Introduction to Interactive Query Language

Order Number AA-0947B-TK

July 1979

ABSTRACT

This manual contains introductory usage information about the query capability of Interactive Query Language.

SUPERSESSION INFORMATION: This document replaces *DECsystem-10 Introduction to Interactive Query Language,* Order Number DEC-10-LIQLA-A-D.

SOFTWARE VERSION:
IQL Version 3
DBMS Version 5
COBOL Version 12
SORT Version 3 (TOPS-10)
SORT Version 4 (TOPS-20)

OPERATING SYSTEM:
TOPS-20 Version 3
TOPS-10 Version 6.02

Software and manuals should be ordered by title and order number. In the United States, send orders to the nearest distribution center. Outside the United States, orders should be directed to the nearest DIGITAL Field Sales Office or representative.

NORTHEAST/MID-ATLANTIC REGION

Technical Documentation Center
Cotton Road
Nashua, NH 03060
Telephone: (800) 258-1710
New Hampshire residents: (603) 884-6660

CENTRAL REGION

Technical Documentation Center
1050 East Remington Road
Schaumburg, Illinois 60195
Telephone: (312) 640-5612

WESTERN REGION

Technical Documentation Center
2525 Augustine Drive
Santa Clara, California 95051
Telephone: (408) 984-0200

digital equipment corporation • marlboro, massachusetts

First Printing, December 1975
Revised, July 1979

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

Digital Equipment Corporation assumes no responsibility for the use or reliability of its software on equipment that is not supplied by DIGITAL.

The postage prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DIGITAL	DECSYSTEM-10	MASSBUS
DEC	DECTAPE	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-10
DECCOMM		TYPESET-11

CONTENTS

PREFACE

The **Introduction to Interactive Query Language** is written for the user who has never worked with the Interactive Query Language (IQL). Users who have some experience with IQL but want to review its concepts should also find it useful. This manual contains examples showing how to use all the basic query statements, with explanations of how the statements are put together to form a complete query.

You do not need to know a great deal about TOPS-10 or TOPS-20 to use IQL. If you are familiar with **Getting Started with the DECsystem-10** or **Getting Started with TOPS-20**, you are ready to learn IQL. This manual assumes that you know how to log in to the system and print files on a line printer. Some experience with a text editor is also helpful.

Programmers who are familiar with COBOL will notice that many of the features of IQL are similar to features of COBOL. However, IQL is not a programming language. IQL is a general-purpose query language with the ability to interactively interrogate data files and to generate reports based on that data. Rather than writing a separate COBOL program for each application, you can use IQL.

When you have finished reading this manual, and have tried out the examples, you should proceed on to the **IQL User's Guide.** That manual is a more complete reference for all the features of IQL, including many that are not fully treated in this manual.

The following documentation is referenced in this manual:

1. **Interactive Query Language User's Guide** (SDC Order Number AA-H282A-TK).
2. **Getting Started with TOPS-20** (SDC Order Number AA-4187C-TM).
3. **Getting Started with the DECsystem-10** (SDC Order Number AA-C790A-TB).
4. **TOPS-20 Edit User's Guide** (SDC Order Number AA-4182A-TM).

CHAPTER 1

INTRODUCTION

The Interactive Query Language (IQL) is an information retrieval and reporting system. You can use IQL to:

- Create data dictionaries of all your files
- Read a specific piece of information
- Change a specific piece of information
- Update an entire file
- Copy selected portions of your data to a new file
- Write reports based on your data.

When working with a file, you can use IQL to set up the format of your report, sort records, summarize data, compute new data, and write new data in your file.

IQL can be used equally well by experienced computer professionals or by people with no background in computers. This manual gives you the basics of IQL so you can get started working with it right away. You should try all of the examples shown in the following chapters so that you become familiar with the normal input and output of IQL queries. These examples use files that are included on the IQL distribution tape, so you should be able to reproduce the results as shown (note that some of the reports shown in this manual have been shortened to save space).

