SYBASE®

Sybase® Adaptive Server™ Enterprise
Security Administration Guide

Adaptive Server™

Document ID: 35811-01-1150

September 1997

Table of Contents

About This Book

Audience . . . xvii
How to Use This Book . . . xvii
Adaptive Server Enterprise Documents . . . xviii
Other Sources of Information . . . xx
Conventions . . . xx
- Formatting SQL Statements . . . xx
- SQL Syntax Conventions . . . xxi
  - Case . . . xxii
  - Expressions . . . xxiii

If You Need Help . . . xxiii

1. Overview of Security Features

Security Features Available in Adaptive Server . . . 1-1
Discretionary Access Controls . . . 1-2
Identification and Authentication Controls . . . 1-3
Division of Roles . . . 1-3
- System Roles . . . 1-3
- User-Defined Roles . . . 1-5
- Role Hierarchies and Mutual Exclusivity . . . 1-5
  - Role Hierarchy . . . 1-5
  - Mutual Exclusivity . . . 1-5

Auditing . . . 1-6
General Process of Security Administration . . . 1-6

2. Getting Started After Installation

Guidelines For Setting Up Security . . . 2-1
- Using the “sa” Login . . . 2-1
- Changing the “sa” Login Password . . . 2-1
- When To Enable Auditing . . . 2-2
- Assigning Login Names . . . 2-2

Steps For Setting Up Security . . . 2-2
An Example of Setting Up Security . . . 2-4

3. Managing Adaptive Server Logins and Database Users

Adding New Users: An Overview 3-1
Choosing and Creating a Password 3-2
Adding Logins to Adaptive Server 3-3
Creating Groups 3-5
Adding Users to Databases 3-6
Adding a "guest" User to a Database 3-7
"guest" User Permissions 3-8
"guest" User in User Databases 3-8
"guest" User in *pubs2* and *pubs3* 3-9
Creating Visitor Accounts 3-9
Adding Remote Users 3-9
Assigning Roles and Granting Permissions to Users 3-10
Dropping Users and Groups 3-10
Dropping Users 3-10
Dropping Groups 3-11
Locking or Dropping Adaptive Server Login Accounts 3-11
Locking and Unlocking Login Accounts 3-12
Dropping Login Accounts 3-13
Changing User Information 3-13
Changing Passwords 3-14
Requiring New Passwords 3-14
Null Passwords 3-15
Changing User Defaults 3-15
Changing a User's Group Membership 3-16
Using Aliases in Databases 3-17
Adding Aliases 3-18
Dropping Aliases 3-19
Getting Information About Aliases 3-19
Getting Information About Users 3-20
Getting Reports on Users and Processes 3-20
Getting Information About Login Accounts 3-21
Getting Information About Database Users 3-21
Finding User Names and IDs 3-22
Getting Information About Usage: Chargeback Accounting 3-23
Reporting Current Usage Statistics 3-23
Displaying Current Accounting Totals 3-24
Initiating a New Accounting Interval 3-24
Specifying the Interval for Adding Accounting Statistics 3-24

4. Administering Roles

Overview . . . 4-1
System Roles . . . 4-2
User-Defined Roles . . . 4-2
Role Hierarchies and Mutual Exclusivity . . . 4-2
Tasks and Permissions for System Roles . . . 4-3
System and Security Administration Roles . . . 4-4
System Administrator . . . 4-4
System Security Officer . . . 4-5
Operator . . . 4-6
Data Ownership Roles . . . 4-6
Database Owner . . . 4-6
Database Object Owner . . . 4-7
Managing System Roles in Adaptive Server . . . 4-8
Preparing for User-Defined Roles . . . 4-9
Planning User-Defined Roles . . . 4-9
Configuring User-Defined Roles . . . 4-10
Creating and Dropping User-Defined Roles . . . 4-10
Creating a Role . . . 4-10
Dropping a User-defined Role . . . 4-10
Adding and Removing Passwords from a Role . . . 4-11
Defining and Changing Mutual Exclusivity of Roles . . . 4-11
Defining and Changing a Role Hierarchy . . . 4-12
Setting Up Default Activation at Login . . . 4-15
Granting and Revoking Roles . . . 4-16
Granting Roles . . . 4-17
Revoking Roles . . . 4-17
Granting Access to Database Objects . . . 4-18
Activating and Deactivating Roles . . . 4-18
Displaying Information About Roles . . . 4-19
Finding a Role ID . . . 4-19
Finding a Role Name . . . 4-20
Viewing Information About System Roles . . . 4-20
Displaying a Role Hierarchy . . . 4-20
Viewing User Roles in a Hierarchy . . . 4-21
Determining Mutual Exclusivity . . . 4-21
Determining Role Activation . . . 4-21
Checking for Roles in Stored Procedures . . . 4-21
Displaying Login Account Information . . . 4-22
Checking Permissions . . . 4-23
Role Information in the System Tables . . . 4-23

5. Managing User Permissions

Overview 5-1
Types of Users and Their Privileges 5-2
System Administrator Privileges 5-2
Permissions for Creating Databases 5-3
System Security Officer Privileges 5-4
Operator Privileges 5-4
Database Owner Privileges 5-5
Permissions on System Tables 5-6
Permissions on System Procedures 5-6
Acquiring the Permissions of Another User 5-7
Changing Database Ownership 5-8
Database Object Owner Privileges 5-9
Privileges of Other Database Users 5-9
Granting and Revoking Permissions on Database Objects 5-10
Granting and Revoking Object Access Permissions 5-10
Special Requirements for Compliance to the SQL92 Standard 5-14
Examples of Granting Object Access Permissions 5-15
Examples of Revoking Object Access Permissions 5-15
Granting and Revoking Object Creation Permissions 5-16
Examples of Granting Object Creation Permissions 5-17
Example of Revoking Object Creation Permissions 5-17
Combining *grant* and *revoke* Statements 5-18
Avoiding Conflicts In Permissions 5-18
Granting and Revoking Proxy Authorization and Roles 5-19
Reporting on Permissions 5-20
Reporting Permissions on Database Objects or Users 5-20
Reporting Permissions on Specific Tables 5-22
Reporting Permissions on Specific Columns 5-22
Using Views and Stored Procedures As Security Mechanisms 5-23
Using Views As Security Mechanisms 5-23
Using Stored Procedures As Security Mechanisms 5-25
Roles and Stored Procedures 5-26
Understanding Ownership Chains 5-26
Example of Views and Ownership Chains 5-27
Example of Procedures and Ownership Chains 5-29
Permissions on Triggers 5-31

6. Using Proxy Authorization

Understanding Proxy Authorization 6-1

How Users Use Proxy Authorization 6-2
How Applications Use Proxy Authorization 6-2
Administering and Using Proxy Authorization 6-3
Granting Permission to Use Proxy Authorization 6-3
Examples of Granting *set proxy* Permission 6-4
Setting Proxy Authorization 6-4
Examples of Setting Proxy Authorization 6-5
Getting Information About Proxy Authorization 6-6
Displaying Permissions Information 6-7
Querying the *systprotects* Table 6-7
Using *sp_helprotect* 6-7
Displaying Information about Users and Processes 6-7

7. Managing Remote Servers

Overview 7-1
Managing Remote Servers 7-3
Adding a Remote Server 7-3
Examples of Adding Remote Servers 7-4
Managing Remote Server Names 7-5
Setting Server Connection Options 7-5
Using the *timeouts* Option 7-5
Using the *net password encryption* Option 7-6
Getting Information About Servers 7-7
Dropping Remote Servers 7-7
Adding Remote Logins 7-7
Mapping Users' Server IDs 7-8
Mapping Remote Logins to Particular Local Names. 7-8
Mapping All Remote Logins to One Local Name 7-9
Keeping Remote Login Names for Local Servers 7-9
Example of Remote User Login Mapping. 7-10
Password Checking for Remote Users 7-11
Effects of Using the Untrusted Mode. 7-12
Trusted Mode and Mapping Remote Logins 7-12
Getting Information About Remote Logins 7-13
Configuration Parameters for Remote Logins 7-13
Allowing Remote Access 7-13
Controlling the Number of Active User Connections 7-14
Controlling the Number of Remote Sites 7-14
Controlling the Number of Active Remote Connections 7-14
Controlling Number of Pre-read Packets 7-14

8. Auditing

Overview . . . 8-1
The Audit System . . . 8-2
The *sybsecurity* Database . . . 8-2
The Audit Queue . . . 8-4
The Auditing Configuration Parameters . . . 8-5
The Auditing System Procedures . . . 8-5
Establishing and Managing Auditing . . . 8-6
Installing the Audit System . . . 8-7
Tables and Devices for the Audit Trail . . . 8-7
Device for the *syslogs* Transaction Log Table . . . 8-8
Installing Auditing with *installsecurity* . . . 8-8
Moving the Auditing Database to Multiple Devices . . . 8-9
Moving *sybsecurity* Without Saving Global Audit Settings . . . 8-9
Moving *sybsecurity* and Saving Global Audit Settings . . . 8-9
Preparing to Manage the Audit Trail . . . 8-10
Setting Up a Threshold Procedure . . . 8-11
Creating a Threshold Procedure to Handle the Audit Trail . . . 8-11
Attaching the Threshold Procedure to Each Audit Segment . . . 8-13
Setting Auditing Configuration Parameters . . . 8-16
Setting the Size of the Audit Queue . . . 8-16
Setting the Threshold for Auditing Devices . . . 8-17
Preparing to Manage the Transaction Log . . . 8-18
Managing the Transaction Log With Truncation . . . 8-18
Managing the Transaction Log With No Truncation . . . 8-19
Setting Auditing Options . . . 8-20
Examples of Setting Auditing Options . . . 8-20
Determining Current Auditing Settings . . . 8-21
Enabling and Disabling Auditing . . . 8-21
Adding User-Specified Records to the Audit Trail . . . 8-22
Examples of Adding User-Defined Audit Records . . . 8-23
Querying the Audit Trail . . . 8-23
Correlating Adaptive Server and Operating System Audit Records . . . 8-24
Understanding the Audit Tables . . . 8-24
Reading the *extrainfo* Column . . . 8-25

A. Single-Table Auditing

Requirements for Managing a Single-Device Audit Trail . . . A-1
Establishing and Managing Single-Table Auditing . . . A-3
Threshold Procedure for Single-Table Auditing . . . A-4

Archiving the Audit Table A-4
Sample Threshold Procedure A-4
What Happens When the Current Audit Table Is Full? A-5
Recovering When the Current Audit Table Is Full A-6

Index

List of Figures

Figure 4-1: Role hierarchy 4-3
Figure 4-2: Creating a role hierarchy 4-12
Figure 4-3: Explicitly and implicitly granted privileges 4-13
Figure 4-4: Granting a role to a role contained by grantor 4-14
Figure 4-5: Mutual exclusivity at membership 4-14
Figure 4-6: Effect of revoking roles on role hierarchy 4-15
Figure 5-1: Ownership chains and permission checking for views, case 1 5-28
Figure 5-2: Ownership chains and permission checking for views, case 2 5-29
Figure 5-3: Ownership chains and permission checking for stored procedures 5-30
Figure 6-1: Applications and proxy authorization 6-2
Figure 7-1: Setting up servers to allow remote procedure calls 7-2
Figure 8-1: Auditing with multiple audit tables 8-4
Figure A-1: Auditing with a single audit table A-2

List of Tables

Table 1: Font and syntax conventions for this manual xxi
Table 2: Types of expressions used in syntax statements xxiii
Table 1-1: Major security features 1-2
Table 1-2: General process for security administration 1-7
Table 2-1: Steps for getting started 2-2
Table 2-2: Users to whom you will assign roles 2-4
Table 2-3: Examples of commands used to set up security 2-5
Table 3-1: Adding users to Adaptive Server and databases 3-2
Table 3-2: Dropping users and groups 3-10
Table 3-3: Locking or dropping login accounts 3-12
Table 3-4: System procedures for changing user information 3-13
Table 3-5: System procedures for managing aliases 3-17
Table 3-6: Reporting information about Adaptive Server users and groups 3-20
Table 3-7: System functions suser_id and suser_name 3-22
Table 3-8: System functions user_id and user_name 3-23
Table 4-1: System roles and related tasks 4-2
Table 4-2: Tasks, required roles, and commands to use 4-16
Table 4-3: Finding information about roles 4-19
Table 5-1: Permissions and the objects to which they apply 5-11
Table 5-2: Object access permissions 5-12
Table 5-3: ANSI permissions for update and delete 5-14
Table 5-4: System procedures for reporting on permissions 5-20
Table 6-1: Methods for getting information about proxy authorization 6-6
Table 7-1: Tasks related to managing remote servers 7-3
Table 7-2: Configuration parameters that affect RPCs 7-13
Table 8-1: General procedure for auditing 8-6
Table 8-2: Columns in each audit table 8-24
Table 8-3: Information in the extrainfo column 8-25
Table 8-4: Values in event and extrainfo columns 8-26
Table A-1: Auditing process for single-table auditing A-3

About This Book

This manual, the *Security Administration Guide*, describes how to administer security for Sybase® Adaptive Server™ Enterprise. It explains how to use the security features provided by Adaptive Server to control user access to data.

This manual was the primary component of the Trusted Facility Manual (TFM), designed to meet the Trusted Computer System Evaluation Criteria (TCSEC) requirements. SQL Server release 11.0.6 was evaluated at the C2 security level. Adaptive Server release 11.5 includes all the security features of the SQL Server **evaluated configuration**.

Audience

This manual is intecdded for System Adinistrators, System Security Officers, and Database Owners. It is not addressed to the general, non-administrative user with no special roles or privileges. The *Security Features User's Guide*, which is addressed to all users of Adaptive Server, provides instructions and guidelines for using the server in a secure manner. This manual assumes that you have a general understanding of security and database administration and need to understand and administer the security features provided by Adaptive Server.

How to Use This Book

This manual contains the following chapters:

- Chapter 1, "Overview of Security Features," provides an overview of the security features that are available with Adaptive Server and directs you to the chapter where each feature is discussed in detail. It also provides an overall process for administering security and shows you where to read about each task.
- Chapter 2, "Getting Started After Installation," provides guidelines for setting up a secure operating environment after you have installed Adaptive Server.
- Chapter 3, "Managing Adaptive Server Logins and Database Users," describes methods for administering Adaptive Server login accounts and controlling user access to databases.

- Chapter 4, “Administering Roles,” describes how to administer system and user-defined roles.
- Chapter 5, “Managing User Permissions,” describes the discretionary access control (DAC) functionality, including how to specify whether users or groups can access particular database objects and which commands or database operations are available to individual users.
- Chapter 6, “Using Proxy Authorization,”describes how to administer and use proxy authorization. Proxy authorization allows a user to impersonate another user server-wide.
- Chapter 7, “Managing Remote Servers,” discusses the security implications of remote procedure calls (RPCs) and how to manage user access to procedures on remote servers.
- Chapter 8, “Auditing,” explains how to record security-related system activity in an audit trail, how to use the audit trail to detect unauthorized access, and how to maintain the audit trail.
- Appendix A, “Single-Table Auditing,” discusses how to set up auditing when you use only a single audit table.

Adaptive Server Enterprise Documents

The following documents comprise the Sybase Adaptive Server Enterprise documentation:

- The *Release Bulletin* for your platform – contains last-minute information that was too late to be included in the books.

  A more recent version of the *Release Bulletin* may be available on the World Wide Web. To check for critical product or document information that was added after the release of the product CD, use SyBooks™-on-the-Web.
- The Adaptive Server installation documentation for your platform – describes installation and upgrade procedures for all Adaptive Server and related Sybase products.
- The Adaptive Server configuration documentation for your platform – describes configuring a server, creating network connections, configuring for optional functionality, such as auditing, installing most optional system databases, and performing operating system administration tasks.
- *What’s New in Adaptive Server Enterprise?* – describes the new features in Adaptive Server release 11.5, the system changes

added to support those features, and the changes that may affect your existing applications.

- *Navigating the Documentation for Adaptive Server* – an electronic interface for using Adaptive Server. This online document provides links to the concepts and syntax in the documentation that are relevant to each task.
- *Transact-SQL User's Guide* – documents Transact-SQL, Sybase's enhanced version of the relational database language. This manual serves as a textbook for beginning users of the database management system. This manual also contains descriptions of the *pubs2* and *pubs3* sample databases.
- *System Administration Guide* – provides in-depth information about administering servers and databases. This manual includes instructions and guidelines for managing physical resources and user and system databases, and specifying character conversion, international language, and sort order settings.
- *Adaptive Server Reference Manual* – contains detailed information about all Transact-SQL commands, functions, procedures, and datatypes. This manual also contains a list of the Transact-SQL reserved words and definitions of system tables.
- *Performance and Tuning Guide* – explains how to tune Adaptive Server for maximum performance. This manual includes information about database design issues that affect performance, query optimization, how to tune Adaptive Server for very large databases, disk and cache issues, and the effects of locking and cursors on performance.
- The *Utility Programs* manual for your platform – documents the Adaptive Server utility programs, such as **isql** and **bcp**, which are executed at the operating system level.
- *Security Administration Guide* – explains how to use the security features provided by Adaptive Server to control user access to data. This manual includes information about how to add users to Adaptive Server, administer both system and user-defined roles, grant database access to users, and manage remote Adaptive Servers.
- *Security Features User's Guide* – provides instructions and guidelines for using the security options provided in Adaptive Server from the perspective of the non-administrative user.

- *Error Messages* and *Troubleshooting Guide* – explains how to resolve frequently occurring error messages and describes solutions to system problems frequently encountered by users.
- *Component Integration Services User's Guide for Adaptive Server Enterprise and OmniConnect* – explains how to use the Adaptive Server Component Integration Services feature to connect remote Sybase and non-Sybase databases.
- *Adaptive Server Glossary* – defines technical terms used in the Adaptive Server documentation.
- *Master Index for Adaptive Server Publications* – combines the indexes of the *Adaptive Server Reference Manual*, *Component Integration Services User's Guide*, *Performance and Tuning Guide*, *Security Administration Guide*, *Security Features User's Guide*, *System Administration Guide*, and *Transact-SQL User's Guide*.

Other Sources of Information

Use the SyBooks™ and SyBooks-on-the-Web online resources to learn more about your product:

- SyBooks documentation is on the CD that comes with your software. The DynaText browser, also included on the CD, allows you to access technical information about your product in an easy-to-use format.

  Refer to *Installing SyBooks* in your documentation package for instructions on installing and starting SyBooks.
- SyBooks-on-the-Web is an HTML version of SyBooks that you can access using a standard Web browser.

  To use SyBooks-on-the-Web, go to http://www.sybase.com, and choose Documentation.

Conventions

The following sections describe conventions used in this manual.

Formatting SQL Statements

SQL is a free-form language: there are no rules about the number of words you can put on a line, or where you must break a line. However, for readability, all examples and syntax statements in this manual are formatted so that each clause of a statement begins on a new line. Clauses that have more than one part extend to additional lines, which are indented.

SQL Syntax Conventions

- Syntax statements (displaying the syntax and all options for a command) appear as follows:

  ```
  sp_dropdevice [device_name]
  ```

  or, for a command with more options:

  ```
  select column_name
      from table_name
      where search_conditions
  ```

  In syntax statements, command keywords are in normal font. Identifiers are in lowercase, normal font for keywords, and italics for user-supplied words.
- Examples showing the use of Transact-SQL commands are printed like this:

  ```
  select * from publishers
  ```
- Examples of output from the computer are printed like this:

  ```
  pub_id  pub_name              city       state
  ------- --------------------- --------   ----
  0736    New Age Books         Boston     MA
  0877    Binnet & Hardley      Washington DC
  1389    Algodata Infosystems  Berkeley   CA

  (3 rows affected)
  ```

Table 1 describes the conventions for syntax statements used in this manual.

Table 1: Font and syntax conventions for this manual

Element	Example
Command names, command option names, utility names, utility flags, and other keywords are **bold**.	**select** **sp_configure**
Database names, datatypes, file names and path names are in *italics*.	*master* database

Table 1: Font and syntax conventions for this manual (continued)

Element	Example
Variables, or words that stand for values that you fill in, are in *italics*.	`select` *column_name* `from` *table_name* `where` *search_conditions*
Parentheses must be typed as part of the command.	`compute` *`row_aggregate`* `(`*`column_name`*`)`
Curly braces indicate that you must choose at least one of the enclosed options. Do not type the braces.	`{cash, check, credit}`
The vertical bar separating options within curly braces indicate that you may choose only one of the enclosed options.	`{extra_cheese | avocados | sour_cream}`
Square brackets indicate that choosing one or more of the enclosed options is optional. Do not type the brackets.	`[anchovies]`
The comma separating options within square brackets indicates that you may choose as many of the enclosed options as you want. Separate your choices with commas typed as part of the command.	`[extra_cheese, avocados, sour_cream]`

An ellipsis (...) indicates that you may repeat the last choice or group of choices as many times as you like.	`buy thing = price [cash | check | credit]`
	` [, thing = price [cash | check | credit]]...`
	In this example, you must buy at least one thing and give its price. You may choose a method of payment: one of the items enclosed in square brackets. You may also choose to buy additional things: as many of them as you want. For each thing you buy, give its name, its price, and (optionally) a method of payment.

Case

In this manual, most of the examples are in lowercase. However, you can disregard case when typing Transact-SQL keywords. For example, **SELECT**, **Select**, and **select** are the same.

Note that Adaptive Server's sensitivity to the case of database objects, such as table names, depends on the sort order installed on Adaptive Server. You can change case sensitivity for single-byte character sets by reconfiguring the Adaptive Server sort order. See Chapter 13, "Configuring Character Sets, Sort Orders, and Languages," in the *System Administration Guide* for more information.

Expressions

Several different types of expressions are used in SQL Server syntax statements.

Table 2: Types of expressions used in syntax statements

Usage	Definition
expression	Can include constants, literals, functions, column identifiers, variables, or parameters
logical expression	An expression that returns TRUE, FALSE, or UNKNOWN
constant expression	An expression that always returns the same value, such as “5+3” or “ABCDE”
float_expr	Any floating-point expression or expression that implicitly converts to a floating value
integer_expr	Any integer expression, or an expression that implicitly converts to an integer value
numeric_expr	Any numeric expression that returns a single value
char_expr	An expression that returns a single character-type value
binary_expression	An expression that returns a single binary or varbinary value

If You Need Help

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you have any questions about this installation or need assistance during the installation process, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.

1 Overview of Security Features

This chapter provides an overview of the security features available in Adaptive Server. Topics include:

- Security Features Available in Adaptive Server 1-1
- Discretionary Access Controls 1-2
- Identification and Authentication Controls 1-3
- Division of Roles 1-3
- Auditing 1-6
- General Process of Security Administration 1-6

Security Features Available in Adaptive Server

SQL Server release 11.0.6 passed the security evaluation by the National Security Agency (NSA) at the Class C2 criteria. (The requirements for the C2 criteria are given by the Department of Defense in DOD 52.00.28-STD, *Department of Defense Trusted Computer System Evaluation Criteria* [TCSEC], also known as the "Orange Book.")

The configuration of SQL Server release 11.0.6 that was evaluated at the C2 security level by the NSA in 1996 on the HP 9000 HP-UX BLS, 9.09+ platform is referred to as the **evaluated configuration**. Certain features of SQL Server, such as remote procedures and direct updates to system tables, were excluded from the evaluated configuration. Notes in the Adaptive Server documentation indicate particular features that were not included in the evaluated configuration. For a complete list of features that were excluded from the evaluated configuration, see Appendix A in the *SQL Server Installation and Configuration Guide for HP 9000 HP-UX BLS, 9.09+*.

Adaptive Server release 11.5 contains all of the security features included in SQL Server release 11.0.6 plus some new security features. Table 1-1 summarizes the major features.

Table 1-1: Major security features

Security Feature	Description
Discretionary Access Controls (DAC)	Provides access controls that give object owners the ability to restrict access to objects, usually with the **grant** and **revoke** commands. This type of control is dependent upon an object owner's discretion.
Identification and authentication controls	Ensures that only authorized users can log into the system.
Division of roles	Allows you to grant privileged roles to specified users so that only designated users can perform certain tasks. Adaptive Server has predefined roles, called "system roles," such as **System Administrator** and **System Security Officer**. In addition, Adaptive Server allows System Security Officers to define additional roles, called "user-defined roles."
Auditing	Provides the capability to audit events such as logins, logouts, server boot operations, remote procedure calls, accesses to database objects, and all actions by a specific user or with a particular role active. In addition, Adaptive Server provides a single option to audit a set of server-wide security-relevant events.

The following sections provide an overview of each major feature and refer you to the chapter where the feature is covered in detail.

Discretionary Access Controls

Owners of objects can grant access to those objects to other users. The object owners can also grant other users the ability to pass the access permission to other users. With Adaptive Server's discretionary access controls, you can give various kinds of permissions to users, groups, and roles with the **grant** command. The **revoke** command permits you to rescind these permissions. The **grant** and **revoke** commands give users permission to execute specified commands and to access specified tables, views, and columns.

Some commands can be used at any time by any user, with no permission required. Others can be used only by users of a certain status such as a System Administrator and are not transferable.

The ability to assign permissions for the commands that can be granted and revoked is determined by each user's status (as System

Administrator, Database Owner, or database object owner), and by whether or not a particular user has been granted a permission with the option to grant that permission to other users.

Discretionary access controls are discussed in Chapter 5, “Managing User Permissions.”

Identification and Authentication Controls

A user of Adaptive Server is given a login account with a unique ID. All of that user’s activity on the server can be attributed to his or her server user ID and audited.

Adaptive Server passwords are stored in the *master..syslogins* table in encrypted form. When you log into Adaptive Server from a client, you can choose client-side password encryption to encrypt your password before sending it over the network.

A System Security Officer can grant a user the ability to impersonate another user in the server. This ability, called **proxy authorization**, allows administrators to check permissions for a particular user or to perform maintenance on a user’s database objects. Application servers can log into the server and execute procedures and commands on behalf of several users.

Identification and authentication controls are discussed in Chapter 3, “Managing Adaptive Server Logins and Database Users”. In addition, see Chapter 6, “Using Proxy Authorization” and Chapter 7, “Managing Remote Servers.”

Division of Roles

An important feature in Adaptive Server is the division of **roles**. The roles supported by Adaptive Server enable you to enforce and maintain individual accountability. Adaptive Server provides system roles, such as System Administrator and System Security Officer, and user-defined roles, which are created by a System Security Officer.

System Roles

Various security-related, administrative, and operational tasks are grouped into the following system roles:

- **System Administrator**, whose tasks include:

  - Managing disk storage
  - Monitoring Adaptive Server's automatic recovery procedure
  - Fine-tuning Adaptive Server by changing configurable system parameters
  - Diagnosing system problems and reporting them as appropriate
  - Backing up and loading databases
  - Granting and revoking the System Administrator role
  - Modifying and dropping server login accounts
  - Granting permissions to Adaptive Server users
  - Creating user databases and granting ownership of them
  - Setting up groups (which are convenient in granting and revoking permissions)
- **System Security Officer**, who performs security-related tasks such as:
  - Creating server login accounts, which includes assigning initial passwords
  - Changing the password of any account
  - Granting and revoking the System Security Officer and Operator roles
  - Creating, granting, and revoking user-defined roles
  - Granting the capability to impersonate another user throughout the server
  - Setting the password expiration interval
  - Managing the audit system
- **Operator**, a user who can back up and load databases on a server-wide basis. The Operator role allows a single user to use the **dump database, dump transaction, load database**, and **load transaction** commands to back up and restore all databases on a server without having to be the owner of each one.

These roles provide individual accountability for users performing operational and administrative tasks. Their actions can be audited and attributed to them.

User-Defined Roles

Adaptive Server enables a System Security Officer to create roles and grant them to users, groups of users, or other roles. For example, a System Security Officer might create the role "financial_analyst" and grant that role to users in the installation who perform financial analyst tasks. Similarly, the role "salary_administrator" might be created and assigned to those who handle payroll for your organization.

Role Hierarchies and Mutual Exclusivity

A System Security Officer can define:

- Role hierarchies – roles that contain other roles
- Mutual exclusivity of roles – roles that are independent of each other

Role Hierarchy

A System Security Officer can define role hierarchies such that if a user has one role, the user automatically has roles lower in the hierarchy. For example, the "chief_financial_officer" role might contain both the "financial_analyst" and the "salary_administrator" roles. The Chief Financial Analyst can perform all tasks and see all data that can be viewed by the Salary Administrators and Financial Analysts.

Mutual Exclusivity

Two roles can be defined to be mutually exclusive for:

- Membership – a single user cannot be granted both roles. For example, an installation might not want a single user to have both the "payment_requestor" and "payment_approver" roles to be granted to the same user.
- Activation – a single user cannot activate, or enable, both roles. For example, a user might be granted both the "senior_auditor" and the "equipment_buyer" roles, but the installation may not want to permit the user to have both roles enabled at the same time.

System roles, as well as user-defined roles, can be defined to be in a role hierarchy or to be mutually exclusive. For example, you might want a “super_user” role to contain the System Administrator, Operator, and “Tech Support” roles. In addition, you might want to define the System Administrator and System Security Officer roles to be mutually exclusive for membership; that is, a single user cannot be granted both roles.

Administering and using roles is discussed in Chapter 4, “Administering Roles.”

Auditing

A comprehensive audit system is provided with Adaptive Server. The audit system consists of a system database called *sybsecurity*, configuration parameters for managing auditing, a system procedure, **sp_audit**, to set all auditing options, and a system procedure, **sp_addauditrecord**, to add user-defined records to the audit trail. When you install auditing, you can specify the number of audit tables that Adaptive Server will use for the audit trail. If you use two or more audit tables to store the audit trail, you can set up a smoothly running audit system with no manual intervention and no loss of records.

A System Security Officer manages the audit system and is the only user who can start and stop auditing, set up auditing options, and process the audit data. As a System Security Officer, you can establish auditing for events such as:

- Server-wide, security-relevant events
- Creating, deleting, and modifying database objects
- All actions by a particular user or all actions by users with a particular role active
- Granting or revoking database access
- Importing or exporting data
- Logins and logouts

Auditing functionality is discussed in Chapter 8, “Auditing.”

General Process of Security Administration

Table 1-2 describes the major tasks that are required to administer Adaptive Server in a secure manner and refers you to the documentation that contains the instructions for performing each task.

