgnamcsn (package name, consistency token and section #) rdbnam (application server database name). rdbcolid (relational database collection identifier). pkgid (package identifier). pkgcnstkn (package consistency token). pkgsn (package section number).		Y	Y	
TYPDEFNAM (data type definition name)	Y	N	Y	
TYPDEFOVR (TYPDEF override)	Y	Y	Y	
SQLSTT (SQL statement - *cannot contain host variables*).		Y	Y	

Figure D-22. Reply Objects for EXCSQLIMM command

DDM Codepoint	Optional	R	S
Typdefnam (data type definition name)	Y	Y	N
Typdefovr (TYPDEF override)	Y	Y	N
Sqlcard (SQLCA reply data)	Y	Y	Y

Figure D-23 (Page 1 of 2). EXCSQLSTT Command

DDM Codepoint	Optional	R	S	SQL Statement
EXCSQLSTT		Y	Y	EXECUTE
rdbnam (name of relational database as in ACCRDB)	Y	N	Y	
pkgnamcsn (package name, consistency token and section #) rdbnam (application server database name). rdbcolid (relational database collection identifier). pkgid (package identifier). pkgcnstkn (package consistency token). pkgsn (package section number).		Y	Y	
outexp (output expected- *specifies non-cursor SELECT*)	Y	Y	Y	For non-cursor SELECT only.
TYPDEFNAM (data type definition name)	Y	N	Y	
TYPDEFOVR (TYPDEF override)	Y	Y	Y	

Figure D-23 (Page 2 of 2). EXCSQLSTT Command

DDM Codepoint	Optional	R	S	SQL Statement
SQLDTA (SQL program variable data)	Y	Y	Y	Input SQLDA and host variables.

Figure D-24. Reply Objects for EXCSQLSTT command

DDM Codepoint	Optional	R	S
Typdefnam (data type definition name)	Y	Y	N
Typdefovr (TYPDEF override)	Y	Y	N
Sqlcard (SQLCA reply data)		Y	Y
Sqldtard (SQL data reply data)		Y	Y

Figure D-25. INTRDBRQS Command

DDM Codepoint	Optional	R	S	CL Command
INTRDBRQS		Y	Y	ENDRDBRQS
rdbnam (name of relational database as in ACCRDB)	Y	N	Y	
rdbinttkn (relational database interrupt token)		Y	Y	

Figure D-26. CMDCMPRM Reply for INTRDBRQS command

DDM Codepoint	Optional	R	S
CMDCMPRM (cmd completion reply msg)		Y	Y
svrcod (severity code)		Y	Y
srvdgn (server diagnostic information)	Y	I.	N

Figure D-27 (Page 1 of 2). OPNQRY Command

DDM Codepoint	Optional	R	S	SQL Statement
OPNQRY		Y	Y	OPEN
rdbnam (name of relational database as in ACCRDB)	Y	N	Y	
pkgnamcsn (package name, consistency token and section #) rdbnam (application server database name). rdbcolid (relational database collection identifier). pkgid (package identifier). pkgcnstkn (package consistency token). pkgsn (package section number).		Y	Y	
qryblksz (query block size)		Y	Y	
qryblkctl (query block protocol control)	Y	Y	Y	
TYPDEFNAM (data type definition name)	Y	N	Y	
TYPDEFOVR (TYPDEF override)	Y	Y	Y	

Figure D-27 (Page 2 of 2). OPNQRY Command

DDM Codepoint	Optional	R	S	SQL Statement
SQLDTA (input variable data)	Y	Y	Y	Input SQLDA and host variables.

Figure D-28. Reply Message and Reply Objects for OPNQRY command

DDM Codepoint	Optional	R	S
OPNQRYRM (open query reply message)		Y	Y
svrcod (severity code)		Y	Y
qryprctyp (protocol type)		Y	Y
sqlcsrhld (cursor hold flag)	Y	Y	N
srvdgn (server diagnostic information)	Y	I.	N
Typdefnam (data type definition name)	Y	Y	N
Typdefovr (TYPDEF override)	Y	Y	N
Sqlcard (SQLCA reply data)	Y	Y	Y
Qrydsc (query answer set description)	Y	Y	Y
Qrydta (query answer set data)	Y	Y	Y