The fundamental concepts of IQL are presented in Chapter 2. Chapters 3, 4, and 5 contain examples of IQL queries. The first queries show some of the simpler features of IQL. Later queries show some of the more complex features. You should try each example so that when you finish the manual, you will be familiar with the operation of IQL.

The examples in each chapter center around a single theme. Chapter 3 draws on Personnel examples, Chapter 4 works with a Marketing application, and Chapter 5 is based on problems in Accounting. If you work in one of these fields you may find the problems familiar, but you should nonetheless read all of the preceding chapters in order to learn the preliminary skills.

CHAPTER 2

IQL CONCEPTS

2.1 OVERVIEW

The two key elements of the Interactive Query Language are queries and dictionaries. A query is a set of IQL statements that you use to manipulate data and generate reports. A dictionary is an exact description of a data file. IQL uses the dictionary to find the information you request. Section 2.3 discusses dictionaries in detail. Chapters 3, 4, and 5 contain examples of queries and explanations of their use.

The vocabulary in an IQL statement comes from two sources: the IQL vocabulary and the names in your own dictionaries. The IQL vocabulary consists of reserved words like OPEN, PRINT, and COMPUTE. These words have predetermined meanings. The names in your dictionaries, on the other hand, are defined by you. There must be one dictionary for each data file you want to use in IQL. The names in the dictionary refer to the data items in that data file.

2.2 QUERY MODES

IQL operates in three modes: Assistance, Immediate, and Deferred. You enter Assistance mode when you start IQL. You can tell when you are in Assistance mode by the prompt, which is <QA>. Immediate mode is used to interact directly with your files. The prompt for the Immediate mode is <QU>. IQL goes into Deferred mode when you run or execute a query. Deferred mode has no prompt.

2.2.1 Assistance Mode

Many Assistance mode commands give you information about the elements of IQL. Using Assistance mode commands, you can find out the names of all the queries and dictionaries in your directory. You can then print any of them on your terminal. You can also write new queries or edit ones that are already written. You can define or change dictionaries in Assistance mode. You use other Assistance mode commands to enter Immediate mode, or to run a query in Deferred mode.

2.2.2 Immediate Mode

In Immediate mode, you interact directly with your file. In this mode, you can browse through a file, reading data items or records. You can also input new information, either by changing old information or by adding new records to the file.

Immediate mode is useful for getting quick answers to questions. As long as the information is in your file and a dictionary includes that data item, you can access the information. You can make fairly complex queries in Immediate mode. For instance,

```
<QU>LIST ACCOUNT AND SALESMAN IF 78-SALES NOT GREATER 77-SALES
        AND SALARY GREATER 50000
```

As you can see, an Immediate mode query is similar to English. This manual does not include any further discussion of this mode. Refer to the **IQL User's Guide** for a more complete description of the Immediate Mode.

2.2.3 Deferred Mode

When you run a query, IQL goes into Deferred mode. In Deferred mode, IQL first reads and analyzes the statements in the query. If there are any syntax errors, IQL informs you of them at this time. If there are no errors, IQL reads the dictionary and data file, extracts the information, and performs any necessary manipulations. If you request a report, IQL prepares it for you.

You use Deferred mode mainly for queries that involve data manipulation or report writing. Data manipulation includes sorting records, computing new values based on the original data, and transferring the data to new files. IQL queries are especially useful for writing reports. The reports can have a variety of formats to accommodate the information you want to include and the forms you are using. The examples in this manual introduce you to many of IQL's report writing features.

2.3 DICTIONARIES

To access a data item using IQL (in either Immediate or Deferred mode), you must use the item name from a previously defined dictionary. A dictionary is similar to a File Description (FD) in a COBOL Data division. Both contain storage and access information such as item definitions, editing pictures, usage declarations, blocking factors, and definitions of ISAM or DBMS keys. In addition, the dictionary holds a 'title' for each item. The report generator uses this title when column headings are needed. These components are explained in Section 2.3.1.

2.3.1 DICTIONARIES and ITEMS

The first two IQL commands you will need are DICTIONARIES and ITEMS. These are both Assistance mode commands. Assistance mode commands are meaningful only when you type them in response to the Assistance prompt. If you are running IQL with TOPS-20, you can use abbreviation and recognition with IQL commands. If you are running with TOPS-10, you may abbreviate Assistance mode commands to the first three characters.