Table 1-2: General process for security administration

Task	Description	See
1. Install Adaptive Server, including auditing.	This task includes preparing for installation, loading files from your distribution medium, performing the actual installation, and getting started administering the physical resources that are required.	The installation documentation for your platform
2. Set up a secure administrative environment.	Immediately after installation, set up a secure administrative environment for Adaptive Server. This includes enabling auditing, granting roles to individual users to ensure individual accountability, and assigning login names to System Administrators and System Security Officers.	Chapter 2, “Getting Started After Installation”
3. Determine the physical resources required for your user databases and create the databases.	This task includes making storage management decisions, initializing database devices, creating and using segments, and creating user databases.	*System Administration Guide*
4. Add user logins to the server and add users to databases.	Involves adding logins, creating groups, adding users to databases, dropping and locking logins, and assigning initial passwords.	Chapter 3, “Managing Adaptive Server Logins and Database Users”
5. Establish and assign special roles to users.	Includes assigning system roles to individual users, creating user-defined roles, defining role hierarchies and mutual exclusivity of roles, and granting roles to users and other roles.	Chapter 4, “Administering Roles”
6. Administer permissions for users, groups, and roles.	Includes granting and revoking permissions to execute certain SQL commands, execute certain system procedures, and access databases, tables, particular table columns, and views.	Chapter 5, “Managing User Permissions”
7. Administer proxy authorizations.	Includes granting permission for proxy authorization and using **set proxy** or **set session authorization** to impersonate another user on a server-wide basis.	Chapter 6, “Using Proxy Authorization”

Table 1-2: General process for security administration (continued)

Task	Description	See
8. Administer the use of remote servers.	Includes establishing and administering the access that is permitted between servers, adding and dropping remote server access, and mapping remote login names to local login names.	Chapter 7, "Managing Remote Servers" and the Adaptive Server installation and configuration documentation for your platform
9. Set up and maintain auditing.	Includes determining what is to be audited, auditing the use of Adaptive Server, and using the audit trail to detect penetration of the system and misuse of resources.	Chapter 8, "Auditing" and the Adaptive Server installation and configuration documentation for your platform

2 Getting Started After Installation

This chapter provides guidelines for what to do to set up security after Adaptive Server has been installed. Topics include:

- Guidelines For Setting Up Security 2-1
- Steps For Setting Up Security 2-2
- An Example of Setting Up Security 2-4

Guidelines For Setting Up Security

Use the guidelines described in the following sections when you set up security on Adaptive Server.

Using the “sa” Login

When Adaptive Server is installed, a single login called “sa” is configured with the **System Administrator** and **System Security Officer** roles. This means that the “sa” login has unlimited power.

Use the “sa” login only during initial setup. Instead of allowing several users to use the “sa” account, establish individual accountability by assigning specific roles to individual administrators.

To establish individual accountability:

1. Use the “sa” login initially to assign the System Security Officer and System Administrator roles to individual users.
2. Lock the “sa” account so that no user has unlimited access.

◆ ***WARNING!***

When logging into Adaptive Server, do not use the -P option of isql to specify your password because another user may have an opportunity to see it.

Changing the “sa” Login Password

The ”sa” login is configured initially with a “NULL” password. Use **sp_password** to change the password immediately after installation.

When To Enable Auditing

Enable auditing early in the administration process so that you have a record of privileged commands that are executed by System Security Officers and System Administrators. You might also want to audit commands that are executed by those with other special roles, such as Operators when they dump and load databases.

Assigning Login Names

Assign Adaptive Server login names that are the same as their respective operating system login names. This makes login to Adaptive Server easier, simplifies management of server and operating system login accounts, and makes it easier to correlate the audit data generated by Adaptive Server with that of the operating system.

Steps For Setting Up Security

After Adaptive Server is installed, use the steps in Table 2-1 to set up security on Adaptive Server.

➤ ***Note***

Table 2-1 provides an overview of the required steps. Read the sections listed in the "See" column for detailed instructions for each step.

For an example of how to use the "sa" account to set up a secure operating environment, see "An Example of Setting Up Security" on page 2-4.

Table 2-1: Steps for getting started

Task	Use	See
1. Log into your operating system.	The process specific to your operating system.	The documentation for your operating system
2. Log into Adaptive Server as "sa". (Both the **sa_role** and **sso_role** are active.)	**isql**	The *Utility Programs* manual for your platform

Table 2-1: Steps for getting started (continued)

Task	Use	See
3. If you want to use auditing, make sure it is installed. After setting up a threshold procedure for the audit trail and determining how to handle the *sybsecurity* transaction log, set audit options and enable auditing.	**sp_audit** to set auditing options **sp_configure** to enable auditing	Chapter 8, "Auditing"
4. Add logins and passwords for System Security Officers, System Administrators, and Operators. (Make the Adaptive Server user's login name the same as the user's operating system login name.)	**sp_addlogin**	Chapter 3, "Managing Adaptive Server Logins and Database Users"
5. Grant **sa_role** to System Administrators, **sso_role** to System Security Officers, and **oper_role** to Operators.	**grant role**	Chapter 4, "Administering Roles"
6. Give access to the *sybsecurity* database to the System Security Officer or Officers who are responsible for administering auditing. Make a System Security Officer the owner of *sybsecurity.* If you have more than one auditing administrator, give permissions for *sybsecurity* to the other auditing administrators.	**sp_changedbowner**	For changing database ownership, see **sp_changedbowner** in the *Adaptive Server Reference Manual*. To add users to databases, see Chapter 3, "Managing Adaptive Server Logins and Database Users." To grant permissions, see Chapter 5, "Managing User Permissions."
7. Verify that you can log into Adaptive Server as each user you configured with the **sa_role** and **sso_role**. Then, lock the "sa" account.	**sp_locklogin**	Chapter 3, "Managing Adaptive Server Logins and Database Users"

Table 2-1: Steps for getting started (continued)

Task	Use	See
8. Check all scripts that may contain the "sa" login name and password and , if necessary, change the login in each script to the name of a user with the correct role. (These can include scripts that perform backups, run **bcp**, or perform **dbcc** checking. The scripts cannot run if they are meant to run as "sa" and that account is locked.)	Any text editor	For information on **bcp**, see the *Utility Programs* manual for your platform. For information on **dbcc**, see Chapter 18, "Checking Database Consistency" in the *System Administration Guide.*

➤ ***Note***

To use the "sa" account for an Adaptive Server upgrade, you must unlock the account. After completing the upgrades, you can again lock the account. For information about how to lock and unlock accounts, see "Locking or Dropping Adaptive Server Login Accounts" on page 3-11. For information about upgrading Adaptive Server, see the installation documentation for your platform.

An Example of Setting Up Security

Suppose you have decided to assign special roles to the users listed in Table 2-2.

Table 2-2: Users to whom you will assign roles

Name	Role	Operating System Login Name
Ralph Smith	**sso_role**	ralph
Kathy Masters	**sa_role**	kathy
Mary Randolph	**sa_role**	mary
Gene Wishing	**oper_role**	gene

Table 2-3 shows an example of the sequence of commands you would use to set up a secure operating environment for Adaptive Server, based upon the role assignments shown in Table 2-2. After logging into the operating system, you would issue these commands using the initial “sa” account.

Table 2-3: Examples of commands used to set up security

Commands	Result
isql -Usa	Logs into Adaptive Server as “sa”. Both **sa_role** and **sso_role** are active.
sp_audit “security”, “all”, “all”, “on” **sp_audit “all”, “sa_role”, “all”, “on”** **sp_audit “all”, “sso_role”, “all”, “on”‘**	Sets auditing options for server-wide, security-relevant events and the auditing of all actions that have **sa_role** or **sso_role** active.
sp_configure “auditing”, 1	Enables auditing. **Note:** Before enabling auditing, set up a threshold procedure for the audit trail and determine how to handle the transaction log in *sybsecurity*. For details, see Chapter 8, “Auditing.”
sp_addlogin ralph, js&2P3d, @fullname ="Ralph Smith" **sp_addlogin kathy, Fr3ds#1, @fullname ="Kathy Masters"** **sp_addlogin mary, mi5pd1s, @fullname ="Mary Randolph"** **sp_addlogin gene, w1seCrkr, @fullname ="Gene Wishing"**	Adds logins and passwords for Ralph, Kathy, Mary, and Gene. A default database is not specified for any of these users, so their default database is *master*.
grant role sso_role to ralph **grant role sa_role to mary** **grant role sa_role to kathy** **grant role oper_role to gene**	Grants the **sso_role** to Ralph, the **sa_role** to Mary and Kathy, and the **oper_role** to Gene.
use sybsecurity **sp_changedbowner ralph**	Grants access to the auditing database, *sybsecurity*, by making Ralph, who is the System Security Officer, the database owner.

Table 2-3: Examples of commands used to set up security (continued)

Commands	Result
sp_locklogin sa,"lock"	Locks the “sa” login so that no one can log in as “sa”. Individuals can assume only the roles that are configured for them. **Note:** Do not lock the “sa” login until you have granted individual users the **sa_role** and **sso_role** roles and have verified that the roles operate successfully.

3 Managing Adaptive Server Logins and Database Users

This chapter describes methods for managing Adaptive Server login accounts and database users. Topics include:

- Adding New Users: An Overview 3-1
- Choosing and Creating a Password 3-2
- Adding Logins to Adaptive Server 3-3
- Creating Groups 3-5
- Adding Users to Databases 3-6
- Adding a "guest" User to a Database 3-7
- Creating Visitor Accounts 3-9
- Adding Remote Users 3-9
- Assigning Roles and Granting Permissions to Users 3-10
- Dropping Users and Groups 3-10
- Locking or Dropping Adaptive Server Login Accounts 3-11
- Changing User Information 3-13
- Using Aliases in Databases 3-17
- Getting Information About Users 3-20
- Getting Information About Usage: Chargeback Accounting 3-23

Adding New Users: An Overview

The process of adding new logins to Adaptive Server, adding users to databases, and granting them **permission** to use commands and database objects is divided among the System Security Officer, System Administrator, and Database Owner.

The process of adding new users consists of the following steps:

1. A System Security Officer creates a server login account for a new user with **sp_addlogin**.
2. A System Administrator or Database Owner adds a user to a database with **sp_adduser**. This command can also give the user an alias or assign the user to a group. Groups are added using **sp_addgroup**. For more information, see "Creating Groups" on page 3-5.

3. A System Administrator, Database Owner, or object owner grants the user or group specific permissions on specific commands and database objects. Users or groups can also be granted permission to grant specific permissions on objects to other users or groups. See Chapter 5, “Managing User Permissions” for detailed information about permissions.

Table 3-1 summarizes the system procedures and commands used for these tasks.

Table 3-1: Adding users to Adaptive Server and databases

Task	Required Role	Command or Procedure	Database
Create new logins, assign passwords, default databases, default language, and full name	System Security Officer	**sp_addlogin**	Any database
Create groups	Database Owner or System Administrator	**sp_addgroup**	User database
Add users to database, assign aliases, and assign groups	Database Owner or System Administrator	**sp_adduser**	User database
Grant groups or users permission to create or access database objects	Database Owner, System Administrator, or object owner	**grant**	User database

Choosing and Creating a Password

Your password is the first line of defense against Adaptive Server access by unauthorized people. When you create your password, choose one that cannot be guessed, following these guidelines:

- Do not use personal information, such as your birthday, street address, or any other number that has anything to do with your personal life.
- Do not use names of pets or loved ones.
- Do not use words that appear in the dictionary or words spelled backwards.

The most difficult passwords to guess are those that combine uppercase and lowercase letters or both numbers and letters. Protecting your password is your responsibility. Never give anyone your password, and never write it down where anyone can see it.

Following these rules to create a password:

- Passwords must be at least 6 bytes long.
- Passwords can consist of any printable letters, numbers, or symbols.
- A password must be enclosed in quotation marks in **sp_addlogin** if it:
    - Includes any character other than A-Z, a-z, 0-9,_, #, valid single-byte or multibyte alphabetic characters, or accented alphabetic characters
    - Begins with a number 0–9

Adding Logins to Adaptive Server

Use the system procedure **sp_addlogin** to add new **login** names to Adaptive Server. It does not give the user permission to access any user databases; the procedure **sp_adduser** gives users access to specific databases. Only the System Security Officer can execute **sp_addlogin**. The syntax is:

```
sp_addlogin loginame, passwd [, defdb]
   [, deflanguage [, fullname]]]
```

where:

- *loginame* is the new user's login name. The login name must follow the rules for identifiers and must be unique on Adaptive Server. To simplify both the login process and server administration, make the Adaptive Server login name the same as the user's operating system login name. This makes logging into Adaptive Server easier, since many client programs use the operating system login name as a default. It also simplifies management of server and operating system login accounts, and makes it easier to correlate usage and audit data generated by Adaptive Server and by the operating system.
- *passwd* is the password for the new user. For guidelines on choosing and creating secure passwords, see "Choosing and Creating a Password" on page 3-2. For information on changing a password, see "Changing Passwords" on page 3-14.
- *defdb* is the name of the new user's **default database**. This is the database where the user starts each session of Adaptive Server.

➤ ***Note***

If you do not specify a user's default database parameter with **sp_addlogin**, the user's default database is *master*. To discourage users from creating database objects in the *master* database, assign a default database other than *master* to most users.

A System Administrator can change anyone's default database with **sp_modifylogin**. Other users can change only their own default database.

After specifying the default database, add the user to the default database with **sp_adduser** so that he or she can log in directly to the default database.

- *deflanguage* is the **default language** in which the user's prompts and messages are displayed. If you omit this parameter, Adaptive Server's default language is used. A System Administrator can change any user's default language with **sp_modifylogin**. Other users can change only their own language.
- *fullname* is the full name of the user. This is useful for documentation and identification purposes. If omitted, no full name is added. A System Administrator can change any user's full name with **sp_modifylogin**. Other users can change only their own full name.

The following statement sets up an account for the user "maryd" with the password "100cents," the default database (*master*), the default language, and no full name:

```
sp_addlogin "maryd", "100cents"
```

The password requires quotation marks because it begins with 1.

After this statement is executed, "maryd" can log into Adaptive Server. She is automatically treated as a "guest" user in *master*, with limited permissions, unless she has been specifically given access to *master*.

The following statement sets up a login account and password ("rubaiyat") for user "omar_khayyam" and makes *pubs2* the default database for this user:

```
sp_addlogin omar_khayyam, rubaiyat, pubs2
```

To specify a full name for a user and also use the default database and language, you must specify **null** in place of the *defdb* and *deflanguage* parameters. For example:

```
sp_addlogin omar, rubaiyat, null, null,
    "Omar Khayyam"
```

Alternatively, you can specify a parameter name, in which case you do not have to specify all the parameters. For example:

```
sp_addlogin omar, rubaiyat,
    @fullname = "Omar Khayyam"
```

When you execute **sp_addlogin**, Adaptive Server adds a row to *master.dbo.syslogins*, assigns a unique server **user ID** (*suid*) for the new user, and fills in other information. When a user logs in, Adaptive Server looks in *syslogins* for the name and password provided by the user. The *password* column is encrypted with a one-way algorithm so it is not human-readable.

The *suid* column in *syslogins* uniquely identifies the user on Adaptive Server. A user's *suid* remains the same, no matter what database he or she is using. The *suid* 1 is always assigned to the default "sa" account that is created when Adaptive Server is installed. Other users' server user IDs are small integers assigned consecutively by Adaptive Server each time **sp_addlogin** is executed.

Creating Groups

Groups provide a convenient way to grant and revoke permissions to more than one user in a single statement. Groups enable you to provide a collective name to a group of users. They are especially useful if you administer an Adaptive Server installation that has a large numbers of users. Every user is a member of the group "public" and can also be a member of one other group. (Users remain in "public", even when they belong to another group.)

It is probably most convenient to create groups before adding users to a database, since the **sp_adduser** procedure can assign users to groups as well as add them to the database.

A System Administrator or the Database Owner can create a group at any time with **sp_addgroup**. The syntax for **sp_addgroup** is:

sp_addgroup ***grpname***

The group name, a required parameter, must follow the rules for identifiers. The System Administrator can assign or re-assign users to groups with **sp_changegroup**.

To set up the Senior Engineering group, use the following command while using the database to which you want to add the group:

```
sp_addgroup senioreng
```

The **sp_addgroup** system procedure adds a row to *sysusers* in the current database. Therefore, each group in a database, as well as each user, has an entry in *sysusers*.

Adding Users to Databases

The Database Owner or a System Administrator can use the system procedure **sp_adduser** to add a user to a specific database. The user must already have an Adaptive Server login. The syntax is:

```
sp_adduser loginame [, name_in_db [, grpname]]
```

where:

- *loginame* is the login name of an existing user.
- *name_in_db* specifies a name that is different from the login name by which the user is to be known inside the database.

  This feature is supplied to accommodate users' preferences. For example, if there are five Adaptive Server users named Mary, each of them must have a different login name. Mary Doe might log in as "maryd", Mary Jones as "maryj", and so on. However, if these users do not use the same databases, each might prefer to be known simply as "mary" inside a particular database.

  If no *name_in_db* parameter is given, the name inside the database is the same as *loginame*.

➤ ***Note***

This capability is different from the alias mechanism described in "Using Aliases in Databases" on page 3-17, which maps the identity and permissions of one user to another.

- *grpname* is the name of an existing group in the database. If you do not specify a group name, the user is made a member of the default group "public". Users remain in "public" even if they are a member of another group. See "Changing a User's Group Membership" on page 3-16 for information about modifying a user's group membership.

The **sp_adduser** system procedure adds a row to the *sysusers* system table in the current database. When a user has an entry in the *sysusers* table of a database, he or she:

- Can issue the **use *database_name*** command to access that database

- Will use that database by default, if the default database parameter was issued as part of **sp_addlogin**
- Can use **sp_modifylogin** to make that database the default

The following example shows how a database owner could give access permission to “maryh” of the engineering group “eng”, which was created already:

```
sp_adduser maryh, mary, eng
```

The following example shows how to give “maryd” access to a database, keeping her name in the database the same as her login name:

```
sp_adduser maryd
```

The following example shows how to add the user “maryj” and put her in the existing “eng” group, keeping her name in the database the same as her login name by using **null** in place of a new user name:

```
sp_adduser maryj, null, eng
```

Users who have access to a database still need permissions to do things inside the database such as read data, modify data, and use certain commands. These permissions are granted by the owners of the database objects or user databases with the **grant** and **revoke** commands. These commands are discussed in Chapter 5, “Managing User Permissions.”

Adding a “guest” User to a Database

Creating a user named “guest” in a database enables any user with an Adaptive Server account to access the database as a **guest** user. If a user issues the **use** *database_name* command, and his or her name is not found in the database’s *sysusers* or *sysalternates* table, Adaptive Server looks for a guest user. If there is one, the user is allowed to access the database, with the permissions of the guest user.

The Database Owner can add a guest entry to the *sysusers* table of the database with the system procedure **sp_adduser**:

```
sp_adduser guest
```

The guest user can be removed with **sp_dropuser**, as discussed in “Dropping Users” on page 3-10.

If you drop the guest user from the *master* database, server users who have not yet been added to any databases will be unable to log into Adaptive Server.

➤ ***Note***

Although more than one individual can be a guest user in a database, you can still maintain individual accountability by auditing the database operations that are performed by a guest user. Adaptive Server maintains individual accountability by using the user’s server user ID, which is unique within the server, to audit each user’s activity. This ID is stored in the *suid* column of the *master..syslogins* table. For more information about auditing, see Chapter 8, “Auditing.”

“guest” User Permissions

When you first add the user name “guest” to a database, “guest” inherits the privileges of “public”. The Database Owner and the owners of database objects can change these permissions as they want, using **grant** and **revoke**, to make the privileges of “guest” either more restrictive or less restrictive than those of “public.” See Chapter 5, “Managing User Permissions,” for a description of the “public” privileges.

When you install Adaptive Server, *master..sysusers* contains a guest entry. The installation script for the *pubs2* database also contains a guest entry for its *sysusers* table.

“guest” User in User Databases

In user databases, the Database Owner is responsible for setting up any guest mechanisms that are needed. Adding a guest user to a user database allows an owner to permit all Adaptive Server users to use that database without having to use **sp_adduser** to explicitly name each one as a database user.

You can use the guest mechanism to restrict access to database objects while allowing access to the database.

For example, the owner of the *titles* table could grant **select** permission on *titles* to all database users except “guest” by executing the following commands:

```
grant select on titles to public
sp_adduser guest
revoke all on titles from guest
```

“guest” User in *pubs2* and *pubs3*

The “guest” user entry in *pubs2* and *pubs3*, the sample databases, allows new Adaptive Server users to follow the examples in the *Transact-SQL User’s Guide*. The guest in the sample databases is given a wide range of privileges, including:

- **select** permission and data modification permission on all of the user tables
- **execute** permission on all of the procedures
- **create table, create view, create rule, create default** and **create procedure** permissions

Creating Visitor Accounts

As another method of accommodating visiting users on Adaptive Server, the System Security Officer can use **sp_addlogin** to enter a row in *master..syslogins* with a login name and password that visiting users are instructed to use. Typically, such users are granted restricted permissions. A default database may be assigned.

◆ ***WARNING!***

A visitor user account is not the same as the “guest” user account. All users of the visitor account have the same server user ID; therefore, you cannot audit individual activity. Each “guest” user has a unique server ID, so you can audit individual activity and maintain individual accountability. Setting up a visitor account to be used by more than one user is not recommended because you lose individual accountability.

Adding Remote Users

You can allow users on another Adaptive Server to execute stored procedures on your server by enabling remote access. Working with a System Administrator of the remote server, you can also allow users of your server to execute **remote procedure calls** to the remote server.

To enable remote procedure calls, both the local and the remote server must be configured, following several steps. For information about setting up remote servers and adding remote users, see Chapter 7, “Managing Remote Servers.”

➤ ***Note***

Remote users and remote procedure calls are not included in the evaluated configuration.

Assigning Roles and Granting Permissions to Users

The final steps in adding database users are:

- Assigning them special roles, as required. For details, see Chapter 4, “Administering Roles.”
- Granting them permission to use commands and database objects. See Chapter 5, “Managing User Permissions.”

Dropping Users and Groups

The following system procedures allow a System Administrator or Database Owner to drop users and groups.

Table 3-2: Dropping users and groups

Task	Required Role	System Procedure	Database
Drop user from database	Database Owner or System Administrator	**sp_dropuser**	User database
Drop group from database	Database Owner or System Administrator	**sp_dropgroup**	User database

Dropping Users

A Database Owner or a System Administrator can use the system procedure **sp_dropuser** to deny an Adaptive Server user access to the database in which **sp_dropuser** is executed. (If a “guest” user is defined in that database, the user can still access that database as “guest”.)

The syntax is:

```
sp_dropuser name_in_db
```

where *name_in_db* is usually the login name, unless another name has been assigned.

You cannot drop a user who owns objects. Since there is no command to transfer ownership of objects, you must drop objects owned by a user before you drop the user with **sp_dropuser.** If you want to deny access to a user who owns objects, use **sp_locklogin** to lock his or her account.

You also cannot drop a user who has granted permissions to other users. **sp_dropuser** displays an informational message when you attempt to do this. Use **revoke with cascade** to revoke permissions from all users who were granted permissions by the user to be dropped, then drop the user. You must then grant permissions to the users again, if appropriate.

Dropping Groups

Use **sp_dropgroup** to drop a group. The syntax is:

```
sp_dropgroup grpname
```

You cannot drop a group that has members. If you try to do so, the error report displays a list of the members of the group you are attempting to drop. To remove users from a group, execute **sp_changegroup**, discussed in "Changing a User's Group Membership" on page 3-16.

Locking or Dropping Adaptive Server Login Accounts

Locking an Adaptive Server login account and dropping a login account both prevent the user from logging into Adaptive Server. However, locking a login is safer than dropping it because locking a login account maintains the *suid* so that it cannot be reused.

◆ ***WARNING!***

When a login is dropped, the *suid* can be reused, thus compromising accountability and security.

Adaptive Server may reuse the server user ID (*suid*) of a dropped login account when the next login account is created. This occurs only when the dropped login holds the highest *suid* in *syslogins*; however, it can compromise accountability if execution of **sp_droplogin** is not being audited. Also, it is possible for a user with the reused *suid* to access database objects that were authorized for the old *suid*.

You cannot drop a login when:

- The user is in any database
- The login belongs to the last remaining System Security Officer or System Administrator

For more information about locking logins, see “Locking or Dropping Adaptive Server Login Accounts” on page 3-11. For more information about dropping login accounts, see “Dropping Login Accounts” on page 3-13.

Table 3-3: Locking or dropping login accounts

Task	Required Role	System Procedure	Database
Lock login account, which maintains the suid so that it cannot be reused	System Administrator or System Security Officer	**sp_locklogin**	*master*
Drop login account, which allows reuse of *suid*	System Administrator	**sp_droplogin**	*master*

Locking and Unlocking Login Accounts

Use the system procedure **sp_locklogin** to lock and unlock accounts or to display a list of locked accounts. You must be a System Administrator or a System Security Officer to use **sp_locklogin** to lock or unlock an account.

The syntax is:

```
sp_locklogin [loginame, "{lock | unlock}"]
```

where:

- *loginame* is the name of the account to be locked or unlocked. It must be an existing, valid account.
- **lock** | **unlock** specifies whether the account is to be locked or unlocked.

To display a list of all locked logins use **sp_locklogin** with no parameters.

You can lock an account that is currently logged in. When you do this, Adaptive Server displays a warning that the user is not locked out of the account until he or she logs out. You can lock the account of a Database Owner, and a locked account can own objects in databases. In addition, you can use **sp_changedbowner** to specify a locked account as the owner of a database.

Adaptive Server ensures that there is always at least one unlocked System Security Officer's account and one unlocked System Administrator's account. When you attempt to lock the last unlocked account of a System Administrator or a System Security Officer, Adaptive Server displays an error message and does not lock the account.

Dropping Login Accounts

A System Administrator can use the system procedure **sp_droplogin** to deny a user access to Adaptive Server. The syntax is:

```
sp_droplogin loginame
```

If the login to be dropped is for a user in any database on the server, **sp_droplogin** fails. Use **sp_dropuser** to drop the user from a database. You cannot drop a user from a database if that user owns any objects in the database. For more information, see "Dropping Users" on page 3-10.

Changing User Information

Table 3-4 lists the system procedures you use to change passwords, default database, default language, full name or group assignment.

Table 3-4: System procedures for changing user information

Task	Required Role	System Procedure	Database
Change your password	None	**sp_password**	Any database
Change another user's password	System Security Officer	**sp_password**	Any database
Change your default database, default language, or full name	None	**sp_modifylogin**	Any database
Change a login account's default database, default language, or full name	System Administrator	**sp_modifylogin**	Any database
Change the group assignment of a user	System Administrator, Database Owner	**sp_changegroup**	User database

Changing Passwords

All users can change passwords at any time with the system procedure **sp_password**. The System Security Officer can use this system procedure to change any other user's password. The syntax is:

sp_password *caller_passwd*, *new_passwd* [, *loginame*]

where:

- *caller_passwd* is the password of the login account that is currently executing **sp_password**. When you are changing your own password, this is your old password. When a System Security Officer uses **sp_password** to change another user's password, *caller_passwd* is the password of the System Security Officer.
- *new_passwd* is the new password for the user executing **sp_password**, or for the user indicated by *loginame*. For guidelines on choosing and creating secure passwords, see "Choosing and Creating a Password" on page 3-2.
- *loginame* may only be used by a System Security Officer to change another user's password.

For example, a user can change a password from "3blindmice" to "2mediumhot" with the following command:

```
sp_password "3blindmice", "2mediumhot"
```

These passwords are enclosed in quotes because they begin with numbers.

In the following example, the System Security Officer whose password is "2tomato" changes Victoria's password to "sesame1":

```
sp_password "2tomato", sesame1, victoria
```

Requiring New Passwords

Your site may choose to use the **systemwide password expiration** configuration parameter to establish a password expiration interval, which forces all Adaptive Server users to change their passwords on a regular basis. For information about how to set the configuration parameter, see Chapter 11, "Setting Configuration Parameters," in the *System Administration Guide*. Even if you do not use **systemwide password expiration**, it is important, for security reasons, that users change their passwords periodically.

The column *pwdate* in the *syslogins* table records the date of the last password change. The following query selects all login names whose passwords have not changed since September 15, 1997:

```
select name, pwdate
from syslogins
where pwdate < "Sep 15 1997"
```

Null Passwords

You may not assign a null password. However, when Adaptive Server is installed, the default “sa” account has a null password. The following example shows how to change a null password to a valid one:

```
sp_password null, "8M4LNCH"
```

Note that “null” is not enclosed in quotes in the statement.

Changing User Defaults

Any user can use the **sp_modifylogin** system procedure to change his or her default database, default language, or full name. A System Administrator can change these settings for any user. The syntax is:

sp_modifylogin ***account, column, value***

- *account* is the name of the user whose account you are modifying.
- *column* specifies the option that you are changing. The options are:
  - **defdb** – The “home” database to which the user is connected when he or she logs in.
  - **deflanguage** – The official name of the user’s default language, as stored in *master..syslanguages*.
  - **fullname** – The user’s full name.
- *value* is the new value for the specified option.

After you execute **sp_modifylogin** to change the default database, the user is connected to the new default database the next time he or she logs in. However, **sp_modifylogin** does not automatically give the user access to the database. Unless the Database Owner has set up access with **sp_adduser**, **sp_addalias**, or with a guest user mechanism, the user is connected to *master* even after his or her default database has been changed.

The following example changes the default database for "anna" to *pubs2*:

```
sp_modifylogin anna, defdb, pubs2
```

The next example changes the default language for "claire" to French:

```
sp_modifylogin claire, deflanguage, french
```

The following example changes the full name for "mtwain" to "Samuel Clemens."

```
sp_modifylogin mtwain, fullname, "Samuel Clemens"
```

Changing a User's Group Membership

A System Administrator or the Database Owner can use the system procedure **sp_changegroup** to change a user's group affiliation. At any one time, each user can be a member of only one group other than "public," of which all users are always members. When new users are added to the database, they are assigned to the group "public" as well as any group specified with the **sp_adduser** procedure.

Before you execute **sp_changegroup**:

- The group must exist. (Use **sp_addgroup** to create a group.)
- The user must have access to the current database (must be listed in *sysusers*).

The syntax for **sp_changegroup** is:

```
sp_changegroup grpname, username
```

For example, to change the user "jim" from his current group to the group "manage", use the following command:

```
sp_changegroup manage, jim
```

To remove a user from a group without assigning the user to another group, you must change the group affiliation to "public":

```
sp_changegroup "public", jim
```

The name "public" must be in quotes because it is a reserved word. All users are always members of "public". This command reduces Jim's group affiliation to "public" only.

When a user changes from one group to another, the user loses all permissions that he or she had as a result of belonging to the old group, but gains the permissions that have been granted to the new group.

The assignment of users into groups can be changed at any time.

Using Aliases in Databases

The alias mechanism allows you to treat two or more users as the same user inside a database so that they all have the same privileges. This mechanism is often used so that more than one user can assume the role of Database Owner. A Database Owner the capability to use the **setuser** command to impersonate another user in the database. The alias mechanism can also be used to set up a collective user identity, within which the identities of individual users can be traced by auditing their activities.

➤ ***Note***

Although more than one individual can use the alias in a database, you can still maintain individual accountability by auditing the database operations performed by each user. Adaptive Server maintains individual accountability by using the user's server user ID, which is unique within the server, to audit each user's activity. This ID is stored in the *suid* column of the *master..syslogins* table. For more information about auditing, see Chapter 8, "Auditing."

For example, suppose that several vice presidents want to use a database with identical privileges and ownerships. One way to accomplish this is to add the login "vp" to Adaptive Server and the database; each vice president logs in as "vp". The problem with this method is that there is no way to tell the individual users apart. The other approach is to alias all the vice presidents, each of whom has his or her own Adaptive Server account, to the database user name "vp".