Figure D-29. PRPSQLSTT Command

DDM Codepoint	Optional	R	S	SQL Statement
PRPSQLSTT		Y	Y	Dynamic PREPARE
rdbnam (name of remote database as in ACCRDB)	Y	N	Y	
pkgnamcsn (package name, consistency token and section #)		Y	Y	
rdbnam (application server database name). rdbcolid (relational database collection identifier). pkgid (package identifier). pkgcnstkn (package consistency token). pkgsn (package section number).				
rtnsqlda (specifies if SQLDA should be returned)	Y	Y	Y	USING clause
TYPDEFNAM (data type definition name)	Y	N	Y	
TYPDEFOVR (TYPDEF override)	Y	Y	Y	
SQLSTT (SQL Statement)		Y	Y	

Figure D-30. Reply Objects for PRPSQLSTT command

DDM Codepoint	Optional	R	S
Typdefnam (data type definition name)	Y	Y	N
Typdefovr (TYPDEF override)	Y	Y	N
Sqlcard (SQLCA reply data)	Y	Y	Y
Sqldard (SQLDA reply data)	Y	Y	Y

Figure D-31. RDBCMM Command

DDM Codepoint	Optional	R	S	SQL Statement
RDBCMM		Y	Y	COMMIT
rdbnam (name of relational database as in ACCRDB)	Y	N	Y	

Figure D-32. Reply Objects for RDBCMM command

DDM Codepoint	Optional	R	S
Typdefnam (data type definition name)	Y	Y	N
Typdefovr (TYPDEF override)	Y	Y	N
Sqlcard (SQLCA reply data)	Y	Y	Y

Figure D-33. RDBRLLBCK Command

DDM Codepoint	Optional	R	S	SQL Statement
RDBRLLBCK		Y	Y	ROLLBACK
rdbnam (name of relational database as in ACCRDB)	Y	N	Y	

Figure D-34. Reply Objects for RDBRLLBCK command

DDM Codepoint	Optional	R	S
Typdefnam (data type definition name)	Y	Y	N
Typdefovr (TYPDEF override)	Y	Y	N
Sqlcard (SQLCA reply data)	Y	Y	Y

Figure D-35. REBIND Command

DDM Codepoint	Optional	R	S
REBIND	N	N	N

Note: Use of REPLACE(*YES) with CRTSQLPKG provides a function similar to that architected for REBIND. However, since the function is not an exact match, the REBIND command is not supported by the OS/400 licensed program.

Bibliography

This bibliography lists four classifications of manuals available from IBM that are related to this guide. These classifications are:

- AS/400 system library
- Distributed Relational Database Library
- Other IBM Distributed Relational Database Platform Libraries
- Architecture manuals

AS/400 System Information

The following AS/400 manuals contain information you may need. The manuals are listed alphabetically by their short title. The short title is what is used in the text of this guide. The long title and order number are also given for ordering and referencing purposes. For a complete list and description of all the information available for the AS/400 system, see the *Publications Guide*, GC41-9678.

- *Alerts and DSNX Guide*, provides information for configuring an AS/400 system to use the remote management support (distributed host command facility), the change management support (distributed systems node executive) and the problem management support (alerts).

  Long title: *Communications and Systems Management Guide (Alerts and Distributed Systems Node Executive)*, SC41-9661

- *APPC Programmer's Guide*, provides information for developing communication application programs that use APPC and for defining the communications environment for APPC communications.

  Long title: *Communications: Advanced Program-to-Program Communications Programmer's Guide*, SC41-8189

- *APPN Guide*, provides the system programmer with information about the advanced peer-to-peer networking (APPN) support provided by the AS/400 system. It describes APPN concepts, functions and features as carried out on the AS/400 system and also presents considerations when using APPN.

  Long title: *Communications: Advanced Peer-to-Peer Networking Guide*, SC41-8188

- *Advanced Backup and Recovery Guide*, provides the system programmer with information about the different media available to save and restore system data, as well as a description of how to record changes made to database files and how that information can be used for system recovery and activity report information.

  Long title: *Advanced Backup and Recovery Guide*, SC41-8079

- *CL Programmer's Guide*, provides a wide-ranging discussion of programming topics, including a general discussion of objects and libraries, control language (CL) programming, controlling flow and communicating between programs, working with objects in CL programs, and creating CL programs. Other topics include predefined and immediate messages and message handling, defining and creating user-defined commands and menus, and application testing, including debug mode, breakpoints, traces, and display functions.

  Long title: *Programming: Control Language Programmer's Guide*, SC41-8077

- *CL Reference*, provides a description of the control language (CL) and its commands. Each command is defined including its syntax diagram, parameters, default values, and keywords.