When you issue the DICTIONARIES command, IQL lists all dictionaries defined in your directory, and shows some information about each one. The ITEMS command lists all the definitions in a specific dictionary. The following example shows both of these commands.

In this example, the DICTIONARIES command is issued first, to obtain a list of the dictionaries available. (Note that these dictionaries, including PERSONNEL, are available on the IQL distribution tape in the file QPDICT.SEQ.) Then the ITEMS command is issued with an argument of PERSONNEL. This generates a list of all the definitions in the PERSONNEL dictionary. The headings in these two lists are explained following the example.

One thing of interest is the use of passwords in the example. In the DICTIONARIES portion of the example, the descriptions of the PERSONNEL and PAYROLL dictionaries contain asterisks under the headings for passwords (PW). These asterisks indicate that passwords exist, but the passwords are not revealed. In the ITEMS portion of the example, the name of the dictionary (PERSONNEL) is followed by the 'dictionary unlocking password' (SESAME). Thus, the password references are revealed. For more information on passwords, refer to the **IQL User's Guide**.

```
<QA>DICTIONARIES
 DICTIONARIES IN YOUR DIRECTORY:

DICT        FILE    FILE-IN                    REC  BLK  KEY KY KY RD CP RW
NAME        TYPE    NAME        DIRECT         LEN  FAC  LOC LN TP PW PW PW
-----       ----    -----       -------        ---  ---  --- -- -- -- -- --

PERSONNEL   IS DSK7             <IQL30-DIST>                          ** ** **
COLLEGE     SQ DSK7 COLEGESEQ                   40    0    0  0
JOBFILE     SQ DSK7 JOBNMESEQ   <IQL30-DIST>    40    0    0  0
CUSTOMERS   SQ DSK7 CUSTMRSEQ                  145    0    0  0
SALESMEN    SQ DSK7 SLSMENSEQ   <IQL30-DIST>  135    0    0  0
PAYROLL     SQ DSK7 PAYFILSEQ   <IQL30-DIST>   87    0    0  0       ** **
BALANCES    SQ DSK7 BANKS SEQ   <IQL30-DIST>   80    0    0  0
RANGES      SQ DSK7 RANGESSEQ                   80    0    0  0
CHARTS      SQ DSK7 CHARTSSEQ                   80    0    0  0
DICTIONARY  SQ DSK7 QPDICTSEQ                  120    0    0  0
MASTER      IS DSK7 PERSONIDX                  300    2    1  5 AU
QUERIES     SQ DSK7 QPQRYSSEQ                   80    0    0  0
JOB-TABLE   IS DSK6 JOBNMEIDX                   50   25    1  2 AU
COLLEGE-TABLE
            IS DSK7 COLEGEIDX                   40   10    1  2 AU
 (END LIST OF DICTIONARIES)

<QA>ITEMS PERSONNEL SESAME

DICT        FILE    FILE-IN                    REC  BLK  KEY KY KY RD CP RW
NAME        TYPE    NAME        DIRECT         LEN  FAC  LOC LN TP PW PW PW
-----       ----    -----       -------        ---  ---  --- -- -- -- -- --

PERSONNEL   IS DSK7 PERSONIDX   <IQL30-DIST>  300    2    1  5 AU 10 30 50

   ITEM     TOP         BOTTOM      1ST  NO.  T S PRINTING        SCAN
ID NAME     TITLE       TITLE       CHAR CHAR Y C PICTURE         GNNS PT
-- -------- ----------- ----------- ---- ---- - - --------------- ---- --