The following system procedures are used to manage aliases:

Table 3-5: System procedures for managing aliases

Task	Require Role	System Procedure	Database
Add an alias for a user	Database Owner or System Administrator	**sp_addalias**	User database
Drop an alias	Database Owner or System Administrator	**sp_dropalias**	User database

Adding Aliases

Use **sp_addalias** to add an alias for a user. The syntax is:

```
sp_addalias loginame, name_in_db
```

where:

loginame is the name of the user who wants an alias in the current database. This user must have an account in Adaptive Server but cannot be a user in the current database.

name_in_db is the name of the database user to whom the user specified by *loginame* is to be linked. The *name_in_db* name must exist in both *master..syslogins* and in *sysusers* in the current database.

Executing **sp_addalias** maps the user name specified by *loginame* to the user name specified by *name_in_db*. It does this by adding a row to the system table *sysalternates*.

When a user tries to use a database, Adaptive Server checks for the user's server user ID number (*suid*) in *sysusers*. If it is not found, Adaptive Server then checks *sysalternates*. If the user's *suid* is found there, and it is mapped to a database user's *suid*, the first user is treated as the second user while the first user is using the database.

For example, suppose that Mary owns a database. She wants to allow both Jane and Sarah to use the database as if they were its owner. Jane and Sarah have logins on Adaptive Server but are not authorized to use Mary's database. Mary executes the following commands:

```
sp_addalias jane, dbo
exec sp_addalias sarah, dbo
```

Now both Jane and Sarah can access Mary's database and Adaptive Server reacognizes each as the owner of the database.

◆ ***WARNING!***

Users who are aliased to the database owner have all the permissions and can perform all the actions that can be performed by the real database owner, with respect to the database in question. Before a database owner aliases another user to be database owner, he or she should carefully consider the implications of vesting another user with full access to that database.

Dropping Aliases

Use the system procedure **sp_dropalias** to drop the mapping of an alternate *suid* to a user ID. Doing this deletes the relevant row from *sysalternates*. The syntax is:

```
sp_dropalias loginame
```

where *loginame* is the name of the user specified by *loginame* when the name was mapped with **sp_addalias**. After a user's alias is dropped, the user no longer has access to the database, unless his or her name is then added to *sysusers* (with **sp_adduser**) or the database has a "guest" user in *sysusers*.

Getting Information About Aliases

To display information about aliases, use the system procedure **sp_helpuser**. For example, to find the aliases for "dbo", execute:

```
              sp_helpuser dbo
Users_name     ID_in_db    Group_name   Login_name
----------     --------    ----------   ----------
dbo            1           public       sa

(1 row affected)
              Users aliased to user.
              Login_name
              ----------------------
              andy
              christa
              howard
              linda

              (4 rows affected)
```

Getting Information About Users

The following procedures allow users to obtain information about users, groups, and current Adaptive Server usage.

Table 3-6: Reporting information about Adaptive Server users and groups

Task	System Procedure
Report current Adaptive Server users and processes	**sp_who**
Display information about login accounts	**sp_displaylogin**
Report users and aliases in a database	**sp_helpuser**
Report groups within a database	**sp_helpgroup**

Getting Reports on Users and Processes

Use the system procedure **sp_who** to report information about current users and processes on Adaptive Server. The syntax is:

sp_who [*loginame* | "*spid*"]

where:

- *loginame* is the user's Adaptive Server login name. If you give a login name, **sp_who** reports information about processes being run by the specified user.
- *spid* is the number of a specific process. Enclose it in quotes, because a character type argument is expected.

For each process being run, **sp_who** reports the server process ID, its status, the login name of the process user, the name of the host computer, the server process ID of a process that's blocking this one (if any), the name of the database, and the command being run.

If you do not give a login name or *spid*, **sp_who** reports on processes being run by all users.

The following example shows the results of executing **sp_who** without a parameter:

```
spid   status   loginame hostname blk dbname cmd
------ -------- -------- -------- --- ------ ----------------
     1 running  sa       sunbird  0   pubs2  SELECT
     2 sleeping NULL              0   master NETWORK HANDLER
     3 sleeping NULL              0   master MIRROR HANDLER
     4 sleeping NULL              0   master AUDIT PROCESS
     5 sleeping NULL              0   master CHECKPOINT SLEEP

(5 rows affected, return status = 0)
```

For all system processes, **sp_who** reports NULL for the *loginame*.

Getting Information About Login Accounts

Use **sp_displaylogin** to display information about a specified login account. The syntax is:

sp_displaylogin [*loginame*]

where *loginame* is the user login account about which you want information. If you want to display information about your own login, you do not need to specify *loginame*.

When you use **sp_displaylogin** to get information about your own account, you do not need to use the *loginame* parameter. **sp_displaylogin** displays your server user ID, login name, full name, any roles that have been granted to you, date of last password change, default database, default language, and whether your account is locked.

If you are a System Security Officer or System Administrator, you can use the *loginame* parameter to access information about any account.

sp_displaylogin displays all roles that have been granted to you, so even if you have made a role inactive with the **set** command, that role is displayed.

Getting Information About Database Users

The system procedure **sp_helpuser** reports information about authorized users of the current database. Its syntax is:

sp_helpuser [*name_in_db*]

where *name_in_db* is the user's name in the current database. If you give a user's name, **sp_helpuser** reports information about that user. If you do not give a name, it reports information about all users.

The procedure reports the user's name in the database, the user ID, the group name, and the user's login name. The following example shows the results of executing **sp_helpuser** without a parameter in the database *pubs2:*

```
sp_helpuser

Users_name  ID_in_db  Group_name Login_name
----------  --------  ---------- ----------
dbo         1         public     sa
marcy       4         public     marcy
sandy       3         public     sandy
judy        5         public     judy
linda       6         public     linda
anne        2         public     anne
jim         7         senioreng  jim

(7 rows affected)
```

Finding User Names and IDs

To find a user's server user ID or login name, use the system functions **suser_id** and **suser_name**.

Table 3-7: System functions suser_id and suser_name

To Find	Use	With the Argument
Server user ID	**suser_id**	**(["server_user_name"])**
Server user name (login name)	**suser_name**	**([server_user_ID])**

The arguments for these system functions are optional. If you do not provide one, Adaptive Server displays information about the current user.

The following example shows how to find the server user ID for the user "sandy":

```
select suser_id("sandy")

------
     3
```

The following example shows how a System Administrator whose login name is "mary" issues the commands without arguments:

```
select suser_name(), suser_id()
------------------------------ ------
mary                                4
```

To find a user's ID number or name inside a database, use the system functions **user_id** and **user_name**.

Table 3-8: System functions user_id and user_name

To Find	Use	With the Argument
User ID	**user_id**	**([“db_user_name”])**
User name	**user_name**	**([db_user_ID])**

The arguments for these system functions are optional. If you do not provide one, Adaptive Server displays information about the current user. For example:

```
select user_name(10)
select user_name( )
select user_id("joe")
```

Getting Information About Usage: Chargeback Accounting

When a user logs into Adaptive Server, the server begins accumulating CPU and I/O usage for that user. Using this information, Adaptive Server can report total usage for an individual or for all users. Information for each user is kept in the *syslogins* system table in the *master* database.

A System Administrator can configure the frequency with which usage statistics are updated by setting the configuration parameters **cpu accounting flush interval** and **i/o accounting flush interval**. When the user logs out of an Adaptive Server session or accumulates more CPU or I/O usage than the configured **cpu accounting flush interval** or **i/o accounting flush interval** value, the new totals are written to *master..syslogins*.

Reporting Current Usage Statistics

The System Administrator can use the system procedure **sp_reportstats** or **sp_clearstats** to get or clear current total usage data for individuals or for all users on Adaptive Server. To get or clear one user's totals, use the Adaptive Server login as a parameter; to get or clear the total for all users, use **sp_reportstats** or **sp_clearstats** with no parameters.

Displaying Current Accounting Totals

sp_reportstats displays a report of current accounting totals for Adaptive Server users. It reports total CPU and total I/O, as well as the percentage of those resources used. It does not record statistics for the "sa" login (processes with an *suid* of 1), checkpoint, network, and mirror handlers.

Initiating a New Accounting Interval

Adaptive Server continues to accumulate CPU and I/O statistics until you clear the totals from *syslogins* by running **sp_clearstats**. **sp_clearstats** initiates a new accounting interval for Adaptive Server users. It executes **sp_reportstats** to print out statistics for the previous period and clears *syslogins* of the accumulated CPU and I/O statistics.

Choose the length of your accounting interval by deciding how you want to use the statistics at your site. For example, to do monthly cross-department charging for the percentage of Adaptive Server CPU and I/O usage, the System Administrator would run **sp_clearstats** once a month.

For detailed information about these stored procedures, see **sp_reportstats** and **sp_clearstats** in the *Adaptive Server Reference Manual*.

Specifying the Interval for Adding Accounting Statistics

A System Administrator can use configuration parameters to decide how often accounting statistics are added to *syslogins*.

To specify how many machine clock ticks are accumulated before accounting statistics are added to *syslogins*, use the **cpu accounting flush interval** configuration parameter. The default value is 200. For example:

```
sp_configure "cpu accounting flush interval", 600
```

To find out how many microseconds a tick is on your system, run the following query in Adaptive Server:

```
select @@timeticks
```

To specify how many read or write I/Os are accumulated before the information is added (flushed) to *syslogins*, use the **i/o accounting flush interval** configuration parameter. The default value is 1000. For example:

```
sp_configure "i/o accounting flush interval", 2000
```

I/O and CPU statistics for a user are flushed when the user accumulates more I/O or CPU usage than the specified value. The information is also flushed when the user exits an Adaptive Server session.

The minimum value allowed for either configuration parameter is 1. The maximum value allowed is 2,147,483,647.

4 Administering Roles

This chapter describes methods of administering system roles as well as methods for creating and administering user-defined roles. Topics include:

- Overview 4-1
- Tasks and Permissions for System Roles 4-3
- Preparing for User-Defined Roles 4-9
- Creating and Dropping User-Defined Roles 4-10
- Adding and Removing Passwords from a Role 4-11
- Defining and Changing Mutual Exclusivity of Roles 4-11
- Defining and Changing a Role Hierarchy 4-12
- Setting Up Default Activation at Login 4-15
- Granting and Revoking Roles 4-16
- Granting Access to Database Objects 4-18
- Activating and Deactivating Roles 4-18
- Displaying Information About Roles 4-19
- Role Information in the System Tables 4-23

Overview

The roles supported by Adaptive Server enable you to enforce and maintain individual accountability. Adaptive Server provides **system roles**, such as System Administrator and System Security Officer, and **user-defined roles**, which are created by a System Security Officer. A System Security Officer can also define:

- Role hierarchies – roles that contain other roles. The roles can be either user-defined or part of the system.
- Mutual exclusivity of roles – roles that are independent of each other.

More than one login account in an Adaptive Server can be granted any role, and one account can possess more than one role.

System Roles

Table 4-1 lists the system roles, the value to use for the *role_granted* option of the **grant role** or **revoke role** command, the related login, and the tasks usually performed by a person with that role.

Table 4-1: System roles and related tasks

Role	Value for *role_granted*	Description
System Administrator	**sa_role**	Manages and maintains Adaptive Server databases and disk storage
System Security Officer	**sso_role**	Performs security-related tasks
Operator	**oper_role**	Backs up and loads databases server-wide

➤ ***Note***

The **sybase_ts_role**, **replication_role**, and **navigation_role** roles are not included in the evaluated configuration.

User-Defined Roles

Adaptive Server enables a System Security Officer to create roles and grant them to users, groups of users, or other roles. For example, a System Security Officer might create the role “financial_analyst” and grant that role to users that perform financial analyst tasks. Similarly, the role “salary_administrator” might be created and assigned to users who perform payroll tasks. Object owners can grant database access as appropriate to each role.

Role Hierarchies and Mutual Exclusivity

A System Security Officer can define role hierarchies such that if a user has one role, the user also has roles lower in the hierarchy. For example, the “chief_financial_officer” role might contain both the "financial_analyst" and the "salary_administrator" roles, as shown in Figure 4-1.

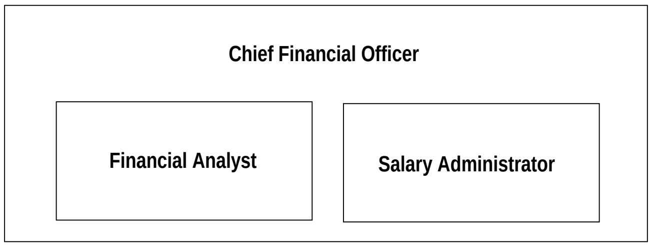

Figure 4-1: Role hierarchy

The Chief Financial Analyst can perform all tasks and see all data that can be viewed by the Salary Administrators and Financial Analysts.

Two roles can be defined to be mutually exclusive for:

- Membership – one user cannot be granted both roles. For example, an installation might not want the "payment_requestor" and "payment_approver" roles to be granted to the same user.
- Activation – one user cannot activate, or enable, both roles. For example, a user might be granted both the "senior_auditor" and the "equipment_buyer" roles, but the installation may not want to permit the user to have both roles enabled at the same time.

System roles, as well as user-defined roles, can be defined to be in a role hierarchy or to be mutually exclusive. For example, you might want a "super_user" role to contain the System Administrator, Operator, and "tech_support" roles. In addition, you might want to define the System Administrator and System Security Officer roles to be mutually exclusive for membership; that is, one user cannot be granted both roles.

Tasks and Permissions for System Roles

Adaptive Server users can be granted special operational and administrative roles, called **system roles**. System roles, which are pre-defined by Adaptive Server, provide individual accountability for users performing system administration and security-related tasks. Roles are granted to individual server login accounts, and actions performed by these users can be audited and attributed to them. These roles are:

- System Administrator
- System Security Officer
- Operator

In addition, two kinds of object owners have special status because of the objects they own. These ownership types are:

- Database Owner
- Database object owner

All of these are discussed in the following sections.

System and Security Administration Roles

The System Administrator, System Security Officer, and Operator roles are essential for managing Adaptive Server. These roles are granted to individual users with the **grant role** command. See “Granting and Revoking Roles” on page 4-16 for more information.

System Administrator

A **System Administrator** performs administrative tasks that are not related to specific applications. The System Administrator is not necessarily one individual; the role can be granted to any number of individual login accounts. In a large organization, the System Administrator’s role may be carried out by several people or groups. It is important, however, that the System Administrator’s functions be centralized or very well coordinated.

System Administrator tasks include:

- Installing Adaptive Server
- Managing disk storage
- Granting permissions to Adaptive Server users
- Transfering bulk data between Adaptive Server and other software programs
- Modifying, dropping, and locking server login accounts
- Monitoring Adaptive Server’s automatic recovery procedure
- Diagnosing system problems and reporting them, as appropriate

- Fine-tuning Adaptive Server by changing the configurable system parameters
- Creating and granting ownership of user databases
- Granting and revoking the System Administrator role
- Setting up groups (which is convenient for granting and revoking permissions)

A System Administrator takes on the identity of Database Owner in any database he or she enters, including *master*, by assuming the user ID 1. A System Administrator operates outside the discretionary access control (DAC) protection system; that is when a System Administrator accesses objects Adaptive Server does not check the DAC permissions.

There are several commands and system procedures that can be issued only by a System Administrator. Permissions on these commands cannot be transferred to other users. For details about System Administrator permissions, see Chapter 5, "Managing User Permissions."

System Security Officer

The System Security Officer is responsible for security-sensitive tasks in Adaptive Server, such as:

- Creating server login accounts
- Granting and revoking the System Security Officer and Operator roles
- Granting permission to use the **set proxy** or **set session authorization** commands
- Creating user-defined roles
- Granting and revoking user-defined roles
- Setting security-related configuration parameters, such as **max roles enabled per user**
- Changing the password of any account
- Setting the password expiration interval
- Managing the audit system

The System Security Officer can **access** any database, but usually has no special permissions on database objects. An exception is the *sybsecurity* database, where only a System Security Officer can access the audit trail tables. There are several commands and system procedures that can be issued only by a System Security Officer. Permissions on these commands cannot be transferred to other users.

Operator

An Operator is a user who can back up and load databases server-wide. The Operator role allows a single user to use the **dump database, dump transaction**, **load database**, and **load transaction** commands to back up and restore all databases on a server without having to be the owner of each database. These operations can be performed in a single database by the Database Owner or a System Administrator.

Data Ownership Roles

Adaptive Server recognizes two types of owners:

- Database Owner
- Database object owner

Database Owner

The **Database Owner** is the creator of a database or someone to whom database ownership has been transferred. A System Administrator grants users the authority to create databases with the **grant** command.

A Database Owner logs into Adaptive Server using his or her assigned login name and password. In other databases, that owner is known by his or her regular user name. In his or her own database Adaptive Server recognizes the user as having the "dbo" account.

A Database Owner can:

- Run the system procedure **sp_adduser** to allow other Adaptive Server users access to the database
- Use the **grant** command to give other users permission to create objects and execute commands within the database

Adding users to databases is discussed in Chapter 3, "Managing Adaptive Server Logins and Database Users." Granting permissions to users is discussed in Chapter 5, "Managing User Permissions."

The Database Owner does not automatically receive permissions on objects owned by other users. However, a Database Owner can impersonate other users in the database at any time by using the

setuser command, and temporarily assume their permissions. Using a combination of the **setuser** and **grant** commands, the Database Owner can acquire permissions on any object in the database.

➤ ***Note***

Because the Database Owner role is so powerful, the System Administrator should plan carefully who should own databases in the server. The System Security Officer should consider auditing the database activity of all Database Owners.

Database Object Owner

A **Database object owner** is a user who creates a database object. **Database objects** are tables, indexes, views, defaults, triggers, rules, constraints, and procedures. Before a user can create a database object, the Database Owner must grant the user permission to create objects of a particular type. There is no special login name or password for a database object owner.

The database object owner creates an object using the appropriate **create** statement, and then grants permission to other users.

The creator of a database object is automatically granted all permissions on that object. The System Administrator also has all permissions on the object. The owner of an object must explicitly grant permissions to other users before they can access the object. Even the Database Owner cannot use an object directly unless the object owner grants him or her the appropriate permission. However, the Database Owner can always use the **setuser** command to impersonate any other user in the database, including the object owner.

➤ ***Note***

When a database object is owned by someone other than the Database Owner, the user (including a System Administrator) must qualify the name of that object with the object owner's name—*ownername.objectname*—to access the object. If an object or a procedure needs to be accessed by a large number of users, particularly in ad hoc queries, having these objects owned by "dbo" greatly simplifies access.

Managing System Roles in Adaptive Server

When Adaptive Server is installed, it includes the “sa” account, which is automatically granted to the System Administrator, System Security Officer, and Operator roles during installation. The “sa” user is the Database Owner of the system databases. There are two options for managing the System Administrator role:

- Creating multiple logins to distribute the responsibility and increase accountability for performing system administration and security-related functions. To use this option, complete the following steps:
  1. Log into Adaptive Server using the “sa” account.
  2. Create new server logins for the users who are to be granted the System Administrator and System Security Officer roles.
  3. Grant the correct roles to those users by using **grant role**.
  4. Verify that the individual logins function correctly.
  5. Lock the “sa” account.

➤ ***Note***

When you lock the “sa” account, be sure to check all scripts that may contain the “sa” login name and password. These can include scripts that perform backups, run **bcp**, or perform **dbcc** checking. The scripts cannot run if they are meant to run as “sa” and the “sa” account is locked. Change the logins in those scripts to the name of a user with the correct role.

- Using the “sa” account to perform all system administration and security-related functions. Any user who knows the “sa” password can use the account, and any actions performed can only be traced to “sa”. It is not possible to determine the individual user who was logged in as “sa”.

➤ ***Note***

Sybase strongly recommends against using this option. If a single account is used to perform all security and system administration functions, and that account becomes compromised, the potential for damage is vastly greater than if the security and administrative functions were distributed across several accounts.

For more information about assigning system roles to individual users, see Chapter 2, “Getting Started After Installation.”

Preparing for User-Defined Roles

A System Security Officer can define user-defined roles for security purposes. The System Security Officer grants each user-defined role to users according to the functions they perform. This simplifies the task of granting and preventing access to data.

For example, instead of having object owners grant privileges on each object individually to each employee, the System Security Officer can create user-defined roles, request object owners to grant privileges to each, and grant these roles to individual employees based on the functions they perform in the organization. The System Security Officer can also revoke user-defined roles granted to the employee.

Before creating user-defined roles, plan what roles you want to create and determine the maximum number of roles you plan to have for your installation.

Planning User-Defined Roles

Before you implement user-defined roles, decide:

- The roles you want to create
- The responsibilities for each role
- The position of each in the role hierarchy
- Which roles in the hierarchy will be mutually exclusive
- Whether such exclusivity will be at the membership level or activation level.

User-defined role names can duplicate user names; therefore, you can avoid conflict by planning a naming convention. For example, you could use the “_role” suffix for role names, although the system does not check for such restrictions.

If a role has the same name as a user, you can avoid conflict by creating a new role, having it contain the original role, and then granting the new role to the user.

If role names are identical to user names, Adaptive Server grants the role to the user.

Configuring User-Defined Roles

After you have planned the roles to create and the relationships among them, configure your system for user-defined roles with the **max roles enabled per user** configuration parameter.

The maximum number of roles that a user can activate per user session is 127. The default value is 20. The minimum number of roles, which is 10, includes the system roles that come with Adaptive Server.

The maximum number of roles that can be activated server-wide is 992. The first 32 roles are reserved for Sybase system roles.

Creating and Dropping User-Defined Roles

Only a System Security Officer can create or drop a user-defined role.

Creating a Role

Use the **create role** command to create a role. The syntax is:

```
create role role_name [with passwd "password"]
```

where:

- *role_name* is the name of a new role.
- *password* is an optional password that must be specified by the user who will use the role.

For example, to create the **intern_role** without a password, enter:

```
create role intern_role
```

To create the **doctor_role** and assign the password “physician”, enter:

```
create role doctor_role with passwd "physician"
```

Dropping a User-defined Role

Use the **drop role** command to drop a role. The syntax is:

```
drop role role_name [with override]
```

where:

- *role_name* is the name of a user-defined role.
- **with override** revokes all access privileges granted to the role in every database server-wide.

If the role has any access privileges already granted, you must revoke all privileges granted to the role in all databases before dropping the role. If you do not, the command fails. Use either of the following methods to revoke privileges:

- Use the **revoke** command.
- Use the **with override** option with the **drop role** command.

  Dropping a role **with override** means the object owners of each affected database need not individually revoke a role's privileges. The **with override** option ensures that Adaptive Server automatically removes permission information for the role from all databases.

You need not drop memberships before dropping a role. Dropping a role automatically removes any user's membership in that role, regardless of whether you use the **with override** option.

For example, to drop the **intern_role** user-defined role and remove its access privileges from all databases, enter:

```
drop role intern_role with override
```

Adding and Removing Passwords from a Role

Only a System Security Officer can add or drop a password from a role. If a role has a password, the user must specify the password when activating the role with **set role**. For more information, see "Activating and Deactivating Roles" on page 4-18.

Use the **alter role** command to add or drop a password from either a system or user-defined role. The syntax is:

```
alter role role_name [add passwd password |
   drop passwd]
```

For example, to require the password "oper8x" for the **oper_role**, enter:

```
alter role oper_role add passwd oper8x
```

To drop the password from the role, enter:

```
alter role oper_role drop passwd
```

Defining and Changing Mutual Exclusivity of Roles

Use the following syntax to define mutual exclusivity between two roles:

```
alter role role1 { add | drop } exclusive {
   membership | activation } role2
```

For example, to define **intern_role** and **specialist_role** as mutually exclusive at the membership level, enter:

```
alter role intern_role add exclusive membership
   specialist_role
```

To define **sso_role** and **sa_role** as mutually exclusive at the activation level, enter:

```
alter role sso_role add exclusive activation
   sso_role
```

Defining and Changing a Role Hierarchy

Defining a role hierarchy involves choosing the type of hierarchy to create and the roles in that hierarchy, and then implementing the hierarchy by granting roles to other roles.

For example:

```
grant role intern_role to specialist_role
grant role doctor_role to specialist_role
```

When you grant one role to another, you create a role hierarchy, as shown in Figure 4-2:

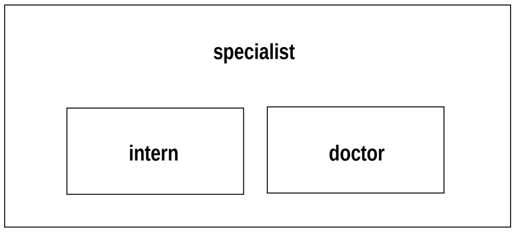

Figure 4-2: Creating a role hierarchy

In Figure 4-2, the “specialist” role contains the “doctor” and “intern” roles. This means that the “specialist” role has all the privileges of both the “doctor” and “intern”roles.

In a similar fashion, if you want to establish a hierarchy with a “super_user” role containing the **sa_role** and **oper_role** system roles, specify:

```
grant role sa_role to super_user
grant role oper_role to super_user
```

When creating role hierarchies:

- You cannot grant a role to another role that directly contains it. If you try, you get a warning to that effect. This prevents duplication.

  For example, in Figure 4-2, you cannot grant the “doctor” role to the “specialist” role because “specialist” already contains “doctor”.

- You can grant a role to another role that does not directly contain it.

  For example, in Figure 4-3, you can grant the “intern” role to the “specialist” role, even though “specialist” already contains the “doctor” role, which contains “intern”.

  If you subsequently drop the “doctor” role from the “specialist” role, then “specialist” still contains the “intern”role.

  In Figure 4-3, the “doctor” role has “consultant” role permissions because the “consultant” role has been granted to the “doctor” role. The “specialist” role also has “consultant” role permissions because “specialist” contains the “doctor” role, which in turn contains the “consultant” role.

  However, the “intern” role does not have “consultant” role privileges, because “intern” does not contain the “consultant” role, either directly or indirectly.

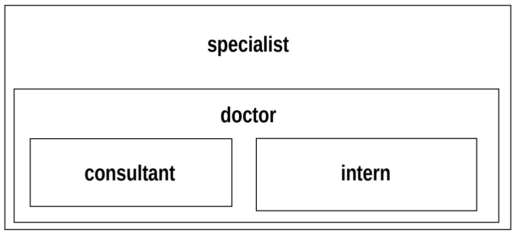

Figure 4-3: Explicitly and implicitly granted privileges

- You cannot grant a role to another role that is contained by the first role. This prevents “loops” within the hierarchy.

For example, in Figure 4-4, you cannot grant the “specialist” role to the “consultant” role. The grant fails because “consultant” is already contained in “specialist”.

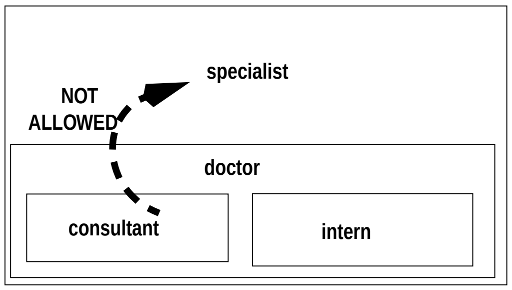

Figure 4-4: Granting a role to a role contained by grantor

- When the System Security Officer grants a user a role that contains other roles, the user implicitly gets membership in all roles contained by the granted role. However, a role can only be activated or deactivated directly if the user has explicit membership in that role.
- The System Security Officer cannot grant one role to another role that is explicitly or implicitly mutually exclusive at the membership level with the first role.

  For example, in Figure 4-5, if the “intern” role is defined as mutually exclusive at the membership level with the “consultant” role, the System Security Officer cannot grant the “intern” role to the “doctor” role.

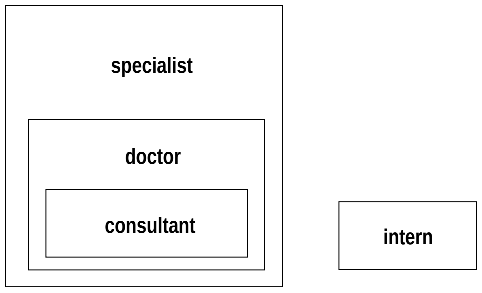

Figure 4-5: Mutual exclusivity at membership

- The user can activate or deactivate only directly granted roles.

  For example, in the hierarchy shown in Figure 4-5, assume that you have been granted the “specialist” role. You have all the permissions of the “specialist” role, and, implicitly, because of the hierarchy, you have all the permissions of the “doctor” and “consultant” roles. However, you can activate only the “specialist” role. You cannot activate the “doctor” or “consultant” roles because they were not directly granted to you. For information about using **set role** to activate or deactivate a role, see “Activating and Deactivating Roles” on page 4-18.

Revoking roles from other roles is similar to granting roles to other roles. It removes a containment relationship, and the containment relationship must be a direct one, as shown in Figure 4-6:

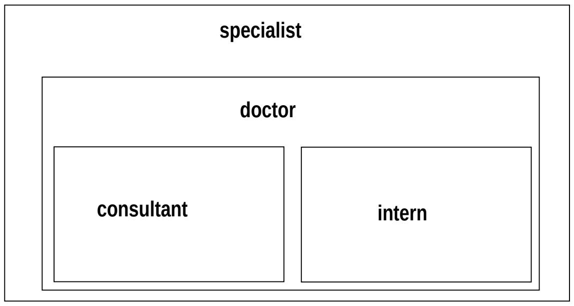

Figure 4-6: Effect of revoking roles on role hierarchy

For example, the following points apply to the role hierarchy displayed in Figure 4-6:

- If the System Security Officer revokes the “doctor” role from the “specialist” role, “specialist” no longer contains the “consultant” role or the “intern” role.
- The System Security Officer cannot revoke the “intern” role from the “specialist” role because “intern” is not directly contained by “specialist”.

Setting Up Default Activation at Login

A System Security officer can change a role’s default setting for any user. Individual users can change only their own default settings.

When a user logs into Adaptive Server, the user's roles are not necessarily active, depending upon the default that is set for the role. If a role has a password associated with it, the role is never active when the user logs on. The user must use the **set role** command to activate the role.

The System Security Officer or user determines whether to activate any roles granted by default at login. The system procedure **sp_modifylogin** sets the default status of user roles individually for each user.

By default, user-defined roles are not activated at login, but system roles are automatically activated, if they do not have passwords associated with them.

To set up a role to activate at login, use the following syntax:

```
sp_modifylogin loginname, "add default role",
   role_name
```

To ensure that a role is inactive at login, use the following syntax:

```
sp_modifylogin loginname, "drop default role",
   role_name
```

For example, to change the default setting for Ralph's **intern_role** to be active automatically at login, execute:

```
sp_modifylogin ralph, "add default role", intern_role
```

To cause the **gastroenterologist_role** to be inactive at login for "csmith", execute:

```
sp_modifylogin csmith, "drop default role",
   gastroenterologist_role
```

Granting and Revoking Roles

After a role is defined in the system, it can be granted to any login account or any role in the server, provided that it does not violate the rules of mutual exclusivity and hierarchy. Table 4-2 lists the tasks related to roles, the role required to perform the task, and the command or system procedure to use.