  Long title: *Programming: Control Language Reference*, SC41-0030

- *Communications Management Guide*, contains information on working with communications status, communications-related work management topics, communications errors, performance, line speed and subsystem storage.

  Long title: *Communications: Management Guide*, SC41-0024

- *DDM Guide*, provides the application programmer or system programmer with information about remote file processing. It describes how to define a remote file to OS/400 distributed data management (DDM), how to create a DDM file, which file utilities are supported through DDM, and the requirements of OS/400 DDM as related to other systems.

  Long title: *Distributed Data Management Guide*, SC41-9600

- *Device Configuration Guide*, provides the system operator or system administrator with information on how to do an initial local hardware configuration and how to change that configuration. It also contains conceptual information for device configuration, and planning information for device configuration on the 9406, 9404, and 9402 System Units.

  Long title: *Device Configuration Guide*, SC41-8106

- *DFU User's Guide and Reference*, provides the application programmer, programmer or help desk aide with information about the Application Development Tools data file utility (DFU) to create programs to enter data into files, update files, inquire into files and run DFU programs. This guide also provides

the work station operator with activities and material to learn about DFU.

Long title: *Application Development Tools: Data File Utility User's Guide and Reference*, SC09-1381

- *Distribution Services Network Guide*, provides the system programmer or network administrator with information about configuring a communications network for Systems Application Architecture distribution services (SNADS) and the Virtual Machine/Multiple Virtual Storage (VM/MVS) bridge. In addition, object distribution functions, document library services and system distribution directory services are also discussed.

  Long title: *Communications: Distribution Services Network Guide*, SC41-9588

- *ICF Programmer's Guide*, provides the application programmer with the information needed to write application programs that use AS/400 communications and ICF files. It also contains information on data description specifications (DDS) keywords, system-supplied formats, return codes, file transfer support, and programming examples.

  Long title: *Communications: Intersystem Communications Function Programmer's Guide*, SC41-9590

- *ISDN Guide*, contains information on using an AS/400 system in an Integrated Services Digital Network (ISDN) network environment.

  Long title: *Communications: Integrated Services Digital Network Guide*, SC41-0003

- *Local Area Network Guide*, contains information on using an AS/400 system in a token-ring network, an Ethernet network, or bridged network environment.

  Long title: *Communications: Local Area Network Guide*, SC41-0004

- *National Language Support Planning Guide*, provides information required to understand and use the national language support function on the AS/400 system. This manual prepares the AS/400 user for planning and using the national language support (NLS) and the multilingual support of the AS/400 system. It also provides an explanation of the database management of multilingual data and application considerations for a multilingual system.

  Long title: *National Language Support Planning Guide*, GC41-9877

- *OSI Communications Subsystem Programming Concepts and Guide*,

  explains distributed processing from an OSI perspective and helps the user understand how ISO standards and the OSI Reference Model are used to satisfy the design goals and provide many-vendor distributed processing environments with connectivity and inter-operability. It also tells how to use the OSI communications subsystem to organize and distribute applications, represent data, and manage the communications dialog used between applications.

  Long title: *OSI Communications Subsystem Programming Concepts and Guide*, SL23-0191

- *OS/400* Communications Configuration Reference*, provides information for the application programmer or system programmer about configuration commands and defining lines, controllers, and devices.

  Long title: *Communications: Operating System/400* Communications Configuration Reference*, SC41-0001

- *Query Management/400 Programmer's Guide and Reference*, provides the application programmer with information on how to determine the database files to be queried for a report, define a structured query language (SQL) query definition, and use and write procedures that use query management commands. This manual also includes information on how to use the query global variable support and understanding the relationship between the OS/400 query management and the AS/400 Query licensed program.

  Long title: *Query Management/400 Programmer's Guide and Reference*, SC41-0090

- *Query Management/400 Programmer's Guide and Reference*, provides the application programmer with information on how to write applications that use OS/400 query management.

  Long title: *Query Management/400 Programmer's Guide and Reference*, SC41-0090

- *Remote Work Station Guide*, provides information on how to set up and use remote work station support, such as display station pass-through, distributed host command facility, and 3270 remote attachment.

  Long title: *Communications: Remote Work Station Guide*, SC41-0002

- *Security Reference*, provides the system programmer (or someone who is assigned the responsibilities of a security officer) with information about system security concepts, planning for security, and setting up security on the system.