PD TIGER                            50
PD BEAR                             30
PD FOX                              10
KD EMPNO    EMPLOY      NUMBER       1    5 N 0 ZZZZ9
DD CAT      CAT                      6    1 A 0
DD LOCATOR                LOCATOR    7   12 A 0 XXX-XXXX-XXXXX
DD CO       CO                       7    3 N 0 ZZ9
DD DIV      DIV                     10    4 N 0 ZZZ9
DD DEPT     DEPT                    14    5 A 0
DD SOC-SEC  SOCIAL      SECURITY    19    9 N 0 999-99-9999
```

```
DD LNAME      LAST         NAME          28   15 A 0
DD FNAME      FIRST        NAME          43   10 A 0
DD INIT       I                          53    1 A 0
DD NAME       FULL NAME                  28   26 A 0
DD STREET     STREET                     55   20 A 0
DD COLLEGE-ID
              COLL         CODE          77    2 A 0
DD JOB-ID     JOB          CODE          79    2 N 0 99
DD DATE-CH    CHANGE        DATE         81    6 N 0 99-99-99          A129
DD YR-CH      CH           YR            81    2 N 0 SZ9               A129
DD MO-CH      CH           MO            83    2 N 0 SZ9               A129
DD DA-CH      CH           DA            85    2 N 0 SZ9               A129
DD CITY       CITY                      154   20 A 0
DD STATE      STATE                     181   14 A 0
DD MAILCODE   ZIP                       207    5 N 0 99999
DD ZIP         ZIP         CODE          207    5 N 0 99999
DD PHONE      FULL         PHONE         212   10 N 0 (999)999-9999
DD AREA       AREA         CODE          212    3 N 0 (999)
DD LCL-PHN    LOCAL        PHONE         215    7 N 0 999-9999
DD SALARY     SALARY       WEEKLY        222    6 N 2 SZZZ9.99                30
DD HRLY       HOURLY       RATE          228    5 N 3 Z9.999                  30
DD OT-HRLY    OVT-TME      RATE          233    5 N 3 Z9.999
DD OLD-SAL    PREVIOUS     SALARY        238    6 N 2 ZZZ9.99
DD OLD-HRL    PREV-HRLY    RATE          244    5 N 3 Z9.999
DD DATE-LR    LAST RAISE   YY-MM-DD      249    6 N 0 99-99-99
DD DATE-LRA   DATE         LRA           249    6 A 0 XX-XX-XX
DD YR-LR      YR           LR            249    2 N 0
DD MO-LR      MO           LR            251    2 N 0
DD DA-LR      DAY          LR            253    2 N 0
DD DATE-HR    DATE         HIRED         255    6 N 0 99-99-99
DD YR-HR      YR           HR            255    2 N 0
DD MO-HR      MO           HR            257    2 N 0
DD DA-HR      DAY          HR            259    2 N 0
DD DATE-BD    DATE         BIRTH         261    6 N 0 99-99-99
DD YR-BD      YR           BD            261    2 N 0
DD MO-BD      MO           BD            263    2 N 0
DD DA-BD      DAY          BD            265    2 N 0
DD SKILL       JOB         SKILL         273    4 A 0 XXX-X
DD SK-CDE     SKL          CDE           273    3 A 0
DD SK-C1      S            1             273    3 A 0
DD SK-C2      S            2             274    3 A 0
DD SK-C3      S            3             275    3 A 0
DD SK-LVL     SKL          LVL           276    1 A 0 X

(END LIST OF ITEMS)
```