Table 4-2: Tasks, required roles, and commands to use

Task	Required Role	Command
Grant the **sa_role** role	System Administrator	**grant role**
Grant the **sso_role** role	System Security Officer	**grant role**

Table 4-2: Tasks, required roles, and commands to use

Task	Required Role	Command
Grant the **oper_role** role	System Security Officer	**grant role**
Grant user-defined roles	System Security Officer	**grant role**
Create role hierarchies	System Security Officer	**grant role**
Modify role hierarchies	System Security Officer	**revoke role**
Revoke system roles	System Security Officer	**revoke role**
Revoke user-defined roles	System Security Officer	**revoke role**

Granting Roles

To grant roles to users or other roles, use the following syntax:

```
grant role role_granted [{, role_granted}...]
   to grantee [{, grantee}...]
```

where:

- *role_granted* is the name of the role being granted. You can specify any number of roles to be granted.
- *grantee* is the name of the user or role. You can specify any number of grantees.

All roles listed in the **grant** statement are granted to all grantees. If you grant one role to another, it creates a role hierarchy.

For example, to grant Susan, Mary, and John the "financial_analyst" and the "payroll_specialist" roles, enter:

```
grant role financial_analyst, payroll_specialist
  to susan, mary, john
```

Revoking Roles

You can use **revoke role** to revoke roles from users and other roles. To revoke a role, use the following syntax:

```
revoke role role_name [{, role_name}...]from grantee
   [{, grantee}...]
```

where:

- *role_name* is the name of the role being revoked. You can specify any number of roles to be revoked.

- *grantee* is the name of the user or role. You can specify any number of grantees.

All roles listed in the **revoke** statement are revoked from all grantees.

You cannot revoke a role from a user while the user is logged in.

Granting Access to Database Objects

Use the **grant** command to grant access privileges to a user, a group, or a role. Be sure that user names and role names in the database where the grant is made do not conflict. If there is a conflict, the grant is made to the user.

Only grantable privileges, such as **select, update,** or **delete** from a table, or **create** a table or a database, can be granted. Privileges requiring **sa_role** or **sso_role** permissions are not grantable. For a complete list of grantable privileges, see Chapter 5, "Managing User Permissions."

Activating and Deactivating Roles

Roles must be active to have the access privileges of each role. Depending on the default set for a role, the role may or may not be active at login. If the role has a password, it will always be inactive at login.

To activate or deactivate a role, use the following syntax:

set role *role_name* [on|off]

To activate or deactivate a role that has an attached password, use the following syntax:

set role *role_name* with passwd "*password*" [on|off]

For example, to activate the "financial_analyst" role with the password "sailing19", enter:

```
set role financial_analyst with passwd "sailing19"
```

It is a good idea to activate roles only when you need them and to turn them off when you no longer need them. For example, when the **sa_role** is active, you assume the identity of Database Owner within any database that you use. If you want to turn off the System Administrator role and assume your "real" user identity, use this command:

```
set role sa_role off
```

If you are granted a role during a session, and you want to activate it immediately, use **set role** to turn it on.

Displaying Information About Roles

Table 4-3 lists the system procedures and functions to use to find information about roles and the section in this chapter that provides details.

Table 4-3: Finding information about roles

To Display Information About	Use	See
The role ID of a role name	**role_id** system function	"Finding a Role ID" on page 4-19
The role name of a role ID	**role_name** system function	"Finding a Role Name" on page 4-20
System roles	**show_role** system function	"Viewing Information About System Roles" on page 4-20
Role hierarchies and roles that have been granted to a user or users	**sp_displayroles** system procedure	"Displaying a Role Hierarchy" on page 4-20
Whether one role contains another role in a role hierarchy	**role_contain** system function	"Viewing User Roles in a Hierarchy" on page 4-21
Whether two roles are mutually exclusive	**mut_excl_roles** system function	"Determining Mutual Exclusivity" on page 4-21
Roles that are active for the current session	**sp_activeroles** system procedure	"Determining Role Activation" on page 4-21
Whether you have activated the correct role to execute a procedure	**proc_role** system function	"Checking for Roles in Stored Procedures" on page 4-21
Logins, including roles that have been granted	**sp_displaylogin** system procedure	"Displaying Login Account Information" on page 4-22
Permissions for a user, group, or role	**sp_helprotect** system procedure	"Checking Permissions" on page 4-23

Finding a Role ID

To find out the **role_id** of a role when you know the role name, use the **role_id** system function. The syntax is:

```
role_id(role_name)
```

Any user can execute **role_id**. If the role is a valid one, **role_id** returns the server-wide ID of the role (*srid*). The *syssrvroles* system table contains an *srid* column with the role ID and a *name* column with the role name. If the role is not valid, **role_id** returns NULL.

Finding a Role Name

To find out the **role_name** of a role when you know the role ID, use the **role_name** system function. The syntax is:

```
role_name(role_id)
```

Any user can execute **role_name**.

Viewing Information About System Roles

The **show_role** system function shows the currently active **system roles** for the specified login. The syntax is:

```
show_role()
```

If you have not activated any system role, **show_role** returns NULL. If you are a Database Owner, and you execute **show_role** after using **setuser** to impersonate another user, **show_role** returns your own active system roles, not the active system roles of the user you are impersonating.

Any user can execute **show_role**.

➤ ***Note***

The **show_role** function does not give information about user-defined roles.

Displaying a Role Hierarchy

You can see all roles granted to your login name or see the entire hierarchy tree of roles displayed in table format using **sp_displayroles**. The syntax is:

```
sp_displayroles {login_name | rolename [, expand_up |
   expand_down]}
```

Any user can execute **sp_displayroles** to see his or her own roles. Only the System Security Officer or the System Administrator can view information about roles granted to other users.

Viewing User Roles in a Hierarchy

The **role_contain** built-in system function shows whether any one role you specify contains any other role you specify. The syntax is:

```
role_contain (["role1", "role2"])
```

If *role1* contains *role2*, **role_contain** returns 1.

Any user can execute **role_contain**.

Determining Mutual Exclusivity

You can use the **mut_excl_roles** function to determine whether any two roles assigned to you are mutually exclusive and the level at which they are mutually exclusive. The syntax is:

```
mut_excl (role1, role2) [membership | activation]
```

Any user can execute **mut_excl_roles**. If the specified roles, or any role contained by either specified role, are mutually exclusive, **mut_excl_roles** returns 1; if the roles are not mutually exclusive, **mut_excl_roles** returns 0.

Determining Role Activation

To find all active roles for the current login session of Adaptive Server, use **sp_activeroles**. The syntax is:

```
sp_activeroles [expand_down]
```

If you specify **expand_down**, **sp_activeroles** displays the hierarchy of all roles contained by any roles granted to you.

Any user can execute **sp_activeroles**.

Checking for Roles in Stored Procedures

Use **proc_role** within a stored procedure to guarantee that only users with a specific role can execute the procedure. Only **proc_role** provides a fail-safe way to prevent inappropriate access to a particular stored procedure.

You can use the **grant execute** command to grant execute permission on a stored procedure to all users who have been granted a specified role. Similarly, **revoke execute** removes this permission.

However, **grant execute** permission does not prevent users who do not have the specified role from being granted execute permission on a stored procedure. If you want to ensure, for example, that all users who are not System Administrators can never be granted permission to execute a stored procedure, you can use the **proc_role** system function within the stored procedure itself. It checks to see whether the invoking user has the correct role to execute the procedure.

proc_role takes a string for the required role and returns 1 if the invoker possesses it. Otherwise, it returns 0.

For example, here is a procedure that uses **proc_role** to see if the user has the **sa_role** role:

```
create proc test_proc
as
if (proc_role("sa_role") = 0)
begin
    print "You don’t have the right role"
    return -1
end
else
    print "You have System Administrator role"
    return 0
```

Displaying Login Account Information

You can use the **sp_displaylogin** system procedure to display information about a login account, including any roles granted to that account. The syntax is:

sp_displaylogin [*loginame*]

If you are not a System Security Officer or System Administrator, you can get information only about your own account, and you do not need to use the *loginame* parameter. **sp_displaylogin** displays your server user ID, login name, full name, roles, date of last password change, and whether your account is locked.

If you are a System Security Officer or System Administrator, you can use the *loginame* parameter to access information about any login.

➤ ***Note***

You cannot use the **sp_helpuser** system procedure to display role information. **sp_helpuser** displays user, alias, or group information only.

Checking Permissions

To check on the permissions granted to users, groups, or roles, or permissions granted on database objects, use the **sp_helprotect** system procedure. The syntax is:

```
sp_helprotect [name [,username [,"grant"
   [,"none"|"granted"|"enabled"|role_name]]]]
```

Any user can use **sp_helprotect** to view his or her own permissions. Depending on the parameter you specify, **sp_helprotect** displays all permissions granted a user or role, permissions on any specified database object, permissions granted to active roles, and whether the permissions can be granted to others. The display always includes information on permissions granted to the group of which the specified user is a member.

If you do not specify any parameters, **sp_helprotect** returns permission information on all your active roles.

Only the System Security Officer or System Administrator can use **sp_helprotect** to display the permissions granted to other users.

Role Information in the System Tables

The *syssrvroles, sysloginroles, sysprotects* and *sysroles* system tables contain information about both system and user-defined roles.

The *syssrvroles* table contains a row for each role. When you install Adaptive Server, *syssrvroles* contains a row for **sa_role**, **sso_role**, and **oper_role**. When a System Security Officer creates a user-defined role with **create role**, a row is added to *syssrvroles* for the new role.

Whenever a user is granted a role, another row is added to *sysloginroles*. When you install Adaptive Server, *sysloginroles* contains rows only for the “sa” account, with a server user ID (*suid*) of 1, showing that the “sa” account has been granted the **sa_role**, **sso_role**, **oper_role**, and **replication_role**.

The *sysprotects* table contains entries for every permission granted to the role, as well as permissions granted to individual users and to groups. When a database permission is granted to a role, entries indicating the permission are added to the *sysprotects* table in the database. If an entry is not already in *syssrvroles* for the role, Adaptive Server adds an entry to *sysroles* to map the local role ID (*lrid*) to the server-wide role ID (*srid*), which is also stored in *syssrvroles*.

5 Managing User Permissions

This chapter describes the use and implementation of user permissions. Topics include:

- Overview 5-1
- Types of Users and Their Privileges 5-2
- Granting and Revoking Permissions on Database Objects 5-10
- Granting and Revoking Proxy Authorization and Roles 5-19
- Reporting on Permissions 5-20
- Using Views and Stored Procedures As Security Mechanisms 5-23

Overview

Discretionary access controls (DAC) allow you to restrict access to objects and commands based on a user's identity or group membership. The controls are "discretionary" because a user with a certain access permission, such as an object owner, can choose to pass that access permission on to other users.

System Administrators operate outside the DAC system and have access permissions on all database objects at all times. The exception is that only a System Security Officer can access the audit trail tables in the *sybsecurity* database.

Database Owners do not automatically receive permissions on objects owned by other users; however, they can:

- Temporarily acquire all permissions of a user in the database by using the **setuser** command to assume the identity of that user.
- Permanently acquire permission on a specific object by using the **setuser** command to assume the identity of the object owner, and then using **grant** commands to grant the permissions.

For more information on users and privileges, see "Types of Users and Their Privileges" on page 5-2. For details on assuming another user's identity to acquire that user's permissions on a database or object, see "Acquiring the Permissions of Another User" on page 5-7.

You can give various permissions to users, groups, and roles with the **grant** command, and rescind them with the **revoke** command. Use these commands to give users permission to create databases, to create objects within a database, execute certain commands such as **set proxy**, and to access specified tables, views, and columns. For permissions that default to "public", no **grant** or **revoke** statements are needed.

Some commands can be used at any time by any user, with no permission required. Others can be used only by users of a particular status and they are not transferable.

The ability to assign permissions for the commands that can be granted and revoked is determined by each user's role or status and by whether the user was granted a role with permission that includes the option to grant that permission to other users.

To supplement and complement the **grant** and **revoke** commands, you can use views and stored procedures as security mechanisms. For details, see "Using Views and Stored Procedures As Security Mechanisms" on page 5-23.

Types of Users and Their Privileges

Adaptive Server's discretionary access control system recognizes the following types of users:

- System Administrators
- System Security Officers
- Operators
- Database Owners
- Database object owners
- Other users (also known as "public")

System Administrator Privileges

System Administrators are special users who:

- Handle tasks that are not specific to applications
- Work outside Adaptive Server's discretionary access control system

The role of System Administrator is usually granted to individual Adaptive Server logins. This provides a high degree of individual accountability because all actions taken by that user can be traced to his or her individual server user ID. If the server administration tasks at your site are few enough that they are performed by a single individual, you may instead choose to use the “sa” account that is installed with Adaptive Server. At installation, the “sa” account user has permission to assume the System Administrator, System Security Officer, and Operator roles. Any user who knows the “sa” password can log into that account and assume any or all of these roles.

The fact that a System Administrator operates outside the protection system serves as a safety precaution. For example, if the Database Owner accidentally deletes all the entries in the *sysusers* table, the System Administrator can restore the table (provided, of course, that the Database Owner is making regular backups). Several commands can be issued only by a System Administrator. They cannot be granted to any other user. They include **disk init**, **disk refit**, **disk reinit**, **shutdown**, **kill**, and the disk mirroring commands.

In granting permissions, a System Administrator is treated as the object owner. If a System Administrator grants permission on another user’s object, the owner‘s name appears as the grantor in *sysprotects* and in **sp_helprotect** output.

In addition, System Administrators participate in login management in that they are responsible for dropping logins and can lock and unlock logins. System Security Officers share login management responsibilities with System Administrators. System Security Officers are responsible for adding logins and can also lock and unlock logins.

Permissions for Creating Databases

Only a System Administrator can grant permission to use the **create database** command. The user that receives **create database** permission must also be a valid user of the *master* database because all databases are created while using *master*.

In many installations, the System Administrator maintains a monopoly on **create database** permission to centralize control of database placement and database device space allocation. In these situations, a System Administrator creates new databases on behalf of other users, and then transfers ownership to the appropriate user.

To create a database that is to be owned by another user:

1. Issue the **create database** command in the *master* database.
2. Switch to the new database with the **use** command.
3. Execute the **sp_changedbowner** system procedure.

System Security Officer Privileges

System Security Officers perform security-sensitive tasks in Adaptive Server. These tasks include:

- Granting the System Security Officer and Operator roles
- Administering the audit system
- Changing passwords
- Adding new logins
- Locking and unlocking login accounts
- Creating and granting user-defined roles
- Granting permission to use the **set proxy** or **set session authorization** commands

System Administrators share login management responsibilities with System Security Officers. System Administrators are responsible for dropping logins and can also lock and unlock logins.

The System Security Officer can **access** any database but, in general, has no special permissions on database objects. An exception is the *sybsecurity* database, where only a System Security Officer can access the *sysaudits* table. System Security Officers can access any database so they can enable auditing in any database. There are also several system procedures that can be executed only by a System Security Officer. Permission to execute the procedures cannot be transferred to other users.

System Security Officers can repair any damage inadvertently done to the protection system by a user. For example, if the Database Owner forgets his or her password, a System Security Officer can change the password to allow the Database Owner to log in.

System Security Officers can also create and grant user-defined roles to users, other roles, or groups. For information about creating and granting user-defined roles, see Chapter 4, “Administering Roles.”

Operator Privileges

Users who have been granted the Operator role can back up and restore databases on a server-wide basis without having to be the owner of each database. The Operator role allows a user to use the following commands on any database:

dump database
dump transaction

load database
load transaction

Database Owners also have dump and load permissions, but only on the databases they own. System Administrators can dump and load all databases.

Database Owner Privileges

Database Owners and System Administrators are the only users who can grant object creation permissions to other users. The Database Owner has full privileges to do anything inside that database, and must explicitly grant permissions to other users with the **grant** command.

Permission to use the following commands is automatically granted to the Database Owner and cannot be transferred to other users:

checkpoint
dbcc
drop database
dump database
dump transaction
grant (object creation permissions)
load database
load transaction
revoke (object creation permissions)
setuser

Users with the Operator role can use **dump database**, **dump transaction**, **load database**, and **load transaction** on any database.

Database Owners can grant permission to use the following commands to other users:

create default
create procedure
create rule
create table
create view
grant (permissions on system tables)
grant (**select, insert, delete, update, references,** and **execute** permissions on database objects)
revoke (permissions on system tables)
revoke (**select, insert, delete, update, references,** and **execute** permissions on database objects)

Permissions on System Tables

Permissions for use of the system tables can be controlled by the database owner, just like permissions on any other tables. By default, when Adaptive Server is installed, the **installmodel** script grants **select** access to "public" (all users) for most system tables and for most fields in the tables. However, no access is given for some system tables, such as *systhresholds*, and no access is given for certain fields in other system tables. For example, all users, by default, can select all columns of *sysobjects* except *audflags*.

To determine the current permissions for a particular system table, execute:

```
sp_helprotect system_table_name
```

For example, to check the permissions of *systhresholds* in *your_database*, execute:

```
use your_database
go
sp_helprotect systhresholds
go
```

The default situation is that no users—including Database Owners—can modify the system tables directly. Instead, the system procedures supplied with Adaptive Server modify the system tables. This helps guarantee integrity. Adaptive Server does provide a mechanism to permit ad hoc changes to system tables; however, this mechanism must be used with caution.

◆ ***WARNING!***

Updating certain fields in the system tables prevents Adaptive Server from running. Therefore, allowing direct updates to the system tables is not recommended. For information and strict guidelines for allowing direct system table updates, see the *System Administration Guide*.

Permissions on System Procedures

Permissions on system procedures are set in the *sybsystemprocs* database, where the system procedures are stored.

Security-related system procedures can only be run by System Security Officers. Certain other system procedures can only be run by System Administrators.

Some of the system procedures can be run only by Database Owners. These procedures make sure that the user executing the procedure is the owner of the database from which they are being executed.

Other system procedures can be executed by any user who has been granted permission. A user must have permission to execute a system procedure in all databases, or in none of them.

Users who are not listed in *sybsystemprocs..sysusers* are treated as “guest” in *sybsystemprocs*, and are automatically granted permission on many of the system procedures. To deny a user permission on a system procedure, the System Administrator must add him or her to *sybsystemprocs..sysusers* and issue a **revoke** statement that applies to that procedure. The owner of a user database cannot directly control permissions on the system procedures from within his or her own database.

Acquiring the Permissions of Another User

A Database Owner can use the **setuser** command to “impersonate” another user’s identity and permissions status in the current database. A Database Owner may find it convenient to use **setuser** to:

- Access an object owned by another user
- Grant permissions on an object owned by another user
- Create an object that will be owned by another user
- Temporarily assume the DAC permissions of another user for some other reason.

While the **setuser** command enables the Database Owner to automatically acquire another user’s DAC permissions, the command does not affect the roles that have been granted.

setuser permission defaults to the Database Owner and cannot be transferred. The user being impersonated must be an authorized user of the database. When the Database Owner uses the **setuser** command, Adaptive Server checks the permissions of the user being impersonated.

System Administrators can use **setuser** to create objects that will be owned by another user. However, System Administrators operate outside the discretionary access control permissions system; therfore, they need not use **setuser** to acquire another user’s permissions. The **setuser** command remains in effect until another **setuser** command is given, or until the current database is changed with the **use** command, or until the user logs off.

The syntax is:

```
setuser ["user_name"]
```

where *user_name* is the name of the user to be impersonated.The user must be a valid user in the database.

To reestablish your original identity, use the **setuser** command with no value for *user_name*.

The following example shows how the Database Owner would grant Joe permission to read the *authors* table, which is owned by Mary:

```
setuser "mary"

grant select on authors to joe

setuser     /*re-establishes original identity*/
```

Changing Database Ownership

Use the system procedure **sp_changedbowner** to change the ownership of a database. Often, System Administrators create the user databases, then give ownership to another user after some of the initial work is complete. Only the System Administrator can execute **sp_changedbowner**.

It is a good idea to transfer ownership before the user has been added to the database, and before the user has begun creating objects in the database. The new owner must already have a login name on Adaptive Server, but cannot be a user of the database, or have an alias in the database. You may have to use **sp_dropuser** or **sp_dropalias** before you can change a database's ownership, and you may have to drop objects before you can drop the user.

Issue **sp_changedbowner** in the database whose ownership will be changed. The syntax is:

```
sp_changedbowner loginame [, true ]
```

The following example makes the user "albert" the owner of the current database and drops aliases of users who could act as the old "dbo":

```
sp_changedbowner albert
```

To transfer aliases and their permissions to the new "dbo," add the second parameter with the value **true**.

➤ ***Note***

You cannot change the ownership of the *master* database and should not change the ownership of any other system databases.

Database Object Owner Privileges

A user who creates a database object (a table, view, or stored procedure) owns the object and is automatically granted all object access permissions on it. Users other than the object owner, including the owner of the database, are automatically denied all permissions on that object, unless they are explicitly granted by either the owner or a user who has **grant** permission on that object.

As an example, suppose that Mary is the owner of the *pubs2* database, and has granted Joe permission to create tables in it. Now Joe creates the table *new_authors*; he is the owner of this database object.

Initially, object access permissions on *new_authors* belong to Joe and Joe alone. Joe can grant or revoke object access permissions for this table to other users.

The following object creation permissions default to the owner of a table and cannot be transferred to other users:

alter table
drop table
create index
create trigger
truncate table
update statistics

Permission to use the **grant** and **revoke** commands to grant specific users **select, insert, update, delete, references,** and **execute** permissions on specific database objects can be transferred, using the **grant with grant option** command.

Permission to **drop** an object—a table, view, index, stored procedure, rule, or default—defaults to the object owner and cannot be transferred.

Privileges of Other Database Users

At the bottom of the hierarchy are other database users. Permissions are granted to or revoked from them by object owners, Database

Owners, users who were granted permissions with the grant option, or a System Administrator. These users are specified by user name, by group name, or by the keyword **public**.

Granting and Revoking Permissions on Database Objects

Two types of permissions exist for objects:

- **Object access permissions** – Permissions for using the **select**, **update**, **insert**, **delete**, **references**, and **execute** commands, which access database objects. The **references** permission refers to referential integrity constraints that you can specify in an **alter table** or **create table** command. The other permissions refer to SQL commands. For more information, see “Granting and Revoking Object Access Permissions” on page 5-10.
- **Object creation permissions** – Permissions for creating objects. They can be granted only by a System Administrator or a Database Owner. For more information, see “Granting and Revoking Object Creation Permissions” on page 5-16.

Both types of permissions are controlled with the **grant** and **revoke** commands.

Each database has its own independent protection system. Having permission to use a certain command in one database does not give you permission to use that command in other databases.

If you try to use a command or database object for which you have not been assigned permission, Adaptive Server displays an error message.

Granting and Revoking Object Access Permissions

Object access permissions regulate the use of certain commands that access certain database objects. For example, you must explicitly be granted permission to use the **select** command on the *authors* table. Object access permissions are granted and revoked by the object owner, who can grant them to other users.

Table 5-1 lists the types of object access permissions and the objects to which they apply:

Table 5-1: Permissions and the objects to which they apply

Permission	Object
select	Table, view, column
update	Table, view, column
insert	Table, view
delete	Table, view
references	Table, column
execute	Stored procedure

Object access permissions default to System Administrators and the object's owner, and can be granted to other users.

Use the **grant** command to grant object access permissions (permission to use tables, views, columns, and stored procedures). The syntax is:

```
grant {all [privileges]| permission_list}
   on { table_name [(column_list)]
        | view_name[(column_list)]
        | stored_procedure_name}
   to {public | name_list | role_name}
   [with grant option]
```

Use the revoke command to revoke object access permissions (permission to use tables, views, columns, and stored procedures). The syntax is:

```
revoke [grant option for]
   {all [privileges] | permission_list}
   on { table_name [(column_list)]
        | view_name [(column_list)]
        | stored_procedure_name}
   from {public | name_list | role_name}
   [cascade]
```

Notes on the keywords and parameters are as follows:

- **all** or **all privileges** specifies that all permissions applicable to the specified object are granted or revoked. All object owners can use **all** with an object name to grant or revoke permissions on their own objects. If you are granting or revoking permissions on a stored procedure, **all** is the same as **execute**.

➤ ***Note***

insert and **delete** permissions do not apply to columns, so you cannot include them in a permission list (or use the keyword **all**) if you specify a column list.

- *permission_list* is the list of permissions that you are granting. If you name more than one permission, separate them with commas. The following table illustrates the access permissions that can be granted on each type of object:

Table 5-2: Object access permissions

Object	*permission_list* Can Include
Table or view	**select**, **insert**, **delete**, **update**, **references.** **references** applies to tables but not views; the other permissions apply to both tables and views.
Column	**select**, **update**, **references**
Stored procedure	**execute**

  You can specify columns in the *permission_list* or the *column_list*, but not both.

- **on** specifies the object for which the permission is being granted or revoked. You can grant or revoke permissions for only one table, view, or stored procedure object at a time. You can grant or revoke permissions for more than one column at a time, but all the columns must be in the same table or view. You can only grant or revoke permissions on objects in your current database.
- **public** refers to the group "public", which includes all the users of Adaptive Server. **public** means slightly different things for **grant** and **revoke**:
  - For **grant**, **public** includes you, the object owner. Therefore, if you have revoked permissions from yourself on your object, and later you **grant** permissions to **public**, you regain the permissions along with the rest of "public".
  - For **revoke** on object access permissions, **public** excludes the owner. For **revoke** on object creation permissions, **public** excludes the Database Owner (who "owns" object creation permissions).
- *name_list* is a list of the names of:

  - Groups
  - Users
  - A combination of users and groups, each name separated from the next by a comma

- *role_name* is the name of an Adaptive Server system-defined or user-defined role. You can create and define a hierarchy of user-defined roles and grant them differing privileges based on the specific role granted. System-defined roles include **sa_role** (System Administrator), **sso_role** (System Security Officer), and **oper_role** (Operator). You cannot create or modify system-defined roles.
- **with grant option** in a **grant** statement allows the user(s) specified in *name_list* to grant the specified object access permission(s) to other users. If a user has **with grant option** permission on an object, that permission is not revoked when permissions on the object are revoked from **public** or a group of which the user is a member.
- The **grant option for** revokes **with grant option** permissions, so that the user(s) specified in *name_list* can no longer grant the specified permissions to other users. If those other users have granted permissions to other users, you must use the **cascade** option to revoke permissions from them as well. The user specified in *name_list* retains permission to access the object, but can no longer grant access to other users. **grant option for** applies only to object access permissions, not to object creation permissions.
- The **cascade** option in a **revoke** statement removes the specified object access permissions from the user(s) specified in *name_list*, and also from any users they granted those permissions to.

You may only grant and revoke permissions on objects in the current database.

Permissions granted to roles override permissions granted to users or groups. For example, suppose that John has been granted the System Security Officer role, and **sso_role** has been granted permission on the *sales* table. If John's individual permission on *sales* is revoked, he can still access *sales* because his role permissions override his individual permissions.

If several users grant access to an object to a particular user, the user's access remains until access is revoked by all those who granted access or until a System Administrator revokes the access. That is, if a System Administrator revokes access, the user is denied access even though other users have granted access.

Permission to issue the **create trigger** command is granted to users by default. When you revoke permission for a user to create triggers, a revoke row is added in the *sysprotects* table for that user. To grant permission to issue **create trigger**, you must issue two grant commands. The first command removes the revoke row from *sysprotects*; the second inserts a grant row. If you revoke permission to create triggers, the user cannot create triggers even on tables that the user owns. Revoking permission to create triggers from a user affects only the database where the revoke command was issued. Only a System Security Officer can grant or revoke permissions to create triggers.

Special Requirements for Compliance to the SQL92 Standard

When you have used the **set** command to turn **ansi_permissions** on, additional permissions are required for **update** and **delete** statements. The following table summarizes the required permissions.

Table 5-3: ANSI permissions for update and delete

	Permissions Required: *set ansi_permissions off*	**Permissions Required: *set ansi_permissions on***
update	**update** permission on columns where values are being set	**update** permission on columns where values are being set and **select** permission on all columns appearing in the **where** clause **select** permission on all columns on the right side of the **set** clause
delete	**delete** permission on the table	**delete** permission on the table from which rows are being deleted and **select** permission on all columns appearing in the **where** clause

If **ansi permissions** is on and you attempt to update or delete without having all the additional **select** permissions, the transaction is rolled back and you receive an error message. If this occurs, the object owner must grant you **select** permission on all relevant columns.

Examples of Granting Object Access Permissions

The following statement gives Mary and the "sales" group permission to insert into and delete from the *titles* table:

```
grant insert, delete
on titles
to mary, sales
```

The following statement gives Harold permission to use the stored procedure **makelist**:

```
grant execute
on makelist
to harold
```

The following statement grants permission to execute the stored procedure *sa_only_proc* to users who have been granted the System Administrator role:

```
grant execute
on sa_only_proc
to sa_role
```

The following statement gives Aubrey permission to select, update, and delete from the *authors* table and to grant the same permissions to other users:

```
grant select, update, delete
on authors
to aubrey
with grant option
```

Examples of Revoking Object Access Permissions

Both of the following statements revoke permission for all users except the table owner to update the *price* and *total_sales* columns of the *titles* table:

```
revoke update
on titles (price, total_sales)
from public

revoke update(price, total_sales)
on titles
from public
```

The following statement revokes permission from Clare to update the *authors* table and simultaneously revokes that permission from all users to whom she had granted that permission:

```
revoke update
on authors
from clare
cascade
```

The following statement revokes permission from Operators to execute the stored procedure *new_sproc:*

```
revoke execute
on new_sproc
from oper_role
```

Granting and Revoking Object Creation Permissions

Object creation permissions regulate the use of commands that create objects. These permissions can be granted only by a System Administrator or a Database Owner.

The object creation commands are:

create database
create default
create procedure
create rule
create table
create view

The syntax for object creation permissions differs slightly from the syntax for object access permissions. For object creation permissions the syntax for **grant** is:

```
grant {all [privileges] | command_list}
   to {public | name_list | role_name}
```

For object creation permissions the syntax for **revoke** is:

```
revoke {all [privileges] | command_list}
   from {public | name_list | role_name}
```

where:

- **all** or **all privileges** can be used only by a System Administrator or the Database Owner. When used by a System Administrator in the *master* database, **grant all** assigns all **create** permissions, including **create database**. If the System Administrator executes **grant all** from another database, all **create** permissions are granted except **create database**. When the Database Owner uses **grant all**, Adaptive Server grants all **create** permissions except **create database**, and prints an informational message.

- *command_list* is a list of the object creation permissions that you are granting or revoking. If more than one command is listed, separate them with commas. The command list can include **create database**, **create default**, **create procedure**, **create rule**, **create table**, and **create view. create database** permission can only be granted by a System Administrator, and only from within the *master* database.
- **public** is all users. For object creation permissions, **public** excludes the Database Owner (who "owns" object creation permissions within the database).
- *name_list* is a list of user or group names, separated by commas.
- *role_name* is the name of an Adaptive Server system or user-defined role. You can create and define a hierarchy of user-defined roles and grant them differing privileges based on the specific role granted.