  Long title: *Security Reference*, SC41-8083

- *SQL/400* Programmer's Guide*, provides the application programmer, programmer, or database administrator with an overview of how to design, write, test and run SQL/400 statements. It also describes interactive Structured Query Language (SQL).

  Long title: *Systems Application Architecture* Structured Query Language/400 Programmer's Guide*, SC41-9609

- *SQL/400* Reference*, provides the application programmer, programmer, or database administrator with detailed information about Structured Query Language/400 statements and their parameters.

  Long title: *Systems Application Architecture* Structured Query Language/400 Reference*, SC41-9608

- *System Concepts*, provides the programmer and system user with information about the concepts related to the overall design and use of the AS/400 system and its operating system. This manual includes general information about topics such as user interface, object management, work management, system management, data management, database, communications, environments, OfficeVision/400, PC Support/400, and architecture.

  Long title: *System Concepts*, GC41-9802

- *Operator's Guide*, provides information about how to use the system unit control panel and console, send and receive messages, respond to error messages, start and stop the system, use control devices, work with program temporary fixes (PTFs), and process and manage jobs on the system.

  Long title: *System Operator's Guide*, SC41-8082

- *TCP/IP Guide*, provides the application programmer or end user with information about how the AS/400 system carries out TCP/IP. This guide describes how to use and configure TCP/IP applications of FTP, SMTP and TELNET. It also provides information about the relationship of TCP/IP to other AS/400 communications protocols.

  Long title: *Transmission Control Protocol/Internet Protocol Guide*, SC41-9875

- *Work Management Guide*, provides the system programmer with information about how to create and change a work management environment. It also includes a description of tuning the system, collecting performance data, working with system values to control or change the overall operation of the system, and a description of how to gather data to determine who is using the system and what resources are being used.

  Long title: *Programming: Work Management Guide*, SC41-8078

- *X.25 Network Guide*, contains information on using AS/400 systems in an X.25 network.

  Long title: *Communications: X.25 Network Guide*, SC41-0005

Distributed Relational Database Library

The following manuals provide background and general support information for IBM Distributed Relational Database Architecture implementations.

- *SAA Introduction to Distributed Data*, GC26-4831, provides concise, high-level education on distributed relational database and distributed file. This manual describes how IBM's Systems Application Architecture supports the development of distributed data systems, and discusses some current IBM products and announced support for distributed data. The information in this manual is intended to help executives, managers, and technical personnel understand the concepts of distributed data.

- *Planning for Distributed Data*, SC26-4650, helps you plan for distributed relational data. It describes the steps to take, the decisions to make, and the options from which to choose in making those decisions. The manual also covers the distributed relational database products and capabilities that are now available or that have been announced, and it discusses IBM's stated direction for supporting distributed relational data in the future. The information in this manual is intended for planners.

- *Distributed Relational Database Connectivity Concepts and Scenarios*, SC26-4783, describes how to interconnect IBM products that support Distributed Relational Database Architecture. It explains concepts and terminology associated with distributed relational database and network systems. This manual tells you how to connect unlike systems in a distributed environment. The information in the Connectivity Guide is not included in any product documentation. The information in this manual is intended for system administrators, database administrators, communication administrators, and system programmers.

- *Distributed Relational Database Application Programming Guide*, SC26-4773, describes how to design, build, and modify application programs that access IBM's SAA relational database management systems. This manual focuses on what a programmer should do differently when writing distributed relational database applications for unlike environments. Topics include program design, preparation, and execution, as well as performance considerations. Programming examples written in IBM SAA C are included. The information in this manual is designed for application programmers who work with at least one of IBM's SAA high-level languages and with Structured Query Language (SQL).

- *Distributed Relational Database Problem Determination Guide*, SC26-4782, helps you define the source of problems in a distributed relational data-

base environment. This manual contains introductory material on each product, for people not familiar with those products, and gives detailed information on how to diagnose and report problems with each product. The guide describes procedures and tools unique to each host system and those common among the different systems. The information in this manual is intended for the people who report distributed relational database problems to the IBM Support Center.

Other IBM Distributed Relational Database Platform Libraries

The following lists of books are available for IBM distributed relational databases that are supported with the AS/400 distributed relational database at this time.