The output of the ITEMS command is explained below. The first twelve headings in the output are the same for both commands. Note that the dictionary shown above is shortened, and the last few items are not shown here.

DICT NAME -- The name of the dictionary.

FILE TYPE -- There are two pieces of information under this heading. First is a code that identifies the access method of the file. The IS stands for Indexed Sequential. If you refer back to the DICTIONARIES output, you see that many of the other dictionaries are for sequential files, denoted by the code SQ. If a dictionary refers to a data base, the code is DTABASE.

The second piece of information under this heading indicates the recording mode of the data in the file.

DSK7 means that the data is stored in ASCII mode. DSK6 means that the data file is written in SIXBIT format; the JOB-TABLE dictionary has an example of this. (The recording mode does not affect your interaction with the data; it refers only to the internal storage mode.)

FILE-IN NAME-- This is the name of the file in which the data can be found. Notice that the period between the file name and the extension is omitted.

DIRECT -- This is the name of the directory or P,PN where you are keeping the file. This is for record-keeping information only, because IQL looks only in your connected directory to find the file.

REC LEN -- This is the length of the records in the data file.

BLK FAC -- This and the next three headings apply only to dictionaries of Indexed Sequential files. The four parameters (blocking factor, key location, key length, and key type) must be set up with the ISAM utility when the file is created. The blocking factor is listed under the heading BLK FAC.

KEY LOC -- This is the location of the first character of the key. The key can be located anywhere in the record; in the PERSONNEL dictionary, the key begins at the first character in the record.

KY LN -- This stands for 'key length', which is 5 characters in this example.

KY TP -- The two characters in this column identify the key type. The first letter can be A, for alphabetic, or N, for numeric. The second letter determines whether the key is signed (S) or unsigned (U). Alphabetic keys, such as the one in this example, must be unsigned.

RD PW -- This indicates 'read password'. The number in this column identifies the password level. If you look further down in the dictionary, you can see that the name FOX has a 10 alongside it. FOX is the password that allows you to read the data in the file using this dictionary.

CP PW -- This column identifies the level number of the copy password. This is the password that you must give if you want to copy, update, or write the data in the file. It is important to note in this example that BEAR has a higher level number (20) than FOX (10). This means that the password BEAR gives you all the authority that FOX would, plus more. In other words, if you give the password BEAR but not the password FOX, you are not prevented from reading the file.

RW PW -- This is the level number of the rewrite password. Rewriting applies only to Indexed Sequential files. Notice that this password has the highest level of the three. This means that the password TIGER gives you complete authority to do anything with the file.

This ends the description of the first 12 headings. The information in these columns is the same in both the DICTIONARIES and ITEMS

output. The discussion that follows applies only to the output of the ITEMS command.

ID -- The ID tells you what type of definition the line contains. The first item in the example is a PD, or password definition. The fourth item is a KD, or key definition (you can see the correspondence between this item and the references to the key in the first portion of the output). All the DD's are data definitions. Other IDs are RD (record definitions), AD (area definitions), SD (set definitions), and CD (comments).

ITEM NAME -- This is the name you must use to reference a data item whenever you make a query. CAT, for instance, denotes a data field in every record of the PERSON.IDA data file.

TOP and BOTTOM TITLE--The words in these two columns form a column heading whenever you have the item printed in a report. The headings can be two lines long. Each line may contain any printing character, and may be up to ten characters long.

1ST CHAR -- This number denotes the location in the record of the first character of the data item.

NO. CHAR -- The length, in characters, of the field containing this data item.

TY -- The data type: A for alphanumeric, N for numeric, or B for binary.

SC -- The scale, for numeric items. The scale is the number of places to the right of the decimal point.

PRINTING PICTURE--When IQL prints the data in an item, IQL edits it according to the picture specified.

SCAN G, NN, and S--The column under G contains the name of the scan group. NN refers to the number of repeats in the group (for those familiar with COBOL, this is the number of times the item OCCURS). The column under S contains the stop character.

PT -- This is the protection level of an individual data item. The level number refers to one of the PD entries.

Refer to the **IQL User's Guide** for more complete discussions of the topics described in this section.

2.3.2 Writing Queries

This manual contains examples that illustrate most of IQL's query capability. You can write these queries on your own computer by following this procedure:

1. After logging in, type R IQL to the operating system command prompt. The program prints the Assistance prompt, <QA>.

2. Type WRITE. IQL calls an editor for your use. If you are running under TOPS-20, IQL calls EDIT, the standard system editor (refer to the **TOPS-20 EDIT User's Guide**). If you are running under TOPS-10, IQL calls its own editor (refer to Appendix E in the **IQL User's Guide**).

3. Copy the example queries from this book, and end the editing session. IQL returns to Assistance mode (indicated by the <QA> prompt).

4. Type RUN. IQL runs the query and displays the results on your terminal. Finally, IQL informs you about a listing file, where a copy of your report is now stored. You are now back in Assistance mode, and you may move on to the next example.