Examples of Granting Object Creation Permissions

The following examples show how to grant and revoke object creation permissions.

The first example grants Mary and John permission to use the **create database** and **create table** commands. Because **create database** permission is being granted, this command can only be executed by a System Administrator within the *master* database. Mary and John's **create table** permission applies only to the *master* database.

```
grant create table, create database
to mary, john
```

The following command grants permission to create tables and views in the current database to all users:

```
grant create table, create view
to public
```

Example of Revoking Object Creation Permissions

The following example revokes permission to create tables and rules from "mary":

```
revoke create table, create rule
from mary
```

Combining *grant* and *revoke* Statements

There are two basic styles of setting up permissions in a database or on a database object. The most straightforward is to assign specific permissions to specific users. However, if most users are going to be granted most privileges, it is easier to assign all permissions to all users, and then revoke specific permissions from specific users.

For example, a Database Owner can grant all permissions on the *titles* table to all users by issuing the following statement:

```
grant all
on titles
to public
```

Then, the Database Owner can issue a series of **revoke** statements, for example:

```
revoke update
on titles (price, advance)
from public

revoke delete
on titles
from mary, sales, john
```

Avoiding Conflicts In Permissions

➤ ***Note***

The order in which **grant** and **revoke** commands are used is significant. Lack of attention to this fact can result in the wrong permissions being granted to or revoked from users.

The **grant** and **revoke** statements are sensitive to the order in which they are issued. So, for example, if Joe's group has been granted **select** permission on the *titles* table and then Joe's permission to select the *advance* column has been revoked, Joe can select all the columns except *advance*, while the other users in his group can still select all the columns.

A **grant** or **revoke** command that applies to a group changes any conflicting permissions that have been assigned to any member of that group. For example, suppose that the owner of the *titles* table has granted different permissions to various members of the *sales* group, and then decides to standardize. He or she might issue the following statements:

```
revoke all on titles from sales
grant select on titles(title, title_id, type,
    pub_id)
to sales
```

Similarly, a **grant** or **revoke** statement issued to **public** changes, for all users, any previously issued permissions that conflict with the new regime.

The same **grant** and **revoke** statements issued in different orders can create entirely different situations. For example, the following set of statements leaves Joe without any **select** permission on *titles*:

```
grant select on titles(title_id, title) to joe
revoke select on titles from public
```

In contrast, the same statements issued in the opposite order result in only Joe having **select** permission, and only on the *title_id* and *title* columns.

Remember that when you use the keyword **public** with **grant**, you are including yourself. With **revoke** on object creation permissions, you are included in **public** unless you are the Database Owner. With **revoke** on object access permissions, you are included in **public** unless you are the object owner. You may want to deny yourself permission to use your own table, while giving yourself permission to access a view built on it. To do this you must issue **grant** and **revoke** statements explicitly setting your permissions. (You can always change your mind and reinstitute the permission with a **grant** statement.)

Granting and Revoking Proxy Authorization and Roles

In addition to granting access to database objects, you can use the **grant** and **revoke** commands to grant and revoke:

- Roles to users or other roles
- Permission to execute **set proxy** or **set session authorization**

For information about granting and revoking roles, see “Granting and Revoking Roles” on page 4-16. For information about granting permission for proxy authorizations, see “Granting Permission to Use Proxy Authorization” on page 6-3.

Reporting on Permissions

Table 5-4 lists the system procedures to use to report information about object creation and object access permissions:

Table 5-4: System procedures for reporting on permissions

To Report Permissions Information On	Use
Database objects or users	**sp_helprotect**
Specific tables	**sp_table_privileges**
Specific columns in a table	**sp_column_privileges**

Reporting Permissions on Database Objects or Users

Use the system procedure **sp_helprotect** to report on permissions by database object or by user, and (optionally) by user for a specified object. Any user can execute this procedure. The syntax is:

```
sp_helprotect [name [, username [, "grant"
   [,"none"|"granted"|"enabled"|role_name]]]]
```

where:

name is either the name of the table, view, or stored procedure, or the name of a user or group in the current database. If you do not provide a name, **sp_helprotect** reports on all permissions in the database.

username is a user's name in the current database.

The first parameter, *name*, is either the name of the table, view, or stored procedure; or the name of a user, group, or role in the current database. If you specify the second parameter, *username*, only that user's permissions on the specified object are reported. If *name* is not an object, **sp_helprotect** checks whether *name* is a user, group, or role. If so, the permissions for the user, group, or role are listed. If you specify the third parameter, the keyword **grant**, and *name* is not an object, **sp_helprotect** displays all permissions granted by **with grant option.**

The fourth parameter specifies whether to include permissions for roles. The parameter can be one of the following values:

grant displays the permissions granted to *name* **with grant option**.

none ignores roles granted to the user when determining permissions granted.

granted includes information on all roles granted to the user when determining permissions granted.

enabled includes information on all roles activated by the user when determining permissions granted.

role_name displays permission information for the specified role only, regardless of whether this role has been granted to the user.

For example, suppose you issue the following series of **grant** and **revoke** statements:

```
grant select on titles to judy
grant update on titles to judy
revoke update on titles(contract) from judy
grant select on publishers to judy
    with grant option
```

To determine the permissions Judy now has on each column in the *titles* table, enter:

```
sp_helprotect titles, judy
```

The following display results:

```
grantor grantee type   action   object     column      grantable
------- ------ -----   ------   ------     ------      -------
dbo     judy   Grant   Select   titles     All         FALSE
dbo     judy   Grant   Update   titles     advance     FALSE
dbo     judy   Grant   Update   titles     notes       FALSE
dbo     judy   Grant   Update   titles     price       FALSE
dbo     judy   Grant   Update   titles     pub_id      FALSE
dbo     judy   Grant   Update   titles     pubdate     FALSE
dbo     judy   Grant   Update   titles     title       FALSE
dbo     judy   Grant   Update   titles     title_id    FALSE
dbo     judy   Grant   Update   titles     total_sales FALSE
dbo     judy   Grant   Update   titles     type        FALSE
```

The first row of the display shows that the Database Owner ("dbo") gave Judy permission to select all columns of the *titles* table. The rest of the lines indicate that she can update only the columns listed in the display. Judy's permissions are not grantable: she cannot give **select** or **update** permissions to any other user.

To see Judy's permissions on the *publishers* table, enter:

```
sp_helprotect publishers, judy
```

In the display below, the *grantable* column indicates TRUE, meaning that Judy can grant the permission to other users.

```
grantor grantee type   action   object     column      grantable
------- ------ -----   ------   ------     ------      -------
dbo     judy   Grant   Select   publishers all         TRUE
```

Reporting Permissions on Specific Tables

Use the catalog stored procedure **sp_table_privileges** to return permissions information about a specified table. The syntax is:

sp_table_privileges *table_name* [, *table_owner*
[, *table_qualifier*]]

where:

- *table_name* is the name of the table. It is required.
- *table_owner* can be used to specify the name of the table owner, if it is not "dbo" or the user executing **sp_table_privileges**.
- *table_qualifier* is the name of the current database.

Use **null** for parameters that you want to skip.

For example, the following statement:

```
sp_table_privileges titles
```

returns information about all permissions granted on the *titles* table. For more specific information about the output of **sp_table_privileges** see the *Adaptive Server Reference Manual*.

Reporting Permissions on Specific Columns

Use the catalog stored procedure **sp_column_privileges** to return information about permissions on columns in a table. The syntax is:

sp_column_privileges *table_name* [, *table_owner*
[, *table_qualifier* [, *column_name*]]]

where:

- *table_name* is the name of the table.
- *table_owner* can be used to specify the name of the table owner, if it is not "dbo" or the user executing **sp_column_privileges**.
- *table_qualifier* is the name of the current database.
- *column_name* is the name of the column on which you want to see permissions information.

Use **null** for parameters that you want to skip.

For example, the following statement:

```
sp_column_privileges publishers, null, null, pub_id
```

returns information about the *pub_id* column of the *publishers* table. For more specific information about the output of **sp_column_privileges**, see the *Adaptive Server Reference Manual*.

Using Views and Stored Procedures As Security Mechanisms

Views and stored procedures can serve as security mechanisms. You can give users controlled access to database objects via a view or stored procedure without granting them direct access to the data. For example, you might give a clerk **execute** permission on a procedure that updates cost information in a *projects* table without letting the user see confidential data in the table. To take advantage of this feature, you must own the procedure or view as well as its underlying objects. If you do not own the underlying objects, users must have permission to access the objects. For more information about when permissions are required, see "Understanding Ownership Chains" on page 5-26.

Adaptive Server makes permission checks, as required, when the view or procedure is used. When you create the view or procedure, Adaptive Server makes no permission checks on the underlying objects.

Using Views As Security Mechanisms

Through a view, users can query and modify only the data they can see. The rest of the database is neither visible nor accessible.

Permission to access the view must be explicitly granted or revoked, regardless of the set of permissions in force on the view's underlying tables. If the view and underlying tables are owned by the same owner, no permissions need to be given to the underlying tables. Data in an underlying table that is not included in the view is hidden from users who are authorized to access the view but not the underlying table.

By defining different views and selectively granting permissions on them, a user (or any combination of users) can be restricted to different subsets of data. The following examples illustrate the use of views for security purposes. Access can be restricted to:

- A subset of the rows of a base table (a value-dependent subset). For example, you might define a view that contains only the rows for business and psychology books in order to keep information about other types of books hidden from some users.
- A subset of the columns of a base table (a value-independent subset). For example, you might define a view that contains all the rows of the *titles* table, but omits the *price* and *advance* columns, since this information is sensitive.
- A row-and-column subset of a base table.
- The rows that qualify for a join of more than one base table. For example, you might define a view that joins the *titles*, *authors*, and *titleauthor* tables in order to display the names of the authors and the books they have written. This view would hide personal data about authors and financial information about the books.
- A statistical summary of data in a base table. For example, you might define a view that contains only the average price of each type of book.
- A subset of another view, or of some combination of views and base tables.

For example, suppose you want to prevent some users from accessing the columns in the *titles* table that have to do with money and sales. You could create a view of the *titles* table that omits those columns, and then give all users permission on the view but only the Sales Department permission on the table. The following example shows how:

```
grant all on bookview to public

grant all on titles to sales
```

An equivalent way of setting up these privilege conditions, without using a view, is to use the following statements:

```
grant all on titles to public

revoke select, update on titles (price, advance,
    total_sales)
from public

grant select, update on titles (price, advance,
    total_sales)
to sales
```

One possible problem with the second solution is that users not in the *sales* group who enter the command:

```
select * from titles
```

might be surprised to see the message that includes the phrase:

```
permission denied
```

Adaptive Server expands the asterisk into a list of all the columns in the *titles* table, and since permission on some of these columns has been revoked from non-sales users, access to these columns is denied. The error message lists the columns for which the user does not have access.

To see all the columns for which they do have permission, the non-sales users would have to name them explicitly. For this reason, creating a view and granting the appropriate permissions on it is a better solution.

In addition to protecting data based on a selection of rows and/or columns, views can be used for **context-sensitive protection**. For example, you can create a view that gives a data entry clerk permission to access only those rows that he or she has added or updated. In order to do so, you would add a column to a table in which the user ID of the user entering each row is automatically recorded with a default. You can define this default in the **create table** statement, like this:

```
create table testtable
    (empid       int,
    startdate    datetime,
    username     varchar(30) default user)
```

Next, define a view that includes all the rows of the table where *uid* is the current user:

```
create view context_view
as
    select *
    from testtable
    where username = user_name()
with check option
```

The rows retrievable through this view depend on the identity of the person who issues the **select** command against the view. By adding the **with check option** to the view definition, you make it impossible for any data entry clerk to falsify the information in the *username* column.

Using Stored Procedures As Security Mechanisms

If a stored procedure and all underlying objects are owned by the same user, that owner can grant users permission to use the procedure without granting permissions on the underlying objects. For example, a user might be given permission to execute a stored procedure that updates a row-and-column subset of a specified table, even though that user does not have any other permissions on that table.

Roles and Stored Procedures

You can use the **grant execute** command to grant execute permission on a stored procedure to all users who have been granted a specified role. Similarly, **revoke execute** removes this permission. But **grant execute** permission does not prevent users who do **not** have the specified role from being granted execute permission on the stored procedure.

For further security, you can restrict the use of a stored procedure by using the **proc_role** system function within the procedure to guarantee that a procedure can be executed only by users who have a given role. **proc_role** returns 1 if the user has a specific role (**sa_role**, **sso_role**, **oper_role**, or any user-defined role) and returns 0 if the user does not have that role. For example, here is a procedure that uses **proc_role** to see if the user has the System Administrator role:

```
create proc test_proc
as
if (proc_role("sa_role") = 0)
begin
    print "You don’t have the right role"
    return -1
end
else
    print "You have SA role"
    return 0
```

See “System Functions” in the *Adaptive Server Reference Manual* for more information about **proc_role**.

Understanding Ownership Chains

Views can depend on other views and / or tables. Procedures can depend on other procedures, views, and / or tables. These dependencies can be thought of as an **ownership chain**.

Typically, the owner of a view also owns its underlying objects (other views and tables), and the owner of a stored procedure owns all the procedures, tables, and views referenced by the procedure.

Also, a view and its underlying objects are usually all in the same database, as are a stored procedure and all the objects it references. However, it is not required that all of the underlying objects be in the same database. If these objects are in different databases, a user wanting to use the view or stored procedure must be a valid user or guest user in all of the databases containing the objects. (This prevents users from accessing a database unless the Database Owner has authorized it.)

When a user who has been granted **execute** permission on a procedure or view uses it, Adaptive Server does not check permissions on any of the underlying objects if:

- These objects and the view or procedure are owned by the same user, and
- The user accessing the view or procedure is a valid user or guest user in each of the databases containing the underlying objects.

However, if all objects are not owned by the same user, Adaptive Server checks object permissions when the ownership chain is broken. That is, if object A references object B, and B is not owned by the user who owns object A, Adaptive Server checks the permissions for object B. In this way, Adaptive Server allows the owner of the original data to retain control over who is authorized to access it.

Ordinarily, a user who creates a view need worry only about granting permissions on that view. For example, say Mary has created a view called *auview1* on the *authors* table, which she also owns. If Mary grants **select** permission to Sue on *auview1*, Adaptive Server will let Sue access it without checking permissions on *authors*.

However, a user who creates a view or stored procedure that depends on an object owned by another user must be aware that any permissions he or she grants depend on the permissions allowed by those other owners.

Example of Views and Ownership Chains

Say Joe creates a view called *auview2*, which depends on Mary's view *auview1*. Joe grants Sue **select** permission on *auview2*.

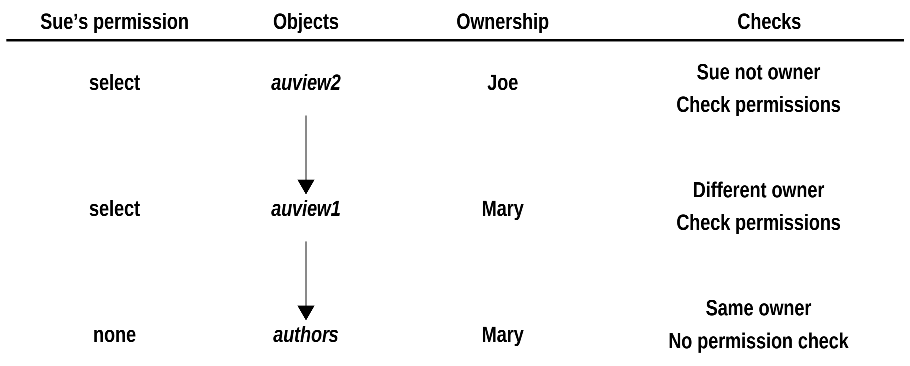

Figure 5-1: Ownership chains and permission checking for views, case 1

Adaptive Server checks the permissions on *auview2* and *auview1*, and finds that Sue can use them. Adaptive Server checks ownership on *auview1* and *authors* and finds that they have the same owner. Therefore, Sue can use *auview2*.

Taking this example a step further, suppose that Joe's view, *auview2*, depends on *auview1*, which depends on *authors*. Mary decides she likes Joe's *auview2* and creates *auview3* on top of it. Both *auview1* and *authors* are owned by Mary.

The ownership chain looks like this:

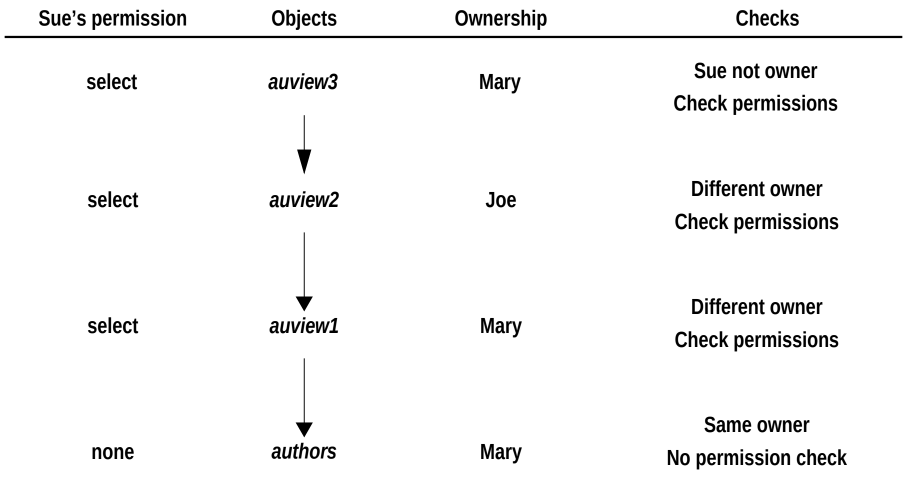

Figure 5-2: Ownership chains and permission checking for views, case 2

When Sue tries to access *auview3*, Adaptive Server checks permissions on *auview3*, *auview2*, and *auview1*. If Joe has granted permission to Sue on *auview2* and Mary has granted her permission on *auview3* and *auview1*, Adaptive Server allows the access. Adaptive Server checks permissions only if the object immediately before it in the chain has a different owner (or if it is the first object in the chain). For example, it checks *auview2* because the object before it—*auview3*—is owned by a different user. It does not check permission on *authors*, because the object which immediately depends on it, *auview1*, is owned by the same user.

Example of Procedures and Ownership Chains

Procedures follow the same rules as views. For example, suppose the ownership chain looks like this:

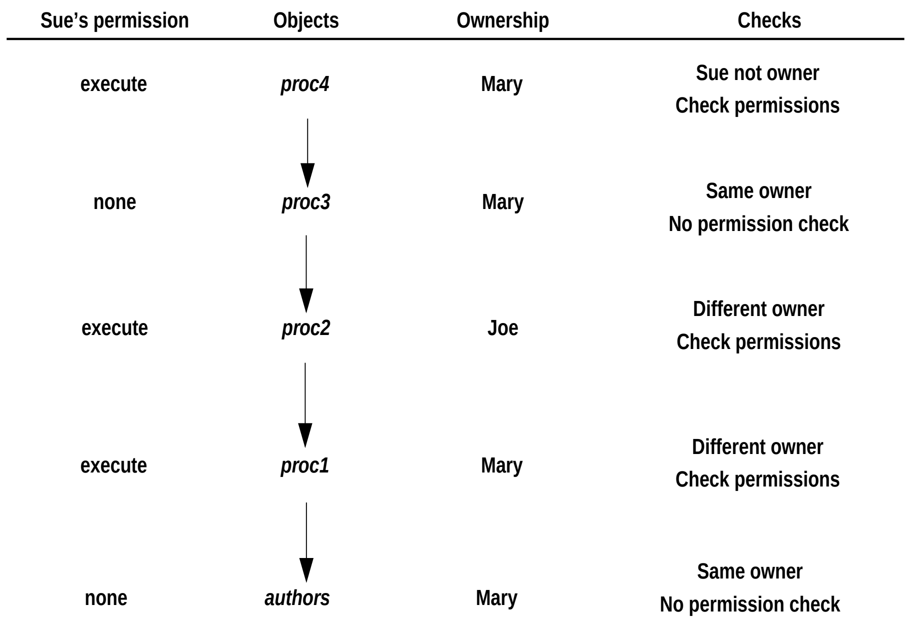

Figure 5-3: Ownership chains and permission checking for stored procedures

To execute *proc4*, Sue must have permission to execute *proc4*, *proc2*, and *proc1*. Permission to execute *proc3* is not necessary, since *proc3* and *proc4* have the same owner.

Adaptive Server checks Sue's permissions on *proc4* and all objects it references each time she executes *proc4*. Adaptive Server knows which referenced objects to check: it determined this the first time Sue executed *proc4*, and it saved the information with the procedure's execution plan. Unless one of the objects referenced by the procedure is dropped or otherwise redefined, Adaptive Server does not change its initial decision about which objects to check.

The purpose of this protection hierarchy is to allow every object's owner to fully control access to the object. Owners can control access to views and stored procedures, as well as to tables.

Permissions on Triggers

A **trigger** is a special kind of stored procedure used to enforce integrity, especially referential integrity. Triggers are never executed directly, but only as a side effect of modifying a table. There is no way to **grant** or **revoke** permissions for triggers.

Only the owner of an object can create a trigger on it. However, the ownership chain can be broken if a trigger on a table references objects owned by different users. The protection hierarchy rules that apply to procedures also apply to triggers.

While the objects that a trigger affects are usually owned by the user who owns the trigger, you can write a trigger that modifies an object owned by another user. If this is the case, any users modifying your object in a way that activates the trigger must have permission on the other object as well.

If Adaptive Server denies permission on a data modification command because a trigger affects an object for which the user does not have permission, the entire data modification transaction is rolled back.

For more information on triggers, see the *Transact-SQL User's Guide* or the *Adaptive Server Reference Manual*.

6 Using Proxy Authorization

This chapter describes the mechanism of **proxy authorization**, the ability for one user to assume the identity of another user on a server-wide basis. Topics include:

- Understanding Proxy Authorization 6-1
- Granting Permission to Use Proxy Authorization 6-3
- Setting Proxy Authorization 6-4
- Getting Information About Proxy Authorization 6-6

Understanding Proxy Authorization

With the proxy authorization capability of Adaptive Server, System Security Officers can grant selected logins the ability to assume the security context of another user. If a login has permission to use proxy authorization, the login can impersonate any other login in Adaptive Server.

System Security Officers use the **grant set proxy** or **grant set session authorization** command to give a user permission to impersonate another user within the server. The user with this permission can then execute either **set proxy** or **set session authorization** to become another user.

➤ ***Note***

set proxy and **set session authorization** are identical in function and can be used interchangeably. The only difference between them is that **set session authorization** is ANSI SQL92 compatible, and **set proxy** is a Transact-SQL extension.

A user executing **set proxy** or **set session authorization** operates with both the login name and server user ID of the user being impersonated. The login name is stored in the *name* column of *master..syslogins* and the server user ID is stored in the *suid* column of *master..syslogins*. These values are active across the entire server in all databases.

How Users Use Proxy Authorization

Security System Officers and System Administrators might want to assume the permissions of other users to make sure that permissions are correct for particular users or to perform maintenance on user's database objects.

◆ ***WARNING!***

The ability to assume another user's identity is extremely powerful and must be strictly limited. A user with this permission can even assume the identity of the "sa" login, and, thereby, have unlimited power within Adaptive Server. Plan to limit this permission to trusted administrators and applications and audit their server activity.

How Applications Use Proxy Authorization

The following illustration shows an application server logging into Adaptive Server with the generic login "appl" to execute procedures and commands for several users. While "appl" impersonates Tom, the application has Tom's permissions. Likewise, when "appl" impersonates Sue and John, the application has only Sue's and John's permissions, respectively.

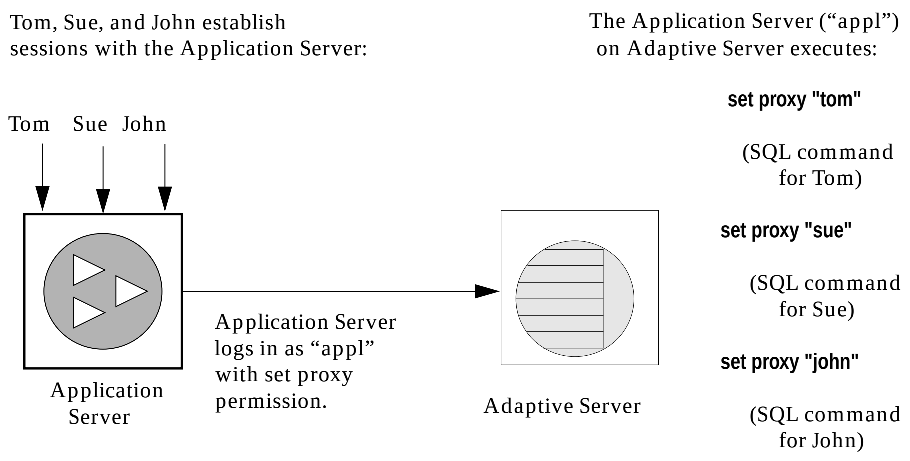
Tom, Sue, and John establish sessions with the Application Server:
The Application Server ("appl") on Adaptive Server executes:
Tom
Sue
John
set proxy "tom"
(SQL command for Tom)
set proxy "sue"
(SQL command for Sue)
set proxy "john"
(SQL command for John)
Application Server logs in as "appl" with set proxy permission.
Application Server
Adaptive Server

Figure 6-1: Applications and proxy authorization

With Proxy Authorization, an application can perform tasks in a controlled manner on behalf of different users.

Administering and Using Proxy Authorization

To use proxy authorization:

1. Use the **grant** command to grant a user the permission to execute **set proxy** or **set session authorization**.
2. As a login with permission to impersonate users, execute the **set proxy** or **set session authorization** commands.

Granting Permission to Use Proxy Authorization

To grant this permission, you must be a System Security Officer and execute the **grant** command from the *master* database. The syntax is:

```
grant set proxy
   to {public | name_list | role_name}
```

or

```
grant set session authorization
   to {public | name_list | role_name}
```

where:

- *public* is all users.

➤ **Note**

Sybase recommends that you not grant this permission to “public”. Instead, restrict the permission to carefully selected and trusted administrators and applications.

- *role_name* is the name of an Adaptive Server system or user-defined role. You can grant permissions to users based on the specific role granted.
- *name_list* is a list of users’ database names or group names, separated by commas. The user must be a valid user in the *master* database. For a list of valid users, execute:

  ```
  select name from sysusers
  ```

Examples of Granting *set proxy* Permission

To grant **set proxy** to "appl", if you do not have **sso_role** currently active, and you are not in the *master* database, execute:

```
use master
go
set role sso_role on
go
grant set proxy to appl
go
```

If "appl" is not a valid user in *master*, have a System Administrator add it to the *master* database with the **sp_adduser** system procedure.

To grant **set proxy** to a user-defined role, "accountant", execute:

```
grant set proxy to accountant
```

To grant **set session authorization** to the "sa" account, whose user name in every database is "dbo", execute:

```
grant set proxy to dbo
```

Setting Proxy Authorization

If you have a login that has been granted permission to use **set proxy** or **set session authorization**, you can use these commands to impersonate another user. The syntax is:

```
set proxy login_name
```

or

```
set session authorization login_name
```

where:

login_name is the name of a valid login in *master..syslogins*. Enclose the login name within quotation marks.

➤ ***Note***

set session authorization is identical to **set proxy**, with this exception: **set session authorization** follows the ANSI SQL92 standard, while **set proxy** is a Transact-SQL extension.

For example, to set proxy to "mary", execute:

```
set proxy mary
```

Follow these rules when you execute **set proxy** or **set session authorization**:

- The **set proxy** or **set session authorization** commands cannot be executed from within a transaction.
- You can execute **set proxy** or **set session authorization** from any database that you are allowed to use. However, the *login_name* you specify must be a valid user in the database, or the database must have a "guest" user defined for it.
- Only one level is permitted; if you want to impersonate more than one user, you must return to your original identity between impersonations.
- If you execute **set proxy** or **set session authorization** from within a procedure, your original identity is automatically resumed upon exit from the procedure.

Examples of Setting Proxy Authorization

Assume that your login is "ralph" and that you have been granted **set proxy** authorization. You want to execute some commands as "sallyn" and as "rudolph" in database "purchasing". "sallyn" has a valid name ("sally") in the database, but Ralph and Rudolph do not. However, "purchasing" has a guest user defined. After setting proxy, you want to check your login name in the server and your user name in the database. You can execute:

```
set proxy "sallyn"
go
use purchasing
go
select suser_name(), user_name()
go
```

Adaptive Server displays:

```
------------------------------ -------------------
sallyn                         sally
```

Then execute:

```
set proxy "ralph"
select suser_name(), user_name()
go
```

Adaptive Server displays:

```
------------------------------ --------------------
ralph                          guest
```

Notice that Ralph is a "guest" in the database because Ralph is not a valid user in the database.

Then execute:

```
set proxy "rudolph"
go
select suser_name(), user_name()
go
```

Adaptive Server displays:

```
------------------------------ --------------------
rudolph                        guest
```

Rudolph is also a guest in the database because Rudolph is not a valid user in the database.

Now, suppose that you decide to impersonate the “sa” account. Execute:

```
set proxy "ralph"
go
set proxy "sa"
go
select suser_name(), user_name()
go
```

Adaptive Server displays:

```
--------------------------- --------------------

 sa                         dbo
```

Getting Information About Proxy Authorization

Table 6-1 lists the methods you can use to get information about proxy authorization

Table 6-1: Methods for getting information about proxy authorization

If You Want To	Use This Method
Display information about permissions that have been granted to, or revoked from, users, groups, and roles	Query the *sysprotects* table
Display information about permissions for database objects, users, or groups	Execute **sp_helprotect**
Display information about all current Adaptive Server users and processes or about a particular user or process	Execute **sp_who**

Displaying Permissions Information

You can use two methods to obtain permissions information:

- Query the *sysprotects* table
- Execute **sp_helprotect**

Querying the *systprotects* Table

To display information about permissions that have been granted to, or revoked from, users, groups, and roles, query the *sysprotects* table. The *action* column of the table specifies the permission. For example, the *action* value for **set proxy** or **set session authorization** is equal to 167.

You might execute this query:

```
select * from sysprotects where action = 167
```

The results provide the user id of the user who granted or revoked the permission (column *grantor*), the user id of the user who has the permission (column *uid*), and the type of protection (column *protecttype*). The *protecttype* column can contain these values:

- 0 for **grant with grant**
- 1 for **grant**
- 2 for **revoke**

For more information about the *sysprotects* table, see **sysprotects** in the *Adaptive Server Reference Manual*.

Using *sp_helprotect*

The **sp_helprotect** system procedure displays information about permissions for database objects, users, or groups. For example, to display all the permissions that “judy” has in the database, execute:

```
sp_helprotect judy
```

For more information, see **sp_helprotect** in the *Adaptive Server Reference Manual*.