IBM DATABASE 2 Database Management System

- *IBM DATABASE General Information*, GC26-4373
- *IBM DATABASE Administration Guide*, SC26-4374
- *IBM DATABASE Application Programming and SQL Guide*, SC26-4377
- *IBM DATABASE Command and Utility Reference*, SC26-4378
- *IBM DATABASE SQL Reference*, SC26-4380
- *IBM DATABASE Diagnosis Guide and Reference*, LY27-9536
- *IBM DATABASE Messages and Codes*, SC26-4379

SQL/DS Database Management System

- *SQL/DS Application Programming for IBM VM System Products*, SH09-8086
- *SQL/DS Concepts and Facilities for IBM VM System Products*, GH09-8044
- *SQL/DS Database Administration for IBM VM System Products*, GH09-8083
- *SQL/DS Database Services Utility for IBM VM System Products*, SH09-8088
- *SQL/DS Diagnosis Guide for IBM VM System Products*, LH09-8054
- *SQL/DS General Information for IBM VM System Products*, GH09-8074
- *SQL/DS SQL Reference for IBM VM System Products*, SH09-8087
- *SQL/DS Reference Summary for IBM VM System Products*, SX09-1173
- *SQL/DS ISQL Reference Summary for IBM VM System Products*, SX09-1057
- *SQL/DS Messages & Codes for IBM VM System Products*, SH09-8078
- *SQL/DS Operation for IBM VM System Products*, SH09-8080
- *SQL/DS System Administration for IBM VM System Products*, GH09-8084
- *Managing SQL/DS System*, SH09-8077
-

Architecture Manuals

- *Character Data Representative Architecture Level 1, Registry*, SC09-1391 SC26-4651.
- *Distributed Relational Database Architecture Reference*, SC26-4651.
- *Distributed Data Management Level 3.0 Architecture Reference*, SC21-9526.
- *System Application Architecture, Common Programming Interface, Database Reference*, SC26-4348
- *System Application Architecture, Database Level 2 Reference*, SC26-4798

Index

Special Characters

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Customer Satisfaction Feedback

Application System/400
Distributed Relational Database Guide
Version 2

Publication No. SC41-0025-01

Overall, how would you rate this manual?

	Very Satisfied	Satisfied	Dissatis-fied	Very Dissatis-fied
Overall satisfaction				

How satisfied are you that the information in this manual is:

Accurate				
Complete				
Easy to find				
Easy to understand				
Well organized				
Applicable to your tasks				
THANK YOU!				

Please tell us how we can improve this manual:

_____

_____

_____

_____

_____

May we contact you to discuss your responses? __ Yes __ No

Phone: (____) _____ Fax: (____) _____

To return this form:

- Mail it
- Fax it
  United States and Canada: **800+937-3430**
  Other countries: **(+1)+507+253-5192**
- Hand it to your IBM representative.

Note that IBM may use or distribute the responses to this form without obligation.

_____ _____
Name Address

_____ _____
Company or Organization

_____ _____
Phone No.

Customer Satisfaction Feedback
SC41-0025-01

IBM®

Cut o
Along

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

ATTN DEPT 245
IBM CORPORATION
3605 HWY 52 N
ROCHESTER MN 55901-7899

Fold and Tape **Please do not staple** Fold and Tape

SC41-0025-01

Cut or
Along

Customer Satisfaction Feedback

Application System/400
Distributed Relational Database Guide
Version 2

Publication No. SC41-0025-01

Overall, how would you rate this manual?

	Very Satisfied	Satisfied	Dissatis-fied	Very Dissatis-fied
Overall satisfaction				

How satisfied are you that the information in this manual is:

Accurate				
Complete				
Easy to find				
Easy to understand				
Well organized				
Applicable to your tasks				
THANK YOU!				

Please tell us how we can improve this manual:

May we contact you to discuss your responses? __ Yes __ No

Phone: (____) _____ Fax: (____) _____

To return this form:

- Mail it
- Fax it
  United States and Canada: **800+937-3430**
  Other countries: **(+1)+507+253-5192**
- Hand it to your IBM representative.

Note that IBM may use or distribute the responses to this form without obligation.

Name _____ Address _____

Company or Organization _____ _____

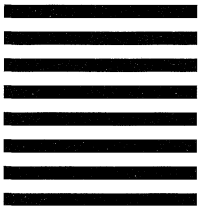
Phone No. _____ _____

IBM®

Cut c
Alon

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

ATTN DEPT 245
IBM CORPORATION
3605 HWY 52 N
ROCHESTER MN 55901-7899

Fold and Tape **Please do not staple** Fold and Tape

Cut o
Along

Program Number: 5738-SS1

Printed in Denmark by Bonde's

SC41-0025-01