2.3.3 Sample Complete Run

The example in this section shows a complete run of IQL. The name of the query, EXAMP2-1, indicates that this is the first example in Chapter 2. You can name a query at any time when you are working on it, using the WRITE, STORE, or EDIT command. If you want to keep the query, you must first give it a name and then store it with the STORE command. You can find out the names of all your stored queries by issuing the QUERIES Assistance mode command. If you do not store the query, it disappears when you begin working with another query.

The first IQL command shown in the example is the WRITE command, and a query is written as explained above. The statements in the query are explained at the end of the example. After the query is written and the editing session ended, the query is stored for future reference using the STORE Assistance mode command. When the RUN command is issued, IQL first reads the query and displays it on your terminal. If there are any errors in your query, IQL gives you error messages during this phase. The query is then processed (this may take 10 to 20 seconds), and the resulting report is displayed on your terminal when it is ready. IQL also writes the report in a listing file, and identifies the name of this file at the end of the query.

The query in this example is four lines long. Each line is like a simple English imperative sentence, and ends with a space and a period.

```
@R IQL(RET)

<QA>WRITE EXAMP2-1(RET)

%File not found, Creating New file
Input: QC011S.TMP.5
00100   OPEN PERSONNEL . (RET)
00200   AUTHORITY TIGER . (RET)
00300   HEADING "PERSONNEL LIST" . (RET)
00400   PRINT NAME,SOC-SEC,DATE-BD,SALARY . (RET)
00500   (ESC)
*EUN    (RET)

[QC011S.TMP.5]

<QA>STORE EXAMP2-1 (RET)
 (EXAMP2-1 STORED)
```

```
<QA>RUN EXAMP2-1 (RET)
**EXAMP2-1

OPEN PERSONNEL .
AUTHORITY TIGER .
HEADING 'PERSONNEL LIST' .
PRINT NAME,SOC-SEC,DATE-BD,SALARY .
```

02/13/79 PERSONNEL LIST PAGE 1

FULL NAME			SOCIAL SECURITY	DATE BIRTH	SALARY WEEKLY
BARKER	WILLIAM	I	109-46-6471	19-07-17	517.72
BROWN	BETTY	J	746-51-9853	31-03-06	169.96
DI VALERA	CHARLES	L	449-98-1742	46-05-11	349.60
MORANDI	ANTHONY	F	377-60-6260	27-05-11	335.36
BRENNER	BRENDA	B	515-75-6669	41-03-07	211.40
BRETTLER	ROBERT	A	525-21-5977	30-04-10	302.80
BARNES	DAVID	M	122-53-8127	40-03-06	192.24
CHAMPION	HELEN	B	791-50-9122	24-03-07	204.72
COREOLIS	CORINNA	D	198-00-3178	43-07-18	554.08
FRANK	ARTHUR	C	848-02-4029	25-07-16	504.56
MORALES	ELENA	M	964-21-1938	33-03-06	176.88
O BRIEN	FREDRICK	N	168-80-7733	45-04-10	299.28
NELMS	KEVIN	S	989-19-0144	29-04-09	281.60
REILLY	SEAN	K	629-41-5727	26-06-15	476.56
HALL	RICHARD	B	101-15-4880	41-04-11	326.56
MEYERS	MELINDA	L	664-89-4029	29-04-10	292.56
GODBY	CLAIR	G	053-00-8823	21-03-07	219.20
EVERINGHAM	LLOYD	A	042-06-7882	23-04-08	253.92
ARAOS	LADALIO	P	950-97-3208	41-06-16	491.04
ZARZEWSKA	IRENE	G	211-09-1204	34-05-11	346.68
MELZER	WILLIAM	M	396-61-0265	48-02-01	122.04
RYAN	DEBBIE	P	813-20-8219	42-03-06	187.16
AWASTHI	BIPIN	N	867-44-3556	46-03-06	185.76
JARAMILLO	CARLOS	A	033-76-1683	16-07-16	503.28
BALLARD	PHYLLIS	S	225-56-9780	22-03-06	188.12
CRUZ	HENRY	Z	167-67-1288	44-04-10	300.96
SLOUGH	SELENA	S	211-61-7237	36-03-06	179.24
PRATT	PEARL	B	165-84-2850	42-06-15	467.48
KOURANEY	OSCAR	P	491-57-3737	18-04-10	306.20
BARFIELD	CHARLES	R	956-75-4298	46-02-01	123.04
MUCKELROY	ROBERT	H	350-83-5913	38-08-19	592.20
SIMMS	MARY	I	399-91-5022	35-03-07	198.04
SCHLAFKE	NANCY	W	848-47-2038	18-07-16	514.84

```
(END QUERY PHASE; PRINT FILE IS QL015ELPT)
```