Displaying Information about Users and Processes

The **sp_who** system procedure displays information about all current Adaptive Server users and processes or about a particular user or process. The results of **sp_who** include the *loginame* and *origname*. If a user is operating under a proxy, *origname* contains the name of the original login. For example, assume that "ralph" executed:

```
set proxy susie
```

and then executes some SQL commands.

sp_who returns "susie" for *loginame* and "ralph" for *origname*.

The **sp_who** system procedure queries the *master..sysprocesses* system table, which contains columns for the server user ID (*suid*) and the original server user ID (*origsuid*).

For more information, see **sp_who** in the *Adaptive Server Reference Manual*.

7 Managing Remote Servers

This chapter discusses the steps the System Administrator and System Security Officer of each Adaptive Server must execute to enable **remote procedure calls** (RPCs). Topics include:

- Overview 7-1
- Managing Remote Servers 7-3
- Adding Remote Logins 7-7
- Example of Remote User Login Mapping 7-10
- Password Checking for Remote Users 7-11
- Configuration Parameters for Remote Logins 7-13
- Getting Information About Remote Logins 7-13

Overview

Users on a local Adaptive Server can execute stored procedures on a remote Adaptive Server. Executing an RPC sends the results of the remote process to the calling process—usually displayed on the user's screen.

➤ ***Note***

The use of remote servers is not included in the **evaluated configuration**.

To enable RPCs, the System Administrator and System Security Officer of each Adaptive Server must execute the following steps:

- On the local server:
  - (System Security Officer) Use **sp_addserver** to list the local server and remote server in the system table *master..sysservers*.
  - List the remote server in the interfaces file or Directory Service for the local server.
  - Reboot the local server so the global variable *@@servername* is set to the name of the local server. If this variable is not set properly, users cannot execute remote procedure calls from the local server on any remote server.

- On the remote server:
  - (System Security Officer) Use **sp_addserver** to list the server originating the RPC in the system table *master..sysservers*.
  - To allow the user who is originating the remote procedure access to the server, a System Security Officer uses **sp_addlogin**, and a System Administrator uses **sp_addremotelogin**.
  - Add the remote login name as a user of the appropriate database and grant that login permission to execute the procedure. (If **execute** permission is granted to "public", the user does not need to be granted specific permission.)

Figure 7-1 provides an example of setting up servers for remote access.

The user "joe" on ROSE needs to access stored procedures on ZINNIA

sp_addserver ROSE, local
sp_addserver ZINNIA

Interfaces files must have an entry for ZINNIA

sp_addserver ROSE
sp_addlogin joe
sp_addremotelogin ROSE, joe

sp_adduser joe (in the appropriate database)
grant execute on *procedure_name* to joe

Figure 7-1: Setting up servers to allow remote procedure calls

For operating system-specific information about handling remote servers, see the installation documentation for your platform.

Managing Remote Servers

Table 7-1 lists the tasks related to managing remote servers and the system procedures you use to perform the tasks.

Table 7-1: Tasks related to managing remote servers

To Do This	Use	See
Add a remote server	**sp_addserver**	"Adding a Remote Server" on page 7-3
Manage remote server names	**sp_addserver**	"Managing Remote Server Names" on page 7-5
Change server connection options	**sp_serveroption**	"Setting Server Connection Options" on page 7-5
Display information about servers	**sp_helpserver**	"Getting Information About Servers" on page 7-7
Drop a server	**sp_dropserver**	"Dropping Remote Servers" on page 7-7

Adding a Remote Server

A System Security Officer uses the system procedure **sp_addserver** to add entries to the *sysservers* table. On the server originating the call, you must add one entry for the local server, and one for each remote server that your server will call.

When you create entries for a remote server, you can choose to:

- Always refer to them by the name listed in the interfaces file, or
- Provide a local name for the remote server. For example, if the name in the interfaces file is "MAIN_PRODUCTION", you may want to call it simply "main".

The syntax is:

```
sp_addserver lname [{, local | null}
   [, pname]]
```

where:

- *lname* provides the local "call name" for the remote server. Users will type this name when they issue remote procedure calls. If this name is **not** the same as the remote server's name in the interfaces file, you must provide that name as the third parameter, *pname*.

The remote server must be listed in the interfaces file on the local machine. If it's not listed, copy the interfaces file entry from the remote server and append it to your existing interfaces file. Be sure to keep the same port numbers.

- **local** identifies the server being added as a local server. The **local** value is used only after start-up, or after a reboot, to identify the local server name so that it can appear in messages printed out by Adaptive Server. **null** specifies that this server is a remote server.

➤ ***Note***

For users to be able to run RPCs successfully from the local server, the local server must be added with the **local** option and rebooted. The rebooting is required to set the global variable *@@servername*. If this parameter is not set properly, RPCs cannot be run from the local server.

- *pname* is the name for the remote server that is listed in the interfaces file for the server named *lname*. This optional third argument permits you to establish local aliases for any other Adaptive Server, Open Server™, or Backup Server that you may need to communicate with. If you do not specify *pname*, it defaults to *lname*.

The server originating a remote procedure call must have a local entry. You must restart Adaptive Server to set the value of the global variable *@@servername*.

Examples of Adding Remote Servers

The following example creates an entry for the local server named DOCS:

```
sp_addserver DOCS, local
```

The next example creates an entry for a remote server named GATEWAY:

```
sp_addserver GATEWAY
```

To run a remote procedure such as **sp_who** on the GATEWAY server, execute:

```
GATEWAY.sybsytemprocs.dbo.sp_who
```

Or:

```
GATEWAY...sp_who
```

The following example gives a remote server called MAIN_PRODUCTION the local alias "main":

```
sp_addserver main, null, MAIN_PRODUCTION
```

The user can then enter:

```
main...sp_who
```

Managing Remote Server Names

The *master.dbo.sysservers* table has two name columns:

- *srvname* is the name that users must supply when executing remote procedure calls. *srvname* must be unique on each server.
- *srvnetname* is the server's network name, which must match the name in the *interfaces* file.

If you need to add or drop servers from your network, you can use **sp_addserver** to update the server's network name in *srvnetname*.

For example, to remove the server MAIN from the network, and move your remote applications to TEMP, you can use the following statement to change the network name, while keeping the local alias:

```
sp_addserver MAIN, null, TEMP
```

The **sp_addserver** system procedure displays a message telling you that it is changing the network name of an existing server entry.

Setting Server Connection Options

The **sp_serveroption** system procedure sets the server **timeouts** and **net password encryption** options, which affect connections with remote servers. The options you specify for **sp_serveroption** do not affect the communication between Adaptive Server and Backup Server.

The following sections describe the **timeouts**, and **net password encryption** options.

Using the *timeouts* Option

A System Administrator can use the **timeouts** option to disable and enable the normal timeout code used by the local server, so the site connection handler does not automatically drop the physical connection after a minute with no logical connection.

By default, **timeouts** is set to **true**, and the site handler process that manages remote logins times out if there has been no remote user activity for one minute. By setting **timeouts** to **false** on both of the servers involved in remote procedure calls, the automatic timeout is disabled. The following example changes the **timeouts** option to **false**:

```
sp_serveroption GATEWAY, "timeouts", false
```

After you set **timeouts** to **false** on both servers, when a user executes a remote procedure call in either direction, the site handler on each machine runs until one of the servers is shut down. (When the server is brought up again, the option remains **false**, and the site handler will be reestablished the next time a user executes a remote procedure call.) If users on the server execute remote procedure calls frequently, it is probably efficient in terms of system resources to set this option to **false**, since there is some system overhead involved in setting up the physical connection.

Using the *net password encryption* Option

A System Security Officer can use the **net password encryption** option to specify whether connections with a remote server are to be initiated with a client-side password encryption handshake or with the usual (unencrypted password) handshake sequence. The default is **false**.

If **net password encryption** is set to **true**, the following occurs:

1. The initial login packet is sent without passwords.
2. The client indicates to the remote server that encryption is desired.
3. The remote server sends back an encryption key, which the client uses to encrypt its plaintext passwords.
4. The client then encrypts its own passwords, and the remote server uses the key to authenticate them when they arrive.

The following example sets this option to **true**:

```
sp_serveroption GATEWAY, "net password encryption",
    true
```

Setting this option does not effect Adaptive Server's interaction with Backup Server.

Getting Information About Servers

The system procedure **sp_helpserver** reports on servers. When it is used without an argument, it provides information about all the servers listed in *sysservers*. When a server name is used as an argument, it provides information about that server only. The syntax is:

```
sp_helpserver [server]
```

sp_helpserver checks for both *srvname* and *srvnetname* in the *master..sysremotelogins* table.

For operating system-specific information about setting up remote servers, see the installation documentation for your platform, which contains detailed examples of setting up your environment for remote procedure calls.

Dropping Remote Servers

A System Security Officer can use the **sp_dropserver** system procedure to drop servers from *sysservers*. The syntax is:

```
sp_dropserver server [, droplogins]
```

where:

- *server* is the name of the server you want to drop.
- The **droplogins** option allows you to drop a remote server and all of that server's remote login information in one step. If you do not use the **droplogins** option, you cannot drop a server that has remote logins associated with it.

The following statement drops the GATEWAY server and all of the remote logins associated with it:

```
sp_dropserver GATEWAY, droplogins
```

The **droplogins** option isn't needed if you want to drop the local server; that entry will not have remote login information associated with it.

Adding Remote Logins

The System Security Officer and System Administrator of any Adaptive Server share control over which remote users can access the server, and what identity the remote users assume. The System Administrator uses **sp_addremotelogin** to add remote logins and **sp_dropremotelogin** to drop remote logins. The System Security Officer uses **sp_remoteoption** to control whether password checking will be required.

Mapping Users' Server IDs

Logins from a remote server can be mapped to a local server in three ways:

- A particular remote login can be mapped to a particular local login name. For example, user "joe" on the remote server might be mapped to "joesmith".
- All logins from one remote server can be mapped to one local name. For example, all users sending remote procedure calls from the MAIN server might be mapped to "remusers".
- All logins from one remote server can use their remote names.

The first option can be combined with the other two options, and its specific mapping takes precedence over the other two more general mappings.

The second and third options are mutually exclusive; you may use either of them, but not both. To change from one of these options to the other, use **sp_dropremotelogin** to remove the old mapping. For information about **sp_dropremotelogin**, see the *Adaptive Server Reference Manual*.

Use **sp_addremotelogin** to add remote logins. The syntax is:

```
sp_addremotelogin remoteserver [, loginame
   [, remotename]]
```

Only a System Administrator can execute **sp_addremotelogin**. If the local names are not listed in *master..syslogins*, add them as Adaptive Server logins with **sp_addlogin** before adding the remote logins.

Mapping Remote Logins to Particular Local Names

The following example maps the login named "pogo" from a remote system to the local login name "bob". The user logs into the remote system as "pogo". When that user executes remote procedure calls from GATEWAY, the local system maps the remote login name to "bob".

```
sp_addlogin bob
sp_addremotelogin GATEWAY, bob, pogo
```

Mapping All Remote Logins to One Local Name

The following example creates an entry that maps all remote login names to the local name "albert". All names are mapped to "albert", except those with specific mapping, as described in the previous section. For example, if you mapped "pogo" to "bob", and then the rest of the logins to "albert", "pogo" still maps to "bob".

```
sp_addlogin albert
sp_addremotelogin GATEWAY, albert
```

If you use **sp_addremotelogin** to map all users from a remote server to the same local name, use **sp_remoteoption** to specify the "trusted" option for those users. For example, if all users from server GATEWAY that are mapped to "albert" are to be trusted, specify:

```
sp_remoteoption GATEWAY, albert, NULL, trusted
    true
```

If you do not specify the logins as trusted, the logins will not be allowed to execute RPCs on the local server unless they specify passwords for the local server when they log into the remote server. Users, when they use Open Client Client-Library can use the routine **ct_remote_pwd** to specify a password for server-to-server connections. **isql** and **bcp** do not permit users to specify a password for RPC connections. See "Password Checking for Remote Users" on page 7-11 for more information about **sp_remoteoption**.

◆ ***WARNING!***

Mapping more than one remote login to a single local login is not recommended, as it reduces individual accountability on the server. Audited actions can be traced only to the local server login, not to the individual logins on the remote server.

Keeping Remote Login Names for Local Servers

To enable remote users to keep their remote login names while using a local server:

1. Use **sp_addlogin** to create a login for each login from the remote server.
2. Use **sp_addremotelogin** for the server as a whole to create an entry in *master..sysremotelogins* with a null value for the remote login name and a value of -1 for the *suid*. For example:

```
sp_addremotelogin GATEWAY
```

Example of Remote User Login Mapping

The following statement displays the local and remote server information recorded in *master..sysservers*:

```
select srvid, srvname from sysservers
```

```
srvid  srvname
-----  ----------
    0  SALES
    1  CORPORATE
    2  MARKETING
    3  PUBLICATIONS
   4  ENGINEERING
```

The SALES server is local. The other servers are remote.

The following statement displays information about the remote servers and users stored in *master..sysremotelogins*:

```
select remoteserverid, remoteusername, suid
  from sysremotelogins
```

```
remoteserverid   remoteusername   suid
--------------   --------------   ------
1                joe              1
1                nancy            2
1                NULL             3
3                NULL             4
4                NULL             -1
```

By matching the value of *remoteserverid* in this result and the value of *srvid* in the previous result, you can find the name of the server for which the *remoteusername* is valid. For example, in the first result, *srvid* 1 indicates the CORPORATE server; in the second result *remoteserverid* 1 indicates that same server. Therefore, the remote user login names "joe" and "nancy" are valid on the CORPORATE server.

The following statement shows the entries in *master..syslogins*:

```
select suid, name from syslogins
```

```
suid    name
------  ------------
    1   sa
    2   vp
    3   admin
    4   writer
```

The results of all three queries together show:

- The remote user name “joe” (*suid* 1) on the remote CORPORATE server (*srvid* and *remoteserverid* 1) is mapped to the “sa” login (*suid* 1).
- The remote user name “nancy” (*suid* 2) on the remote CORPORATE server (*srvid* and *remoteserverid* 1) is mapped to the “vp” login (*suid* 2).
- The other logins from the CORPORATE server (*remoteusername* “NULL”) are mapped to the “admin” login (*suid* 3).
- All logins from the PUBLICATIONS server (*srvid* and *remoteserverid* 3) are mapped to the “writer” login (*suid* 4).
- All logins from the ENGINEERING server (*srvid* and *remoteserverid* 4) are looked up in *master..syslogins* by their remote user names (*suid* -1).
- There is no *remoteserverid* entry for the MARKETING server in *sysremotelogins*. Therefore, users who log into the MARKETING server cannot run remote procedure calls from that server.

The remote user mapping procedures and the ability to set permissions for individual stored procedures give you control over which remote users can access local procedures. For example, you can allow the “vp” login from the CORPORATE server to execute certain local procedures and all other logins from CORPORATE to execute the procedures for which the “admin” login has permission.

➤ ***Note***

In many cases, the passwords for users on the remote server must match passwords on the local server. For details, see “Password Checking for Remote Users”.

Password Checking for Remote Users

A System Security Officer can use the system procedure **sp_remoteoption** to determine whether passwords will be checked when remote users log into the local server. By default, passwords are verified (“untrusted” mode). In **trusted** mode, the local server accepts remote logins from other servers and front-end applications without user-access verification for the particular login.

When **sp_remoteoption** is used with arguments, it changes the mode for the named user. The syntax is:

```
sp_remoteoption [remoteserver, loginame, remotename,
   optname, {true | false}]
```

The following example sets **trusted** mode for the user “bob”:

```
sp_remoteoption GATEWAY, pogo, bob, trusted,
    true
```

Effects of Using the Untrusted Mode

The effects of the “untrusted” mode depend on the user’s client program. **isql** and some user applications require that logins have the same password on the remote server and the local server. Open Client™ applications can be written to allow local logins to have different passwords on different servers.

To change your password in “untrusted” mode, you must first change it on all the remote systems you access before changing it on your local server. This is because of the password checking. If you change your password on the local server first, when you issue the remote procedure call to execute **sp_password** on the remote server your passwords will no longer match.

The syntax for changing your password on the remote server is:

```
remote_server...sp_password caller_passwd, new_passwd
```

On the local server, the syntax is:

```
sp_password caller_passwd, new_passwd
```

See “Changing Passwords” on page 3-14 for more information about changing your password.

Trusted Mode and Mapping Remote Logins

If you have used **sp_addremotelogin** to map all users from a remote server to the same local name, specify the “trusted” option for those users. For example, if all users from server GOODSRV that are mapped to “albert” are to be trusted, specify:

```
sp_remoteoption GOODSRV, albert, NULL, trusted
  true
```

If you do not specify the logins as trusted, the logins will not be allowed to execute RPCs on the local server unless they specify passwords for the local server when they log into the remote server. Users, when they use Open Client Client-Library can use the routine **ct_remote_pwd** to specify a password for server-to-server connections.

isql and **bcp** do not permit users to specify a password for RPC connections.

Getting Information About Remote Logins

The **sp_helpremotelogin** stored procedure prints information about the remote logins on a server. The following example shows the remote login "pogo" mapped locally to login name "bob", with all other remote logins keeping their remote names.

```
sp_helpremotelogin
```

```
server     remote_user_name     local_user_name      options
-------    ----------------     -----------------    --------
GATEWAY    **mapped locally**   **use local name**   untrusted
GATEWAY    pogo                 bob                  untrusted
```

Configuration Parameters for Remote Logins

Each Adaptive Server that allows remote logins to execute procedure calls must have certain configuration parameters set. You set these values with **sp_configure**. Table 7-2 shows the configuration parameters that affect remote procedure calls. All these configuration parameters are static and do not take effect until Adaptive Server is restarted.

Table 7-2: Configuration parameters that affect RPCs

Configuration Parameter	Default
allow remote access	1
number of remote logins	20
number of remote sites	10
number of remote connections	20
remote server pre-read packets	3

Allowing Remote Access

To allow remote access to or from a server, including access to Backup Server set **allow remote access** to 1:

```
sp_configure "allow remote access", 1
```

To disallow remote access at any time, set **allow remote access** to 0:

```
sp_configure "allow remote access", 0
```

Only a System Security Officer can set the **allow remote access** parameter.

◆ ***WARNING!***

You cannot perform database or transaction log dumps while the allow remote access parameter is set to 0.

Controlling the Number of Active User Connections

To control the number of active user connections from this site to remote servers, use the **number of remote logins** configuration parameter. The following command sets **number of remote logins** to 50:

```
sp_configure "number of remote logins", 50
```

Only a System Administrator can set the **number of remote logins** parameter.

Controlling the Number of Remote Sites

To control the number of remote sites that can access a server simultaneously, use the **number of remote sites** configuration parameter. All accesses from an individual site are managed by one site handler. This parameter controls the number of site handlers, not the number of individual, simultaneous procedure calls. For example, if you set **number of remote sites** to 5, and each site initiates three remote procedure calls, **sp_who** shows 5 site handler processes for the 15 processes. Only a System Administrator can set **number of remote sites**.

Controlling the Number of Active Remote Connections

To control the limit on active remote connections that are initiated to and from a server, use the **number of remote connections** parameter. This parameter controls connections initiated from the server and connections initiated from remote sites to the server. Only a System Administrator can set **number of remote connections**.

Controlling Number of Pre-read Packets

All communication between two servers is handled through one site handler to reduce the needed number of connections. This site handler can pre-read and keep track of data packets for each user before the user process that needs them is ready.

To control how many packets a site handler will pre-read, use the **remote server pre-read packets** parameter. The default value, 3, is adequate in all cases; higher values can use too much memory. Only a System Administrator can set **remote server pre-read packets**.For more information about configuration parameters, see Chapter 11, “Setting Configuration Parameters,” in the *System Administration Guide*.

8 Auditing

This chapter describes how to set up auditing for your installation. Topics include:

- Overview 8-1
- Establishing and Managing Auditing 8-6
- Installing the Audit System 8-7.
- Preparing to Manage the Audit Trail 8-10
- Preparing to Manage the Transaction Log 8-18
- Setting Auditing Options 8-20
- Enabling and Disabling Auditing 8-21
- Adding User-Specified Records to the Audit Trail 8-22
- Querying the Audit Trail 8-23
- Correlating Adaptive Server and Operating System Audit Records 8-24
- Understanding the Audit Tables 8-24

Overview

Auditing is an important part of security in a database management system. It records security-related system activity in an audit trail, which can be used to detect penetration of the system and misuse of resources. By examining the audit trail, a System Security Officer can inspect patterns of access to objects in databases and can monitor the activity of specific users. Audit records are traceable to specific users, enabling the audit system to act as a deterrent to users who are attempting to misuse the system.

A System Security Officer manages the audit system and is the only user who can start and stop auditing, set up auditing options, and process the audit data. As a System Security Officer, you can establish auditing for:

- All server-wide security-relevant events
- Creation, deletion, and modification of database objects
- All actions by users with a particular role active, such as all actions performed by a System Administrator

- Granting or revoking of database access
- Importing or exporting of data
- Logins to and logouts from Adaptive Server

The System Security Officer can also add user-defined audit records to the audit trail.

The Audit System

The audit system consists of:

- The *sybsecurity* database, which contains global auditing options and the audit trail
- The in-memory audit queue, to which audit records are sent before they are written to the audit trail
- Configuration parameters for managing auditing
- System procedures for managing auditing

The *sybsecurity* Database

The *sybsecurity* database is essential to auditing in Adaptive Server. It is created during the auditing installation process. In addition to all the system tables found in the *model* database, it contains a system table for keeping track of server-wide auditing options and system tables for the audit trail.

Global Auditing Options: sysauditoptions

The *sysauditoptions* system table contains the current setting of global auditing options, such as whether auditing is enabled for disk commands, remote procedure calls, ad hoc user-defined auditing records, or all security-relevant events. These options affect the entire Adaptive Server.

The Audit Trail: sysaudits_01...sysaudits_08

Adaptive Server stores the audit trail in system tables named *sysaudits_01* through *sysaudits_08*. When you install auditing, you determine the number of audit tables for your installation. For example, if you choose to have two audit tables, they are named *sysaudits_01* and *sysaudits_02*. At any given time, only **one** audit table is **current**. Adaptive Server writes all audit data to the current audit table. A System Security Officer can use **sp_configure** to set, or change, which audit table is current.

The recommended number of tables is two or more with each table on a separate audit device. If you follow this recommendation, you can set up a smoothly running auditing process in which audit tables are archived and processed with no loss of audit records and no manual intervention.

◆ ***WARNING!***

It is possible to set up auditing to use a single audit table; however, Sybase strongly recommends against doing this on production systems. When using a single audit table there is a window of time during which audit records may be lost. If you must use only a single audit table, because of limited system resources, refer to Appendix A, “Single-Table Auditing” for instructions.

Figure 8-1 shows how the auditing process works with multiple audit tables.

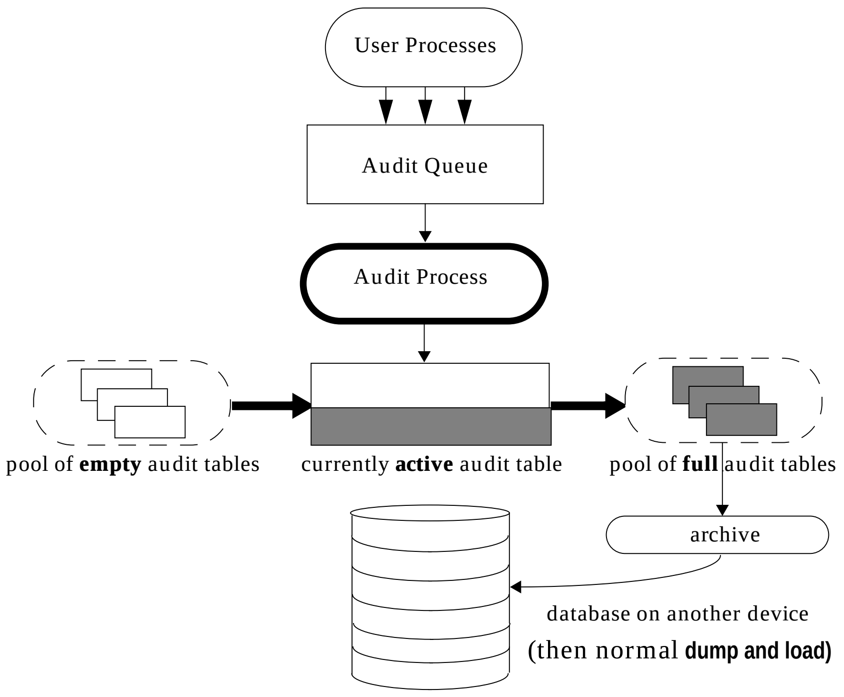
User Processes
Audit Queue
Audit Process
pool of **empty** audit tables
currently **active** audit table
pool of **full** audit tables
archive
database on another device
(then normal **dump and load**)

Figure 8-1: Auditing with multiple audit tables

The auditing system writes audit records from the in-memory audit queue to the current audit table. When the current audit table is nearly full, a threshold procedure can automatically archive the table to another database ready for processing by a System Security Officer. The archive database, just as any standard Adaptive Server database, can be backed up and restored with the **dump** and **load** commands. For more information about managing the audit trail, see "Preparing to Manage the Audit Trail" on page 8-10.

The Audit Queue

When an audited event occurs, an audit record first goes to the in-memory audit queue. The record remains in memory until the audit process writes it to the audit trail. You can configure the size of the audit queue with the **audit queue size** parameter of **sp_configure**.

Before configuring the size of the audit queue, consider the tradeoff between the risk of losing records in the queue if the system crashes and the loss of performance when the queue is full. As long as an audit record is in the queue, it can be lost if the system crashes. However, if the queue repeatedly becomes full, overall system performance is affected. If the audit queue is full when a user process tries to generate an audit record, the process sleeps until space in the queue becomes available.

➤ ***Note***

Because audit records are not written directly to the audit trail, you cannot count on an audit record's being stored immediately in the current audit table.

The Auditing Configuration Parameters

Use the following configuration parameters to manage the auditing process:

- **auditing** enables or disables auditing for the whole Adaptive Server. The parameter is dynamic and takes effect immediately upon execution of **sp_configure**. Auditing occurs only when this parameter is enabled.
- **audit queue size** establishes the size of the audit queue. Because the parameter affects memory allocation, the parameter is static and does not take effect until Adaptive Server is restarted.
- **suspend auditing when full** controls the behavior of the audit process when an audit device becomes full. The parameter is dynamic and takes effect immediately upon execution of **sp_configure**.
- **current audit table** sets the current audit table. The parameter is dynamic and takes effect immediately upon execution of **sp_configure**.

The Auditing System Procedures

Use the following system procedures to manage the auditing process:

- **sp_audit** enables and disables auditing options. This is the only system procedure required to establish the events to be audited.
- **sp_displayaudit** displays the active auditing options.
- **sp_addauditrecord** adds user-defined audit records (comments) into the audit trail. Users can add these records only if a System Security Officer enables ad hoc auditing with **sp_audit**.

Establishing and Managing Auditing

Table 8-1 provides a general procedure for using the auditing functions provided by Adaptive Server.

Table 8-1: General procedure for auditing

Action	Description	See
1. Install auditing.	Installation of auditing, which involves setting the number of audit tables and assigning devices for the audit trail and the *syslogs* transaction log in the *sybsecurity* database.	"Installing the Audit System" on page 8-7 and the Adaptive Server installation and configuration documentation.
2. Set up the audit process to manage the audit trail.	Writing and establishing a threshold procedure that receives control when the current audit table is nearly full. The procedure automatically switches to a new audit table and archives the contents of the current table. In addition, this step involves setting the **audit queue size** and the **suspend auditing when full** configuration parameters.	"Preparing to Manage the Audit Trail" on page 8-10. For single-table auditing, see Appendix A, "Single-Table Auditing." In addition, for information about threshold procedures and setting configuration parameters, see the *System Administration Guide*.
3. Set up the audit process to manage the *syslogs* transaction log in the *sybsecurity* database.	Determining how to handle the *syslogs* transaction log in the *sybsecurity* database. The task includes determining the setting of the **trunc log on chkpt** database option and establishing a last-chance threshold procedure for *syslogs* if **trunc log on chkpt** is off.	"Preparing to Manage the Transaction Log" on page 8-18. For information about thresholds and database options, see the *Systems Administration Guide*.

Table 8-1: General procedure for auditing

Action	Description	See
4. Set auditing options.	Using **sp_audit** to establish the events to be audited. **Note:** No audit records are generated until auditing is turned on with **sp_configure**.	"Setting Auditing Options" on page 8-20.
5. Enable auditing	Using **sp_configure** to turn on the **auditing** configuration parameter. Adaptive Server begins writing audit records for audited events to the current audit table.	"Enabling and Disabling Auditing" on page 8-21
6. Analyze the audit data	Using Adaptive Server to analyze the audit data.	"Querying the Audit Trail" on page 8-23 "Correlating Adaptive Server and Operating System Audit Records" on page 8-24 "Understanding the Audit Tables" on page 8-24

Installing the Audit System

The audit system is usually installed with **auditinit**, the Sybase installation program. Alternatively, you can perform a sequence of steps to install auditing without **auditinit**. For details, see "Installing Auditing with installsecurity" on page 8-8. Installation and **auditinit** are discussed in the Adaptive Server installation and configuration documentation for your platform.

When you install auditing, you can establish the number of system tables you want to use for the audit trail, the device for each audit system table, and the device for the *syslogs* transaction log.

Tables and Devices for the Audit Trail

You can specify up to eight system tables (*sysaudits_01* through *sysaudits_08*). Plan to use at least two tables for the audit trail. Put each table on its own device separate from the master device. If you do this, you can use a threshold procedure to automatically archive the current audit table before it fills up and switch to a new empty table for the subsequent audit records. (Although Sybase strongly recommends against it, you can, if necessary, set up auditing to run with only a single audit table. For instructions and warnings about doing this, see Appendix A, “Single-Table Auditing.”)

Device for the *syslogs* Transaction Log Table

When you install auditing, you must specify a separate device for the *syslogs* systems table, which contains the transaction log. The *syslogs* table, which exists in every database, contains a log of the transactions that are executed in the database.

Installing Auditing with *installsecurity*

The *$SYBASE/scripts* directory contains **installsecurity**, a script for installing auditing.

To use **installsecurity** to install auditing:

1. Create the auditing devices and auditing database with the Transact-SQL **disk init** and **create database** commands. For example:

```
disk init name = "auditdev",
    physname = "/dev/dsk/c2d0s4",
    vdevno = 3, size = 5120

disk init name = "auditlogdev",
    physname = “/dev/dsk/c2d0s5”,
    vdevno = 4, size = 1024

create database sybsecurity on auditdev
    log on auditlogdev
```

2. Use **isql** to execute the **installsecurity** script:

```
cd $SYBASE/scripts
setenv DSQUERY server_name
isql -Usa -Ppassword -Sserver_name < installsecurity
```

3. Shut down and restart Adaptive Server.

When you have completed these steps, the *sybsecurity* database has one audit table (*sysaudits_01*) created on its own segment. You can enable auditing at this time, but should add more auditing tables with the **sp_addaudittable** system procedure. For information about **sp_addaudittable**, see the *Adaptive Server Reference Manual*. For detailed information about **disk init** and **create database**, see the *System Administration Guide*.