2.3.4 OPEN

The first statement is OPEN PERSONNEL. PERSONNEL is the name of the dictionary that this query addresses. (The PERSONNEL dictionary is shown in the example of the ITEMS command.) That dictionary contains the names of the data items, the name of the file containing the data, and information about the format and title of each data item.

2.3.5 AUTHORITY

The second statement is AUTHORITY TIGER. You use the AUTHORITY statement when you must give a password in an IQL query. TIGER is the password for level 30. The level 30 password is referenced twice in the dictionary. The first reference is for rewrite protection for the entire file (in the column labeled RW PW). Thus the password TIGER would allow the query to write new information back into the data file. Also, level 30 protection is referenced by the data definition of the SALARY item (in the column labeled PT). Salary is thereby protected. This query reads SALARY in the PRINT statement, so the query must include the password TIGER to get access to that item. The rest of the file is protected at level 10 (the 'read password' level). Since 30 is higher than 10, the password FOX is not necessary because TIGER is already there.

2.3.6 HEADING

The next statement in the query is a HEADING statement. The heading is the title of the report, which appears at the top of every page along with the today's date and the page number. The words PERSONNEL LIST are enclosed in quotes, so these words are printed exactly as they are typed in. Any series of words enclosed in quotes is a literal. IQL does not try to interpret literals using its own vocabulary.

2.3.7 PRINT

The final statement in the sample query generates the body of the report. The PRINT statement names the items that are printed in the report. The items that this report contains are NAME, SOC-SEC, DATE-BD, and SALARY. If you compare the report with the dictionary definitions of these item names, you can see the correspondence between the dictionary and the query. This correspondence is explained below.

2.3.7.1 **Item Names** - NAME is the first item name referred to. The PERSONNEL dictionary contains a DD entry for NAME. In the TOP TITLE column, the dictionary has the title 'FULL NAME'. As you can see in the report, this title is printed over the column containing the names.

The next two columns in the dictionary (1st CHAR and NO. CHAR) tell IQL where the first character of each item is located within the record and the length of the item. NAME starts at the 28th character and occupies the next 26 characters in the record. If you look at the next three items in the dictionary, you can see that there is an overlap in this part of the record. The LNAME item also starts at character 28, but is only 16 characters long. The FNAME and INIT items are also in the same range, with the one-character initial falling on the last character of the NAME item. NAME is therefore a multiple-entry item, which you can use to access a single field of the record with a variety of item names.

The SOC-SEC item makes use of the 'picture' facility in the dictionary. All Social Security Numbers have the same format, so this format can be stored independently of the data itself. The SOC-SEC data is only nine characters long; this does not include space for

the two hyphens. The hyphens are inserted when the data is printed. To do this, IQL uses the PRINTING PICTURE that was defined in the dictionary. The picture in this case is 999-99-9999. The 9's in this picture are replaced with the numbers stored in the data record, and the data is then printed out with the hyphens inserted. DATE-BD and SALARY also use this feature.

In this query, IQL reads every record in order by the employee number. The employee number is the key of the index, so you do not have to reorder the records. In other instances, you may want to reorder the records in the file, or select a specific subgroup of them for certain reports. Examples later in this manual show how you can do this.

CHAPTER 3

SAMPLE PERSONNEL QUERIES

This chapter contains sample queries that use the PERSONNEL dictionary discussed in Chapter 2. Each query is based on a fictional situation where someone in a large corporation requests a personnel report from you. The Personnel Department keeps all employee data on a computer so you can use IQL to generate the reports.