Moving the Auditing Database to Multiple Devices

The *sybsecurity* database should be on its own device, separate from the *master* database. If you have more than one audit table, each table should also be on its own device. If you currently have *sybsecurity* on the same device as *master*, or if you want to move *sybsecurity* to another device for whatever reason, use one of the procedures described in the following sections. When you move the database, you can choose to save your existing global audit settings or not.

Moving *sybsecurity* Without Saving Global Audit Settings

To move the *sybsecurity* database without saving the global audit settings:

1. Drop the *sybsecurity* database.
2. Install *sybsecurity* again using the installation procedure described one of the following locations:
   - The configuration documentation for your platform
   - The section "Installing Auditing with installsecurity" on page 8-8.
3. During the installation process, be sure to place the *sybsecurity* database on one or more devices, separate from the master device.

Moving *sybsecurity* and Saving Global Audit Settings

To move the *sybsecurity* database and save the global audit settings:

1. Dump *sybsecurity* to a temporary table.

   ```
   dump sybsecurity to ttable
   ```

2. Drop *sybsecurity*.

   ```
   drop sybsecurity
   ```

3. Initialize the first device on which you want to place the *sybsecurity* database.

   ```
   diskinit (dev_1)
   ```

4. Initialize the device on which you want to place the security log.

   ```
   diskinit (logdev)
   ```

5. Create the new *sybsecurity* database.

   ```
   create database sybsecurity on dev_1 on logdev
   ```

6. Load the contents of the old *sybsecurity* database into the newly created database. The global audit settings will be preserved.

   ```
   load sybsecurity from ttable
   ```

7. Run the **online** command, which will upgrade *sysaudits* and *sysauditoptions* if necessary.

   ```
   online database sybsecurity
   ```

8. Load the auditing system procedures using the process described in configuration documentation for your platform.

To create more than one *sysaudits* table in *sybsecurity*:

1. Initialize the device on which you want to place the additional table.

   ```
   diskinit (dev_2)
   ```

2. Extend the *sybsecurity* database to the device you initialized in step 1.

   ```
   alter database sybsecurity on dev_2
   ```

3. Run the **sp_addaudittable** system procedure to create the next *sysaudits* table on the device you initialized in step 1.

   ```
   sp_addaudittable (sysaudits_02, dev_2)
   ```

4. To create more *sysaudits* tables on other devices, repeat steps 1–3 for each *sysaudits* table.

Preparing to Manage the Audit Trail

To effectively manage the audit trail:

1. Be sure that auditing is installed with two or more tables, each on a separate device. If not, consider adding additional audit tables and devices. See the Adaptive Server installation and configuration documentation. If you cannot spare two or more devices for auditing, see Appendix A, “Single-Table Auditing.”
2. Write a threshold procedure and attach it to each audit table segment.
3. Set configuration parameters for the audit queue size and what to do if the current audit table becomes full.

The following sections assume that you have installed auditing on two or more tables, each on a separate device. **auditinit** adds these devices to the audit database, creates one segment on each of the devices, and creates an audit table on each segment. If you have only one device for the audit tables, skip to Appendix A, "Single-Table Auditing."

Setting Up a Threshold Procedure

Before enabling auditing, plan to establish a threshold procedure to automatically switch auditing tables when the current table is full and to archive data from the current audit table. For detailed information about using threshold procedures, see the *System Administration Guide.*

This section describes:

- Creating a threshold procedure for handling the audit trail
- Attaching the procedure to all device segments used by the audit trail

Creating a Threshold Procedure to Handle the Audit Trail

The threshold procedure for the audit device segments should perform these tasks:

- Make the next empty audit table current using **sp_configure**.
- Archive the audit table that is almost full using the **insert** and **select** commands.

The following sections describe how to use Adaptive Server to accomplish the tasks.

Changing the Current Audit Table

The **current audit table** configuration parameter establishes the table where Adaptive Server writes audit rows. As a System Security Officer, you can change the current audit table with **sp_configure**, using the following syntax:

```
sp_configure “current audit table”, n
  [, "with truncate"]
```

where *n* is an integer that determines the new current audit table. The valid values for *n* are:

- 1 means *sysaudits_01*, 2 means *sysaudits_02*, and so forth.
- 0 is a special integer that tells Adaptive Server to automatically set the current audit table to the next table. For example, if your installation has three audit tables, *sysaudits_01*, *sysaudits_02*, and *sysaudits_03*, Adaptive Server sets the current audit table to:

- 2 if the current audit table is *sysaudits_01*
- 3 if the current audit table is *sysaudits_02*
- 1 if the current audit table is *sysaudits_03*

The **with truncate** option specifies that Adaptive Server should truncate the new table if it is not already empty. If this option is not specified and the table is not empty, the **sp_configure** command fails.

➤ ***Note***

If Adaptive Server truncates the current audit table and you have not archived the data, the table's audit records are lost. Be sure that the audit data is archived before using the **with truncate** option.

To execute **sp_configure** to change the current audit table, you must have the **sso_role** active. You can write a threshold procedure to automatically change the current audit table.

Archiving the Audit Table

You can use **insert** with **select** to copy the audit data into an existing table having the same columns as the audit tables in *sybsecurity*.

So that the threshold procedure can successfully copy data into the archive table in another database, perform these preparatory steps:

1. Create the archive database on a separate device from the one containing audit tables in *sybsecurity*.
2. Create an archive table with columns identical to those in the *sybsecurity* audit tables. If such a table does not already exist, you can use **select into** to create an empty one by having a false condition in the **where** clause. For example:

   ```
   use aud_db
   go
   select *
     into audit_data
     from sybsecurity.dbo.sysaudits_01
     where 1 = 2
   ```

   The **where** condition is always false, so an empty table is created that is a duplicate of *sysaudits_01*.

   The **select into/bulk copy** database option must be turned on in the archive database (using **sp_dboption**) before you can use **select into**.

The threshold procedure, after using **sp_configure** to change the audit table, can use **insert** and **select** to copy data to the archive table in the archive database. The procedure can execute commands similar to these:

```
insert aud_db.sso_user.audit_data
select * from sybsecurity.dbo.sysaudits_01
```

Example Threshold Procedure for Audit Segments

The following sample threshold procedure assumes that three tables are configured for auditing:

```
declare @audit_table_number int
/*
** Select the value of the current audit table
*/
select @audit_table_number = value
     from master.dbo.sysconfigures
     where name = "current audit table"
/*
** Set the next audit table to be current.
** When the next audit table is specified as 0,
** the value is automatically set to the next one.
*/
sp_configure "current audit table", 0, "with truncate"
/*
** Copy the audit records from the audit table
** that became full into another table.
*/
if @audit_table_number = 1
     insert aud_db.sso_user.audit_data
          select * from sysaudit_01
     truncate table sysaudit_01
else if @audit_table_number = 2
     insert aud_db.sso_user.audit_data
          select * from sysaudit_02
     truncate table sysaudit_02
else if @audit_table_number = 3
     insert aud_db.sso_user.audit_data
          select * from sysaudit_03
     truncate table sysaudit_03
return(0)
```

Attaching the Threshold Procedure to Each Audit Segment

To attach the threshold procedure to each audit table segment, use the system procedure **sp_addthreshold**.

Before executing **sp_addthreshold**, make sure to:

- Find out the number of audit tables configured for your installation and the names of their device segments
- Have the permissions and roles you need for the **sp_addthreshold** system procedure itself and the permissions and roles you need for all the commands contained in the threshold procedure

◆ ***WARNING!***

sp_addthreshold and sp_modifythreshold check to ensure that only a user with "sa_role" directly granted can add or modify a threshold. All system-defined roles active when you add or modify a threshold are inserted as valid roles for your login in the *systhresholds* table. However, only directly granted roles are activated when the threshold procedure fires.

Audit Tables and Their Segments

When you install auditing, **auditinit** displays the name of each audit table and its segment. The segment names are “aud_seg1” for *sysaudits_01*, “aud_seg2” for *sysaudits_02*, and so forth to “aud_seg8” for *sysaudits_08*. You can find information about the segments in the *sybsecurity* database if you execute **sp_helpsegment** with *sybsecurity* as your current database. One way to find the number of audit tables for your installation is to execute the following SQL commands:

```
use sybsecurity
go
select count(*) from sysobjects
  where name like "sysaudit%"
go
```

In addition, you can get information about the audit tables and the *sybsecurity* database by executing the following SQL commands:

```
sp_helpdb sybsecurity
go
use sybsecurity
go
sp_help sysaudits_01
go
sp_help sysaudits_02
go
  ...
```

Required Roles and Permissions

To execute **sp_addthreshold**, you must be either the database owner or a System Administrator. A System Security Officer should be the owner of the *sybsecurity* database and, therefore, should be able to execute **sp_addthreshold**. In addition to being able to execute **sp_addthreshold**, you must have permission to execute all the commands in your threshold procedure. For example, to execute **sp_configure** for **current audit table**, the **sso_role** must be active. When the threshold procedure fires, Adaptive Server attempts to turn on all the roles and permissions that were in effect when you executed **sp_addthreshold**.

Using sp_addthreshold

The syntax for **sp_addthreshold** is:

```
sp_addthreshold database, segment, free_pages,
       procedure
```

where:

database is the database for which to add the threshold. This must be the name of the current database.

segment is the segment for which to monitor free space. Use quotes when specifying the "default" segment.

free_pages is the number of free pages at which the threshold is crossed. When free space in the segment falls below this level, Adaptive Server executes the associated stored procedure.

procedure is the stored procedure that executes when the amount of free space on *segment* drops below *free_pages*. The procedure can be located in any database on the current Adaptive Server or another Server, or an Open Server application.

You can use the following sequence of commands to attach the threshold procedure **audit_thresh** to three device segments:

```
use sybsecurity
go
sp_addthreshold sybsecurity, aud_seg1, 250,
    audit_thresh
sp_addthreshold sybsecurity, aud_seg2, 250,
    audit_thresh
sp_addthreshold sybsecurity, aud_seg3, 250,
    audit_thresh
go
```

The sample threshold procedure **audit_thresh** receives control when fewer than 250 free pages remain in the current audit table.

For more information about adding threshold procedures, see the *System Administration Guide.*

Auditing with the Sample Threshold Procedure in Place

With the sample threshold procedure **audit_thresh**, auditing can proceed without manual intervention. For information see "Example Threshold Procedure for Audit Segments" on page 8-13.

After you enable auditing, Adaptive Server writes all audit data to the initial current audit table, *sysaudits_01*. When *sysaudits_01* is within 250 pages of being full, the threshold procedure **audit_thresh** fires. The procedure switches the current audit table to *sysaudits_02*, and, immediately, Adaptive Server starts writing new audit records to *sysaudits_02*. The procedure also copies all audit data from *sysaudits_01* to the *audit_data* archive table in the *audit_db* database. The rotation of the audit tables continues in this fashion without manual intervention. For information on enabling auditing, see "Enabling and Disabling Auditing" on page 8-21.

Setting Auditing Configuration Parameters

Set the following configuration parameters for your auditing installation:

- **audit queue size** sets the number of records in the audit queue in memory.
- **suspend auditing when full** determines what Adaptive Server does if the current audit table becomes completely full. The full condition occurs only if the threshold procedure attached to the current table segment is not functioning properly.

Setting the Size of the Audit Queue

The in-memory audit queue holds audit records generated by user processes until the records can be processed and written to the audit trail. A System Security Officer can change the size of the audit queue with the **audit queue size** configuration parameter. Keep in mind the performance / risk trade-offs when you configure the queue size.

If the queue is too large, records can remain in it for a long time. While records are in the queue, they are at risk of being lost if the system crashes. However, if the queue is too small it can repeatedly become full, which affects overall system performance. User processes that generate audit records sleep if the audit queue is full. When you set the queue size, consider how much auditing you plan to do and the storage that is available.

The memory requirement for a single audit record is 424 bytes. The default size for the audit queue is 100 records, which requires approximately 42K.

To set the size of the audit queue, use **sp_configure**. The syntax is:

```
sp_configure "audit queue size", [value]
```

value is the number of records that the audit queue can hold. The minimum value you can specify is 1, and the maximum is 65,535. For example, to set the audit queue size to 300, execute:

```
sp_configure "audit queue size", 300
```

For more information about setting the audit queue size and other configuration parameters, see the *System Administration Guide*.

Setting the Threshold for Auditing Devices

If you have two or more audit tables, each on a separate device other than the master device, and have a threshold procedure for each audit table segment, the audit devices should never become full. Only if a threshold procedure is not functioning properly would the “full” condition occur. You can use **sp_configure** to set the **suspend auditing when full** parameter to determine what happens if the devices do become full. Choose one of these options:

- Suspend the auditing process and all user processes that cause an auditable event. Resume normal operation after a System Security Officer clears the current audit table.
- Truncate the next audit table and start using it. This allows normal operation to proceed without intervention from a System Security Officer.

To set this configuration parameter, use **sp_configure**. You must have the **sso_role** active. The syntax is:

```
sp_configure "suspend auditing when full",
    [0|1]
```

0 truncates the next audit table and starts using it as the current audit table whenever the current audit table becomes full. If you set the parameter to 0, you ensure that the audit process is never suspended. However, you incur the risk that older audit records will get lost if they have not been archived.

1 suspends the audit process and all user processes that cause an auditable event. To resume normal operation, the System Security Officer must log in and set up an empty table as the current audit table. During this period, the System Security Officer is exempt from normal auditing. If the System Security Officer's actions would generate audit records under normal operation, Adaptive Server sends an error message and information about the event to the error log.

The default for **suspend auditing when full** is **1**, but you can use **sp_configure** to change it. If you have a threshold procedure attached to the audit table segments, set **suspend auditing when full** to **1** (on). If it is set to **0** (off), Adaptive Server may truncate the audit table that is full before your threshold procedure has a chance to archive your audit records.

Preparing to Manage the Transaction Log

When you install auditing, you must specify a separate device for the transaction log in the *sybsecurity* database. The transaction log is the same as the *syslogs* system table, which exists in every database. Adaptive Server always places the transaction log on a segment called *syslogs*.

How you manage the transaction log in *sybsecurity* depends upon the setting you choose for the **trunc log on chkpt** database option. If this option is active, Adaptive Server truncates *syslogs* every time it performs an automatic **checkpoint**. After auditing is installed, the value of **trunc log on chkpt** is **on**, but you can use **sp_dboption** to change its value. The following sections describe guidelines for managing the transaction log, depending upon how **trunc log on chkpt** is set.

Managing the Transaction Log With Truncation

If you enable the **trunc log on chkpt** option for the *sybsecurity* database, you do not need to worry about the transaction log becoming full. Adaptive Server truncates the log whenever it performs a checkpoint. With this option on, you cannot dump transactions, but you can dump the database.

If you follow the procedures discussed in "Setting Up a Threshold Procedure" on page 8-11, audit tables are automatically archived to tables in another database. You can use standard backup and recovery procedures for this archive database. That is, you can regularly issue **dump database** and **dump transaction.** Recovery involves reloading the database and its transactions in the proper order.

If a crash occurs on the *sybsecurity* device, you can reload the database and resume auditing. At most, only the records in the in-memory audit queue and the current audit table are lost because the archive database contains all other audit data. After you reload the database, use **sp_configure with truncate** to set and truncate the current audit table.

If you have not changed server-wide auditing options since you dumped the database, all auditing options stored in *sysauditoptions* are automatically restored when you reload *sybsecurity*. If not, you can run a script to set the options as you desire prior to resuming auditing.

For more information about backup and recovery, see the *System Administration Guide*.

Managing the Transaction Log With No Truncation

If you use **db_option** to turn the **trunc log on chkpt** off, the transaction log may fill up. Plan to attach a **last-chance threshold procedure** to the transaction log segment. This procedure gets control when the amount of space remaining on the segment is less than a threshold amount computed automatically by Adaptive Server. The threshold amount is an estimate of the number of free log pages that would be required to back up the transaction log.

The default name of the last-chance threshold procedure is **sp_thresholdaction**, but you can specify a different name with **sp_modifythreshold**.

➤ ***Note***

sp_modifythreshold checks to ensure you have “sa_role” active. See “Attaching the Threshold Procedure to Each Audit Segment” on page 8-13 for more information.

Adaptive Server does not supply a default procedure, but the *System Administration Guide* contains examples of last-chance threshold procedures. The procedure should execute the **dump transaction** command, which truncates the log. When the transaction log reaches the last-chance threshold point, any transaction that is running is suspended until space is available. The suspension occurs because the option **abort xact when log is full** is always set to FALSE for the *sybsecurity* database. You cannot change this option.

With the **trunc log on chkpt** option off, you can use standard backup and recovery procedures for the *sybsecurity* database, but be aware that the audit tables in the restored database may not be in sync with their status at the time of a device failure. For more information about using last-chance threshold procedures, other threshold procedures, and backup and recovery procedures, see the *System Administration Guide*.

Setting Auditing Options

After you have installed auditing, you can set auditing options. The syntax for **sp_audit** is:

```
sp_audit option, login_name, object_name [,setting]
```

If you run **sp_audit** with no parameters, it provides a complete list of the options. For details about **sp_audit**, see the *Adaptive Server Reference Manual*.

➤ ***Note***

No auditing occurs until you use **sp_configure** to activate auditing for the server. For information on how to start auditing, see “Enabling and Disabling Auditing” on page 8-21.

Examples of Setting Auditing Options

Suppose you want to audit:

- All security-relevant global events that fail permission checks. Use the global **security** option:

  ```
  sp_audit “security”, “all”, “all”, “fail”
  ```

- All failed deletes on the *projects* table in the *company_operations* database and for all new tables in the database. Use the object-specific **delete** option for the *projects* table and use **default table** for all future tables in the database. To set object-specific auditing options, you must be in the object’s database before you execute **sp_audit**. For this example, execute:

```
use company_operations
go
sp_audit "delete", "all", "projects", "fail"
go
sp_audit "delete", "all", "default table",
"fail"
go
```

- All actions of a user with the **sa_role** active. Use the global **all** option:

  ```
  sp_audit "all", "sa_role", "all", "on"
  ```

- All successful executions of **create database**. Use the database-specific **create** option for the *master* database:

  ```
  sp_audit "create", "all", "master", "pass"
  ```

- All failed object creations in the *personnel* database. Use the database-specific **create** option for the *personnel* database:

  ```
  sp_audit "create", "all", "personnel", "fail"
  ```

Determining Current Auditing Settings

To determine the current auditing settings for a given option, use **sp_displayaudit**. The syntax is:

```
sp_displayaudit [procedure | object | login |
   database | global | default_object |
   default_procedure [, name]]
```

For more information, see **sp_displayaudit** in the *Adaptive Server Reference Manual*.

Enabling and Disabling Auditing

To enable or disable auditing, use **sp_configure** with the **auditing** configuration parameter. The syntax is:

```
sp_configure "auditing", [0 | 1 ]
```

1 enables auditing. **0** disables auditing. For example, if you want to enable auditing, enter:

```
sp_configure "auditing", 1
```

➤ ***Note***

When you enable or disable auditing, Adaptive Server automatically generates an audit record. See event codes 73 and 74 in Table 8-4 on page 8-26.

Adding User-Specified Records to the Audit Trail

sp_addauditrecord allows users to enter user-defined audit records (comments) into the audit trail. The syntax is:

```
sp_addauditrecord [text] [, db_name] [, obj name]
   [, owner_name] [, dbid] [, objid]
```

All the parameters are optional.

- *text* is the text of the message that you want to add to the current audit table. The text is inserted into the *extrainfo* column of the current audit table.
- *db_name* is the name of the database referred to in the record. This is inserted into the *dbname* column of the current audit table.
- *obj_name* is the name of the object referred to in the record. This is inserted into the *objname* column of the current audit table.
- *owner_name* is the owner of the object referred to in the record. This is inserted into the *objowner* column of the current audit table.
- *dbid* is the database ID number of *db_name*. This is an integer value, and must not be placed in quotes. *dbid* is inserted into the *dbid* column of the current audit table.
- *objid* is the object ID number of *obj_name*. This is an integer value, and must not be placed in quotes. *objid* is inserted into the *objid* column of the current audit table.

You can use **sp_addauditrecord** if:

- You have execute permission on **sp_addauditrecord**.
- The auditing configuration parameter was activated with **sp_configure**.
- The **adhoc** auditing option was enabled with **sp_audit**.

When the audit system is installed, only a System Security Officer and the Database Owner of *sybsecurity* can use **sp_addauditrecord**. Permission to execute it may be granted to other users.

Examples of Adding User-Defined Audit Records

The following example adds a record to the current audit table. The text portion is entered into the *extrainfo* column of the current audit table, “corporate” into the *dbname* column, “payroll” into the *objname* column, “dbo” into the *objowner* column, “10” into the *dbid* column, and “1004738270” into the *objid* column:

```
sp_addauditrecord "I gave A. Smith permission to
view the payroll table in the corporate database.
This permission was in effect from 3:10 to 3:30 pm
on 9/22/92.", "corporate", "payroll", "dbo", 10,
1004738270
```

The following example inserts information only into the *extrainfo* and *dbname* columns of the current audit table:

```
sp_addauditrecord @text="I am disabling auditing
briefly while we reconfigure the system",
@db_name="corporate"
```

Querying the Audit Trail

To query the audit trail, use SQL to select and summarize the audit data. If you follow the procedures discussed in “Preparing to Manage the Audit Trail” on page 8-10, the audit data in *sysaudits_01* through *sysaudits_08* is automatically archived to one or more tables in another database. For example, assume that the audit data resides in a table called *audit_data* in the *audit_db* database. Then, to select audit records for tasks performed by “bob” on July 5, 1993, execute:

```
use audit_db
go
select * from audit_data
  where loginname = "bob"
  and eventtime like "Jul 5% 93"
go
```

This command requests audit records for commands performed in the *pubs2* database by users with the System Security Officer role active:

```
select * from audit_data
  where extrainfo like “%sso_role%
  and dbname = "pubs2"
go
```

This command requests audit records for all table truncations (event 64):

```
select * from audit_data
  where event = 64
go
```

Correlating Adaptive Server and Operating System Audit Records

The easiest way to link Adaptive Server audit records with operating system records is to make Adaptive Server login names the same as operating system login names. For instance, if a user's operating system login name is "thrunobulax," that user's Adaptive Server login name should also be "thrunobulax."

Alternatively, the System Security Officer can maintain a record mapping users' operating system login names to their Adaptive Server login names. However, this approach requires ongoing maintenance, as login names for new users have to be recorded by hand.

Understanding the Audit Tables

Table 8-2 describes the columns in all audit tables:

Table 8-2: Columns in each audit table

Column Name	Datatype	Description
event	*smallint*	Type of event being audited. For more information, see Table 8-4 on page 8-26.
eventmod	*smallint*	More information about the event being audited. Possible values are: 0 = no modifier for this event 1 = the event passed permission checking 2 = the event failed permission checking
spid	*smallint*	Server process ID of the process that caused the audit record to be written.
eventtime	*datetime*	Date and time that the audited event occurred.
sequence	*smallint*	Sequence number of the record within a single event. Some events require more than one audit record.
suid	*smallint*	Server login ID of the user who performed the audited event.
dbid	*int null*	Database ID in which the audited event occurred, orin which the object, stored procedure, or trigger resides, depending on the type of event.

Table 8-2: Columns in each audit table (continued)

Column Name	Datatype	Description
objid	*int null*	ID of the accessed object, stored procedure, or trigger.
xactid	*binary(6) null*	ID of the transaction containing the audited event. For a multi-database transaction, this is the transaction ID from the database where the transaction originated.
loginname	*varchar(30) null*	Login name corresponding to the *suid*.
dbname	*varchar(30) null*	Database name corresponding to the *dbid*.
objname	*varchar(30) null*	Object name corresponding to the *objid*.
objowner	*varchar(30) null*	Name of the owner of *objid*.
extrainfo	*varchar(255) null*	Additional information about the audited event. This column contains a sequence of items separated by semicolons. For details, see "Reading the extrainfo Column" on page 8-25.

Reading the *extrainfo* Column

The *extrainfo* column contains a sequence of data separated by semicolons. The data is organized in the following catagories.

Table 8-3: Information in the extrainfo column

Position	Category	Description
1	Roles	A list of active roles, separated by blanks.
2	Keywords or Options	The name of the keyword or option that was used for the event. For example, for the **alter table** command, the **add column** or **drop constraint** options might have been used. If multiple keywords or options are listed, they are separated by commas.
3	Previous value	If the event resulted in the update of a value, this item contains the value prior to the update.
4	Current value	If the event resulted in the update of a value, this item contains the new value.
5	Other information	Additional security-relevant information that is recorded for the event.
6	Proxy information	The original login name if the event occurred while a **set proxy** was in effect.

Table 8-3: Information in the extrainfo column

Position	Category	Description
7	Principal name	The principal name from the underlying security mechanism if the user's login is the secure default login, and the user logged into Adaptive Server via unified login. The value of this item is NULL if the secure default login is not being used.

The following example shows an *extrainfo* column entry for the event of changing an auditing configuration parameter.

```
sso_role;suspend auditing when full;1;0;;ralph;
```

This entry indicates that a System Security Officer changed the configuration parameter **suspend auditing when full** from **1** to **0**. There is no "other information" for this entry. The sixth category indicates that the user "ralph" was operating with a proxy login. No principal name is provided.

The other fields in the audit record give other pertinent information. For example, the record contains the server user id (*suid*) and the login name (*loginname*).

Table 8-4 lists the values that appear in the *event* column, arranged by **sp_audit** option. The "Information in extrainfo" column in this table describes information that might appear in the *extrainfo* column of an audit table, based on the seven categories described in Table 8-3.

Table 8-4: Values in event and extrainfo columns

Audit Option	Command or Access To Be Audited	*event*	Information in *extrainfo*
(Automatically audited event – not controlled by an option)	Enabling auditing with: **sp_configure auditing**	73	
(Automatically audited event – not controlled by an option)	Disabling auditing with: **sp_configure auditing**	74	

Table 8-4: Values in event and extrainfo columns (continued)

Audit Option	Command or Access To Be Audited	*event*	Information in *extrainfo*
adhoc	User-defined audit record	1	*extrainfo* is filled by the *text* parameter of **sp_addauditrecord**
alter	**alter database**	2	**Keywords or Options:** **alter maxhold** **alter size**
	alter table	3	**Keywords or Options:** **add column** **drop column** **replace column** **add constraint** **drop constraint**
bcp	**bcp in**	4	
bind	**sp_bindefault**	6	**Other information:** Name of the default
	sp_bindmsg	7	**Other information:** Message ID
	sp_bindrule	8	**Other information:** Name of the rule
create	**create database**	9	
	create table	10	
	create procedure	11	
	create trigger	12	
	create rule	13	
	create default	14	
	sp_addmessage	15	**Other information:** Message number
	create view	16	
dbaccess	Any access to the database by any user	17	**Keywords or Options:** **use cmd** **outside reference**
dbcc	**dbcc**	81	**Keywords or Options:** Any of the **dbcc** keywords such as **checkstorage** and the options for that keyword.
delete	**delete** from a table	18	**Keywords or Options:** **delete**
	delete from a view	19	**Keywords or Options:** **delete**

Table 8-4: Values in event and extrainfo columns (continued)

Audit Option	Command or Access To Be Audited	*event*	Information in *extrainfo*
disk	disk init	20	**Keywords or Options:** disk init **Other Information:** Name of the disk
	disk refit	21	**Keywords or Options:** disk refit **Other Information:** Name of the disk
	disk reinit	22	**Keywords or Options:** disk reinit **Other Information:** Name of the disk
	disk mirror	23	**Keywords or Options:** disk mirror **Other Information:** Name of the disk
	disk unmirror	24	**Keywords or Options:** disk unmirror **Other Information:** Name of the disk
	disk remirror	25	**Keywords or Options:** disk remirror **Other Information:** Name of the disk
drop	drop database	26	
	drop table	27	
	drop procedure	28	
	drop trigger	29	
	drop rule	30	
	drop default	31	
	sp_dropmessage	32	**Other Information:** Message number
	drop view	33	
dump	dump database	34	
	dump transaction	35	
errors	Fatal error	36	**Other information:** *Error number.Severity.State*
	Non-fatal error	37	**Other information:** *Error number.Severity.State*
exec_procedure	Execution of a procedure	38	**Other Information:** All input parameters

Table 8-4: Values in event and extrainfo columns (continued)

Audit Option	Command or Access To Be Audited	*event*	Information in *extrainfo*
exec_trigger	Execution of a trigger	39	
func_obj_access, func_dbaccess	Accesses to objects and databases via Transact-SQL functions	85	
grant	**grant**	40	
insert	**insert** into a table	41	**Keywords or Options:** If **insert** is used: **insert** If **select into** is used: **insert into** followed by the fully qualified object name
	insert into a view	42	**Keywords or Options**: **insert**
load	**load database**	43	
	load transaction	44	
login	Any login to the server	45	**Other Information:** Host name of the machine from which login was done
logout	Any logouts from the server	46	**Other Information:** Host name of the machine from which login was done
reference	Creation of references to tables	91	**Keywords or Options**: **reference** **Other Information:** Name of the referencing table
revoke	**revoke**	47	
rpc	Remote procedure call from another server	48	**Keywords or Options:** Name of client program **Other Information:** Server name, host name of the machine from which the RPC was done.
	Remote procedure call to another server	49	**Keywords or Options:** Procedure name

Table 8-4: Values in event and extrainfo columns (continued)

Audit Option	Command or Access To Be Audited	*event*	Information in *extrainfo*
security	Server boot	50	**Other Information:** **-d***masterdevicename* **-i***interfaces file path* **-S***servername* **-e***errorfilename*
	Server shutdown	51	**Keywords or Options:** **shutdown**
	Role toggling	55	**Previous Value: on** or **off** **Current Value: on** or **off** **Other Information:** Name of the role being set
	sp_configure	82	**Other Information:** • If a parameter is being set: number of configuration parameter • If a configuration file is being used to set parameters: name of the configuration file
	online database	83	
	Regeneration of a password by an SSO	76	**Keywords or Options:** Setting SSO password **Other information:** Login name
	Execution of the **kill** command (CIS only)	89	**Keywords or Options:** **kill**
	Execution of **connect to** command (CIS only)	90	**Keywords or Options:** **connect to**
	Execution of the **proc_role** function within a system procedure	80	**Other Information:** Required roles
	Execution of **valid_user** function	85	**Keywords or Options:** **valid_user**
	Execution of **set proxy** or **set session authorization**	88	**Previous value:** Previous *suid* **Current value:** New *suid*

Table 8-4: Values in event and extrainfo columns (continued)

Audit Option	Command or Access To Be Audited	*event*	Information in *extrainfo*
select	**select** from a table	62	**Keywords or Options:** **select into** **select** **readtext**
	select from a view	63	**Keywords or Options:** **select into** **select** **readtext**
setuser	**setuser**	84	**Other Information:** Name of the user being set
table_access	**select**	62	**Keywords or Options:** **select into** **select** **readtext**
	delete	18	**Keywords or Options:** **delete**
	update	70	**Keywords or Options:** **update** **writetext**
	insert	41	
truncate	**truncate table**	64	
unbind	**sp_unbindefault**	67	
	sp_unbindrule	68	
	sp_unbindmsg	69	
update	**update** to a table	70	**Keywords or Options:** **update** **writetext**
	update to a view	71	**Keywords or Options:** **update** **writetext**

Table 8-4: Values in event and extrainfo columns (continued)

Audit Option	Command or Access To Be Audited	*event*	Information in *extrainfo*
view_access	select	63	**Keywords or Options:** select into select readtext
	delete	19	**Keywords or Options:** delete
	insert	42	**Keywords or Options:** insert
	update	71	**Keywords or Options:** update writetext

The system audit tables can be accessed only by a System Security Officer, who can read the tables by executing SQL commands. The only commands that are allowed on the system audit tables are **select** and **truncate**. If you follow the procedures outlined in “Preparing to Manage the Audit Trail” on page 8-10, the audit data is in audit tables in an archive database.

A Single-Table Auditing

This appendix describes the process of auditing with a single audit table. Topics include:

- Requirements for Managing a Single-Device Audit Trail A-1
- Establishing and Managing Single-Table Auditing A-3
- Threshold Procedure for Single-Table Auditing A-4
- What Happens When the Current Audit Table Is Full? A-5
- Recovering When the Current Audit Table Is Full A-6

Requirements for Managing a Single-Device Audit Trail

Sybase strongly recommends that you **not** use single-device auditing for production systems. If you use only a single audit table, you create a window of time while you are archiving audit data and truncating the audit table during which incoming audit records will be lost. There is no way to avoid this when using only a single audit table.

With only a single audit table, you are much more likely to encounter the situation where your audit table fills up. The consequences of having the audit table fill up depend on how you have set the configuration parameter, **suspend auditing when full**. If you have **suspend auditing when full on**, the audit process is suspended, as are all user processes that cause auditable events. If **suspend auditing when full** is off, the audit table is truncated, and you lose all the audit records that were in the audit table.

For **non-production** systems, where the loss of a small number of audit records may be acceptable, you can use a single table for auditing, if you cannot spare the additional disk space for multiple audit tables, or you do not have additional devices to use.

The procedure for using a single audit table is similar to using multiple audit tables, with these exceptions:

- During your installation session, you specify only one system table to use for auditing.
- During your installation session, you specify only one device for the audit system table.

- The threshold procedure you create for archiving audit records is different from the one you would create if you were using multiple audit tables.

Figure A-1 shows how the auditing process works with a single audit table.

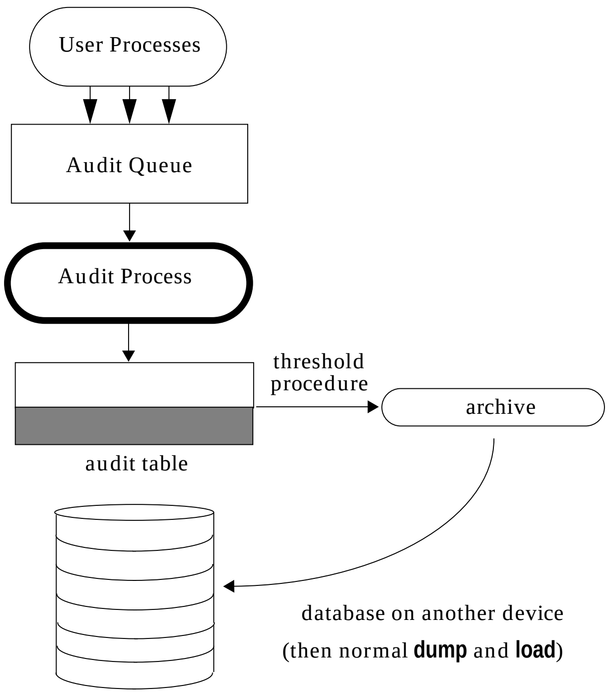

Figure A-1: Auditing with a single audit table

Establishing and Managing Single-Table Auditing

Table A-1 provides an overview of managing single-table auditing.

Table A-1: Auditing process for single-table auditing

Action	Description	See
1. Install auditing.	Installation of auditing, which involves setting the number of audit tables and assigning devices for the audit trail and the *syslogs* transaction log in the *sybsecurity* database.	"Requirements for Managing a Single-Device Audit Trail" on page A-1 and the installation documentation for your platform
2. Set up the audit process to manage the audit trail.	Writing and establishing a threshold procedure that receives control when the audit table is nearly full. The procedure automatically writes the contents of the audit table to another table, and then truncates the audit table. In addition, this step involves setting the **audit queue size** and **suspend auditing when full** configuration parameters.	"Requirements for Managing a Single-Device Audit Trail" on page A-1 and "Establishing and Managing Single-Table Auditing" on page A-3. "Threshold Procedure for Single-Table Auditing" on page A-4. In addition, for information about threshold procedures and setting configuration parameters, see the *System Administration Guide*.
3. Set up the audit process to manage the *syslogs* transaction log in the *sybsecurity* database.	Determining how to handle the *syslogs* transaction log in the *sybsecurity* database. The task includes determining the setting of the **trunc log on chkpt** database option and establishing a last-chance threshold procedure for *syslogs* if **trunc log on chkpt** is off.	"Preparing to Manage the Transaction Log" on page 8-18. For information about thresholds and database options, see the *System Administration Guide*.
4. Set auditing options.	Using **sp_audit** to establish the events to be audited. **Note:** No audit records are generated until auditing is turned on with **sp_configure**.	"Setting Auditing Options" on page 8-20

Table A-1: Auditing process for single-table auditing (continued)

Action	Description	See
5. Enable auditing.	Using **sp_configure** to turn on the **auditing** configuration parameter. Adaptive Server begins writing audit records for audited events to the current audit table.	"Enabling and Disabling Auditing" on page 8-21
6. Analyze the audit data.	Using Adaptive Server to analyze the audit data.	"Querying the Audit Trail" on page 8-23 "Correlating Adaptive Server and Operating System Audit Records" on page 8-24 "Understanding the Audit Tables" on page 8-24

Threshold Procedure for Single-Table Auditing

For single-table auditing, the threshold procedure needs to perform these tasks:

- Archive the almost-full audit table to another table, using the **insert** and **select** commands.
- Truncate the audit table to create space for new audit records, using the **truncate table** command.

A description of how to use Adaptive Server to accomplish these tasks follows.

Archiving the Audit Table

Before you can archive your audit records you need to create an archive table that has the same columns as your audit table, into which you will put the archived records. After you have done this, your threshold procedure can use **insert** with **select** to copy the audit records into the archive table.

Sample Threshold Procedure

Following is a sample threshold procedure for use with a single audit table:

```
create procedure audit_thresh as
/*
** copy the audit records from the audit table to
** the archive table
*/
insert aud_db.sso_user.audit_data
     select * from sysaudits_01
return(0)
go
/*
** truncate the audit table to make room for new
** audit records
*/
truncate table "sysaudits_01"
go
```

After you have created your threshold procedure, you will need to attach the procedure to the audit table segment. For instructions, see "Attaching the Threshold Procedure to Each Audit Segment" on page 8-13.

◆ ***WARNING!***

On a multiprocessor, it is possible for the audit table to fill up, even if you have a threshold procedure that gets triggered before the audit table is full. For example, if the threshold procedure is running on a heavily loaded CPU, and a user process performing auditable events is running on a less heavily loaded CPU, it is possible that enough auditable events could occur to fill up the audit table before the threshold procedure had a chance to trigger. The configuration parameter suspend auditing when full determines what happens when the audit table fills up. For information about setting this parameter, see "Establishing and Managing Single-Table Auditing" on page A-3.

What Happens When the Current Audit Table Is Full?

If the current audit table is full, the following occurs:

1. The audit process attempts to insert the next audit record into the current audit table. This fails, so the audit process terminates. An error message goes to the error log.
2. When a user attempts to perform an auditable event (an event on which auditing has been enabled), the event cannot be completed because auditing cannot proceed. The user process

terminates. Users who do not attempt to perform an auditable event are unaffected.

3. If you have login auditing enabled, no one can log into the server except a System Security Officer.
4. If you are auditing commands executed with the **sso_role** active, the System Security Officer will be unable to execute commands,

Recovering When the Current Audit Table Is Full

If the current audit device becomes full and the audit queue is also full, the System Security Officer becomes exempt from auditing. Every auditable event performed by a System Security Officer after this point causes a warning message to be sent to the error log file. The message states the date and time and a warning that an audit has been missed, as well as the login name, *event* code, and other information that would normally be stored in the *extrainfo* field of the audit table.

When the current audit table is full, the System Security Officer can archive and truncate the audit table as described in “Requirements for Managing a Single-Device Audit Trail” on page A-1. A System Administrator can execute **shutdown** to stop the server and then restart it to reestablish auditing.

If the audit system terminates abnormally, the System Security Officer can execute the **shutdown** command to shut down the server after the current audit table has been archived and truncated. Normally, only the System Administrator can execute **shutdown**.

Index

Page numbers in **bold** are primary references.

Symbols

* (asterisk)
  select and 5-25
{} (curly braces)
  in SQL statements xxii
... (ellipsis) in SQL statements xxii
" " (quotation marks)
  enclosing punctuation 3-4
  enclosing values in 3-3
[ ] (square brackets)
  in SQL statements xxii

A

Access
  restricting guest users 3-8
Access permissions. *See* Object access permissions
Access protection. *See* Permissions; Security functions
Accounting, chargeback 3-23 to 3-25
Accounts, Server. *See* Logins; Users
Activating roles 4-18
Adding
  comments to the audit trail 8-6
  group to a database 3-5 to 3-6
  guest users 3-7
  logins for System Administrators and System Security Officers 2-3
  logins to Server 3-3 to 3-5
  remote logins 3-9, 7-8 to 7-10
  remote servers 7-3 to 7-15
  users to a database 3-1 to 3-10
  users to a group 3-6
Administering security, getting started 2-1 to 2-6
Aliases
  server 7-4
Aliases, user
  See also Logins; Users
  creating 3-17
  database ownership transfer and 5-8
  dropping 3-19
  help on 3-19
  suid (server user ID) and 3-17
all keyword
  grant 5-11, 5-16
  revoke 5-16
allow remote access configuration parameter 7-13
Alternate identity. *See* Alias, user
alter role command **4-11**
ansi_permissions option, **set**
  permissions and 5-14
Assigning
  login names 2-2
Asterisk (*)
  select and 5-25
@@servername global variable 7-4
Auditing 1-6, **8-1** to **8-24**
  See also Audit Options
  adding comments to the audit trail 8-6
  aliases and 3-17
  configuration parameters 8-5
  devices for 8-7
  disabling 8-6
  displaying options for 8-6
  enabling 2-2, 8-6
  enabling and disabling 8-21

establishing after installation 2-3
installing 8-7
managing the audit trail 8-10
managing the transaction log 8-18
overall process 8-6
overview 8-1
queue, size of 8-4, 8-16
sybsecurity database 8-2
sysaudits_01...sysaudits_08 tables 8-24
system procedures for 8-5
threshold procedure for 8-11
auditing configuration parameter 8-21
Audit options
displaying 8-6
setting 8-20
Audit queue 8-4, 8-16
audit queue size configuration parameter 8-5, 8-16
Audit trail 8-1, 8-24 to 8-32
adding comments 8-6, 8-22 to 8-23
changing current audit table 8-11
illustration with multiple audit tables 8-4
managing 8-10
querying 8-23
system audit tables 8-2
threshold procedure for 8-11
Authorizations. *See* Permissions

B

Base tables. *See* Tables
Binary expressions xxiii

C

Calls, remote procedure **7-1** to **7-15**
timeouts 7-5
cascade option, **revoke** 5-13
Case sensitivity
in SQL xxii
Chains, ownership **5-26**
Changing
configuration parameters 7-13
Database Owners 5-8
default database 3-15
passwords for login accounts 3-14
Server logins 3-15
user's group 3-16
user's identity 5-7
user information 3-13 to 3-17
Character expressions xxiii
Chargeback accounting 3-23 to 3-25
Columns
permissions on 5-12, 5-22
Comments
adding to audit trail 8-6, 8-22 to 8-23
Configuration parameters
audit-related 8-5
changing 7-13
chargeback accounting 3-24
remote logins and 7-13 to 7-15
Conflicting permissions 5-18 to 5-19
See also Permissions
Constants xxiii
Context-sensitive protection 5-25
Conventions
Transact-SQL syntax xxi to xxiii
used in manuals xx
cpu accounting flush interval configuration parameter 3-24
CPU usage
per user 3-23
create database command
permission to use 5-3
create role command **4-10**
create trigger command 5-14
Creating
databases 5-3
groups 3-5 to 3-6
guest users 3-7
sybsecurity database 8-8
triggers 5-14
user aliases 3-17
user-defined roles 4-10
Curly braces ({})
in SQL statements xxii

current audit table configuration parameter 8-11
Current usage statistics 3-24
Current user
  set proxy and 6-5

D

DAC. *See* Discretionary access control (DAC)
Data
  See also Permissions
  packets 7-14
Database object owners **4-7**
  See also Database Owners
  permissions 4-7, 5-2, 6-1
  status not transferable 3-11
  tasks of 4-7
Database objects
  See also individual object names
  access permissions for 4-7, 5-11
  creating 5-9
  dependent 5-27
  dropping 5-9
  dropping users who own 3-11
  ownership 3-11, 4-7, 5-9
  permissions on 5-9
  triggers on 5-31
Database Owners **4-6**
  See also Database object owners; Permissions
  changing 5-8
  login name 4-6
  name inside database 3-10, 3-18
  objects not transferred between 3-11
  password forgotten by 5-4
  permissions granted by 5-16
  permissions of 4-6, 5-1 to 5-2, 5-5
  setuser command and 5-7
  several users as same 3-17
  tasks of 4-6
Databases
  See also Database objects; User databases
  adding users 3-6 to 3-10
  auditing 8-8
  changing user's default 3-4
  creation permission 5-3
  default 3-3, 3-4, 3-15
  dropping users from 3-10 to 3-11
  ownership of 5-3
  user information 3-21
"dbo" user name 4-6
Deactivating roles 4-18
Default database
  changing user's 3-15
Defaults
  See also Database objects
Default settings
  databases 3-3, 3-4
  language 3-4
  permissions 5-2
Denying access to a user 3-11, 3-13
Devices
  audit system 8-7
Disabling auditing 8-6
Discretionary access control (DAC) **5-1** to **5-31**
  See also Permissions
  granting and revoking permissions 5-10
  overview 1-2
  stored procedures and 5-25
  System administrators and 5-1
  user alias and 5-7
  views for 5-23
drop logins option, **sp_dropserver** 7-7
Dropping
  groups 3-11
  guest users of *master* 3-7
  logins from Servers 3-13
  remote logins 7-7, 7-8
  servers 7-7
  user aliases 3-19
  user-defined roles 4-10
  user from a database 3-10 to 3-11
  users from Servers 3-13
  users who own database objects 3-11

drop role command **4-10**

E

Ellipsis (...) in SQL statements xxii
Enabling
  auditing 2-2
Enabling auditing 8-6
expand_down parameter
  sp_activeroles 4-21
Expressions
  types of xxiii

F

Finding
  user IDs 3-22
  user names 3-22
  users in a database 3-21
Floating-point data xxiii

G

grant command 5-1, **5-10** to **5-19**
  all keyword 5-16
  "public" group and 5-12, 5-19, 6-3
  roles and 6-3
Granting
  access permissions 4-7
  create trigger permission 5-14
  object creation permissions 4-7
  proxy authorization permission 6-3
  roles 4-9
  roles to roles 4-12
  roles with **grant role** 4-17
grant option
  sp_helprotect 5-20
grant option for option, **revoke** 5-13
Groups
  See also "public" group
  changing 3-16
  conflicting permissions and 5-18 to 5-19
  creating 3-5 to 3-6
  dropping 3-11
  grant and 5-15
  naming 3-5
  permissions hierarchy and 5-13
  revoke and 5-15
Guest users 5-7
  adding 3-7
  creating 3-7
  permissions 3-8
  sample databases and 3-9
Guidelines, security 2-1

H

Hierarchy of permissions. *See* Permissions
Hierarchy of roles. *See* Role hierarchies

I

I/O
  usage statistics 3-24
i/o accounting flush interval configuration parameter 3-25
Identification and authentication
  See also Logins
  controls 1-3
Identities
  alternate 3-17
  proxies and 6-1
  session authorizations and 6-1
Identity of user. *See* Aliases; Logins; Users
IDs, server role
  role_id 4-19
IDs, user 3-22, 5-2
Impersonating a user. *See* **setuser** command
Individual accountability 2-1
Information (Server)
  changing user 3-13 to 3-17
  locked logins 3-12
  logins 3-21
  permissions 5-20 to 5-22

remote server logins 7-13
remote servers 7-7
user aliases 3-19
users, database 3-20 to 3-25
Installation, Server
audit system 8-7
establishing security after 2-1 to 2-6
getting started after 2-2
Integer data
in SQL xxiii
isql utility command
passwords and 7-12

J

Joins
views and 5-24

L

Language defaults 3-4
Linking users. *See* Alias, user
Listing
database users 3-22
Local and remote servers. *See* Remote servers
local option, **sp_addserver** 7-4
Local servers 7-4
Locking
logins 3-11
"sa" account 2-3
Logical expressions xxiii
Login names. *See* Logins
Logins
See also Remote logins; Users
adding to Servers 3-3 to 3-5
alias 3-18
assigning names for 2-2
creating 4-8
database object owner 4-7
"dbo" user name 4-6
displaying account information 4-22
dropping 3-13
finding 3-21
identification and authentication 1-3
information on 3-21
locking 3-11, 4-8
"sa" 2-1, 4-8
unlocking 3-11

M

Managing users. *See* Users
Mapping
remote users 7-7 to 7-11
master database
as default database 3-4
dropping guest users of 3-7
guest user in 3-7
ownership of 5-9
as user default database 3-4
max roles enabled per user configuration parameter 4-10
membership keyword, **alter role** 4-12
Memory
audit records 8-17
Modifying
Server logins 3-15
mut_excl_roles system function 4-21
Mutual exclusivity of roles 1-5, 4-21

N

Names
See also Information (Server); Logins
alias 3-18, 5-7
finding user 3-22
for logins 2-2
group 5-13
mapping remote user 7-8
original identity 5-7
remote server 7-3
remote user 7-8
of roles 4-20
server 7-5
user 3-6, 3-22, 5-10, 5-13
Naming
groups 3-5

servers 7-4
user-defined roles 4-9
net password encryption option 7-6
null keyword
in **sp_addlogin** 3-4
Null passwords 3-15
Number (quantity of)
remote sites 7-14
number of remote connections configuration parameter 7-14
number of remote logins configuration parameter 7-14
number of remote sites configuration parameter 7-14
Numeric expressions xxiii

O

Object access permissions. *See* Permissions
Object owners. *See* Database object owners
Object permissions
grant all 5-11, 5-16
Objects. *See* Database objects
on keyword
grant 5-12
revoke 5-12
Operator role 1-4
permissions 5-4
tasks of 4-6
Options
remote logins 7-11 to 7-12
remote servers 7-5 to 7-6
server 7-5 to 7-6
Order of commands
grant and **revoke** statements 5-18 to 5-19
Owners. *See* Database object owners; Database Owners
Ownership
chains 5-26

P

Packets, network
pre-read 7-14
Parameters, procedure 3-5
Passwords 3-14
changing 3-14
choosing 3-2
choosing secure 3-2
date of last change 3-21
encryption over network 7-6
forgotten 5-4
null 3-15
protecting 3-2
remote users 7-6, 7-11
roles and 4-18
rules for 3-2
sp_password 3-14
Performance
audit queue size 8-17
Permissions
See also Discretionary access control (DAC)
aliases and 3-17
ansi_permissions option and 5-14
assigned by Database Owner 5-16
assigning 5-16
create database 5-3
database object owners 4-7
Database Owners 4-6, 5-1 to 5-2, 5-5
default 5-2
for creating triggers 5-14
granting 5-10 to 5-19
groups and 3-5
guest users 3-7, 3-8
hierarchy of user 5-13
information on 5-20 to 5-22
object 4-7, 5-9
object access 5-10, 5-10 to 5-16
object creation 5-10, 5-16 to 5-17
operator 5-4
ownership chains and 5-26
proxy authorization 6-3
"public" group 5-9, 5-12, 5-19
remote users 7-11

revoking 5-10 to 5-19
selective assignment of 5-18
sp_helprotect display 4-23
stored procedures 5-9, 5-12, 7-11
summary of 5-1 to 5-2
System Administrator 4-5, 5-1 to 5-3
system procedures 5-6
System Security Officer 4-5
system tables 5-6
tables 5-9, 5-12
tables compared to views 5-23
transfers and 5-8
triggers and 5-31
views 5-23 to 5-25
on views instead of columns 5-25
Preferences, user name 3-6
Privileges. *See* Permissions
proc_role system function
stored procedures and 4-21, 5-26
Procedure calls. *See* Remote procedure calls
Processes (Server tasks)
See also Servers
administering Adaptive Server 1-7
current on Server 3-20
information on 3-20
Protection mechanisms. *See* Security functions; Stored procedures; Views
Protection system
See also Permissions
context-sensitive 5-25
hierarchy (ownership chains) 5-27
reports 5-20 to 5-22
summary **5-1** to **5-2**
Proxy authorization **6-1** to **6-8**
getting information about 6-6
granting permission for 6-3
overview 6-1
steps for administering 6-3
steps for using 6-3
"public" group 3-5
See also Groups
grant and 5-12, 5-17, 5-19, 6-3
guest user permissions and 3-8
permissions 5-9
revoke and 5-12, 5-19
sp_adduser and 3-6
sp_changegroup and 3-16
public keyword
grant 5-17, 6-3

R

Records, audit 8-4
Re-establishing original identity 5-7
Remote logins
adding 7-8 to 7-10
configuration parameters for 7-13 to 7-15
dropping 7-7, 7-8
options for 7-11 to 7-12
timing out 7-5
Remote procedure calls **7-1** to **7-15**
configuration parameters for 7-13 to 7-15
remote server pre-read packets configuration parameter 7-14
Remote servers **7-3** to **7-7**
adding 7-3 to 7-15
dropping 7-7
information on 7-7
names of 7-3
options for 7-5 to 7-6
Remote server users. *See* Remote logins
Remote users. *See* Remote logins
Reporting usage statistics 3-24
Reports
Server usage 3-23
revoke command 5-1, **5-10** to **5-19**
"public" group and 5-12, 5-19
Revoking
create trigger permission 5-14
role privileges using **with override** 4-11
roles with **revoke role** 4-17
role_contain system function 4-21
role_id system function 4-19
role_name system function 4-20

Role hierarchies 1-5
creating 4-17
displaying 4-21
displaying with **role_contain** 4-21
displaying with **sp_displayroles** 4-20
Roles
activating 4-18
configured for "sa" login 2-1
deactivating 4-18
displaying permissions with **sp_helprotect** 4-23
in **grant** and **revoke** statements 5-13, 5-17
granting 6-3
granting to System Administrators and System Security Officers 2-3
management of system 4-8
Operator 4-6
permissions and 5-13
stored procedure permissions and 4-21
stored procedures and 5-26
System Administrator 4-4
System Security Officer 4-5
system tables and 4-23
Roles, system 1-3, **4-3** to **4-6**
Operator 1-4
System Administrator 1-3
System Security Officer 1-4
Roles, user-defined
planning 4-9
RPCs. *See* Remote procedure calls
Rules
See also Database objects
protection hierarchy 5-30

S

"sa" login 2-1, 4-8
changing password for 2-1
configured with System Administrator and System Security Officer roles 2-1
locking 2-4
security recommendations for using 2-1
Scripts
permissions on 4-8
Security
auditing 1-6
discretionary access control 1-2
establishing after installation 2-1 to 2-6
functions 1-1 to 1-6
identification and authentication controls 1-3
roles 1-3
Security administration
example of 2-4
getting started 2-1 to 2-6
guidelines 2-1
steps to take 2-2
select * command
error message 5-24
Sensitive information, views of 5-24
Separation of roles 1-3
Sequence of commands. *See* Order of commands
Server aliases 7-4
@@servername global variable 7-4
Servers
See also Processes (Server tasks); Remote servers
adding new logins to 3-3 to 3-5
adding users to 3-3 to 3-5
dropping logins from 3-13
local 7-4
names of 7-4, 7-5
passwords on 7-6, 7-11
remote 7-3 to 7-9
user information 3-20 to 3-25
users currently on 3-21
Server user name and ID 3-22
session authorization option, **set** 6-4
set command
roles and 4-18
setuser command
show_role and 4-20

show_role system function 4-20
Site handlers 7-14
Sites, remote 7-14
sp_activeroles system procedure 4-21
sp_addalias system procedure **3-18**
sp_addauditrecord system procedure 8-22 to 8-23
sp_addgroup system procedure **3-5**
sp_addlogin system procedure **3-3** to **3-5**
  adding System Administrators and System Security Officers 2-3
sp_addremotelogin system procedure **7-8** to **7-10**
sp_addserver system procedure **7-3** to **7-5**
sp_adduser system procedure **3-6** to **3-10**
sp_audit system procedure
  setting options with 8-20
sp_changedbowner system procedure **5-8**
sp_changegroup system procedure 3-5, **3-16**
sp_column_privileges catalog stored procedure 5-22
sp_configure system procedure
  remote logins and 7-13
sp_displaylogin system procedure 3-21, 4-22
sp_displayroles system procedure 4-20
sp_dropalias system procedure **3-19**
sp_dropgroup system procedure 3-10, **3-11**
sp_droplogin system procedure 3-12, **3-13**
sp_dropremotelogin system procedure 7-8
sp_dropserver system procedure **7-7**
sp_dropuser system procedure **3-10**, 3-10
sp_helpremotelogin system procedure **7-13**
sp_helprotect system procedure 4-23, **5-20** to **5-22**, 6-7
sp_helpserver system procedure **7-7**
sp_helpuser system procedure 3-19, **3-21**
sp_locklogin system procedure 3-12
sp_modifylogin system procedure **3-15**
  changing user's default database with 3-4
  changing user's full name with 3-4
sp_password system procedure **3-14**
sp_remoteoption system procedure 7-11 to 7-12
sp_reportstats system procedure 3-24
sp_serveroption system procedure **7-5** to **7-6**
sp_table_privileges catalog stored procedure 5-22
sp_who system procedure **3-20**, 6-7
Square brackets [ ]
  in SQL statements xxii
srvname column, *sysservers* table 7-5
srvnetname column, *sysservers* table 7-5
Statistics
  I/O usage 3-23, 3-24
Steps
  administering security 1-7
  security administration 2-2
Stored procedures
  See also Database objects; System procedures
  checking for roles in 4-21
  granting execution permission to roles 4-21
  ownership chains 5-26
  permissions granted 5-12
  permissions on 5-9, 5-12, 7-11
  remote user access to 7-11
  roles and 5-26
  as security mechanisms 5-25
Stored procedure triggers. *See* Triggers
suid (server user ID) 3-5
  aliases and 3-17
suser_id system function **3-22** to **3-23**
suser_name system function **3-22** to **3-23**
suspend auditing when full configuration parameter 8-17
sybsecurity database 8-2
  giving access to System Security Officers 2-3
sybsystemprocs database
  permissions and 5-7
Symbols
  See also Symbols section of this index
  in SQL statements xxi to xxii

Syntax conventions, Transact-SQL xxi to xxiii
sysalternates table 3-18
  See also sysusers table
syslogins table
  chargeback accounting and 3-24
  sp_addlogin effect on 3-5
  visiting users and 3-9
sysprotects table
  querying 6-7
sysremotelogins table 7-9
sysservers table 7-1, 7-2, 7-3, 7-7
  sp_helpserver and 7-7
  srvname column 7-5
  srvnetname column 7-5
System Administrator 1-3
  adding login for 2-3
  permissions 4-5, 5-1 to 5-3
  role of 4-4
  tasks of 4-4
System audit tables 8-2, **8-24** to **8-32**
System procedures
  for adding users 3-1 to 3-2
  for changing user information 3-13 to 3-17
  for managing remote servers 7-3 to 7-7
  permissions 5-6
System roles 4-2
  activating 4-18
  deactivating 4-18
  displaying permissions with **sp_helprotect** 4-23
  granting with **grant role** 4-17
  max_roles_enabled configuration parameter and 4-10
  Operator 4-6
  show_role and 4-20
  system tables and 4-23
  tasks 4-3
System Security Officer 1-4
  adding logins for 2-3
  permissions 4-5
  role of 4-5
  tasks of 4-5
System tables
  changes allowed to 5-6
  permissions on 5-6
  roles and 4-23
  sp_adduser effect on 3-6
sysusers table
  permissions and 5-7
  sp_adduser effect on 3-6
  sysalternates table and 3-18

T

Table Owners. *See* Database object owners
Tables
  context-sensitive protection of 5-25
  ownership chains for 5-26
  permissions information on 5-22
  permissions on 5-9, 5-12
  permissions on, compared to views 5-23
  underlying 5-24
Threshold procedures
  audit trail 8-11
timeouts option, **sp_serveroption** 7-5
Transferring ownership. *See* Database objects, ownership
Triggers
  See also Database objects; Stored procedures
  creating 5-14
  permissions and 5-31
Trusted mode
  remote logins and 7-11

U

Underlying tables of views (base tables) 5-24
Unlocking login accounts 3-11
Untrusted mode, remote logins and 7-11
Updating

system procedures and 5-26
Usage statistics 3-24
user_id system function 3-23
user_name system function 3-23
User databases
See also Databases; Permissions
User-defined roles
activating 4-18
configuring 4-10
creating 4-10
deactivating 4-18
displaying permissions with **sp_helprotect** 4-23
dropping 4-10
granting with **grant role** 4-17
number of 4-10
planning 4-9
system tables and 4-23
User groups. *See* Groups; "public" group
User IDs 5-2
displaying 3-21
finding 3-22
User names 3-22, 5-10
changing 3-15
finding 3-22
preferences 3-6
User objects. *See* Database objects
Users
See also Aliases; Groups; Logins; Remote logins
adding 3-1 to 3-6, 3-6 to 3-10
aliases 3-17
currently on database 3-20
currently on Server 3-20
dropping from databases 3-10 to 3-11
dropping from groups 3-16
dropping from Servers 3-13
guest 3-7, 5-7
IDs 3-22, 5-2
information on 3-20 to 3-25
permissions to all or specific 5-18, 5-25
remote 7-7 to 7-11
views for specific 5-24
visiting 3-9
Users, object. *See* Database object owners

V

Verification, user-access 7-6
Views
See also Database objects
dependent 5-27
ownership chains 5-26
permissions on 5-12, 5-23 to 5-25
security and 5-23
Visitor accounts 3-9

W

with grant option option, **grant** 5-13
with override option
drop role 4-11