This chapter introduces the basic features of IQL. These features include setting up the format of reports, selecting the records on which to base a report, sorting the records into a desired order, and obtaining summaries. Some of the basic techniques for writing queries are also discussed. The first queries are relatively simple, and more complex queries follow.

If you are a novice user, you should concentrate on the new statements that are introduced in each example. You may want to look up each statement in the **IQL User's Guide** to become familiar with that document. When you are familiar with the IQL statements, you should review this chapter to study the techniques that are shown.

3.1 SETTING UP A FORMAT

The example in this section shows some of the basic features that are used to change report formats. IQL uses the default format, unless you explicitly change some of the format settings. Every setting has a default; for instance, the default for the left margin is column 1 and the default for the right margin is column 132. Using the various format statements, you can change these values to the ones you need for your report.

The Problem:

Ms. Alexander, who is in charge of telephone services, is putting together a new telephone directory. She wants a rough draft to work with, and requests all telephone numbers in order by employee number. She wants the date on the report to be the first of the year.

The Solution:

When writing a query, the first thing you should determine is what items of information you want the report to contain. From the request, you can see that the report should have two pieces of information: the employee names and phone numbers.

Second, you should determine the scope of the report. In this case, the report should contain data on every employee. From these two considerations, you can tell that the PRINT statement, which was shown in EXAMP2-1, should be the one to control the selection of data.

The third thing to determine is the format of the report. One special consideration is that Ms. Alexander does not want today's date on the report. IQL normally prints today's date at the top of each page, so you must override this action. The IQL statement to use in this case is DATE, which performs exactly the action that you need: it allows you to specify the date you want printed at the top of the page.

Another format consideration is that she wants the report to be a rough draft. You can accommodate this by double-spacing the report and by leaving wide margins on both sides. IQL normally single-spaces the reports, but you can override this default using the VSPACE statement followed by the number of vertical spaces you want from one line to the next. To make the left margin wider, you can declare the number of spaces you want with a LMARGIN statement.

The report needs a heading, so you should use the HEADING statement described in EXAMP2-1. To make the report easier to read, the column title of the PHONE item is changed from FULL PHONE to TELEPHONE NUMBER using a TITLES statement.

The Query:

```
<QA>RUN EXAMP3-1
 **EXAMP3-1

HEADING 'DRAFT//TELEPHONE//DIRECTORY' .
HSPACE 2 . VSPACE 2 . LMARGIN 15 .
DATE 010179 .
TITLES PHONE = 'TELEPHONE//NUMBER' .
OPEN PERSONNEL .
AUTHORITY FOX .
PRINT LNAME,FNAME,INIT,PHONE .
```

The first four lines in this query contain the format information. It is a good practice to put the overall format statements at the top of the query, before the OPEN statement. The first statement is HEADING, which is shown in EXAMP2-1. However, the statement in this example is slightly different from the previous one because it contains double slashes (//) within the literal. If you look at the report below you can see the effect that this has. A double slash inside any literal causes the text to be printed on more than one line; the multiple-line heading is centered at the top of each page in the report.

The second line of the query contains three statements separated by periods. You can put more than one short statement on a single line if they are separated like this. (Note that a line in a query cannot be over 80 characters long.) The HSPACE statement sets the number of horizontal spaces between items in the report. The default for this value is one space. In this example, the spacing is changed to two. The VSPACE and LMARGIN statements are written the same way. VSPACE 2 sets the vertical spacing to two, and LMARGIN 15 forces the first column of the report to be printed on column 15 of the page.

On the next line of the query, the date of the report is set to January 1, 1979 (in the format MMDDYY). The date on each page of the report has this value.

The TITLES statement has another use of a literal. If you look back at the dictionary, you can see that the PHONE item normally gets the column title 'FULL PHONE'. You can change this with the TITLES

statement. The new title is written as a literal, like the literals used in the HEADING statement. Notice the use of a double slash to get a two line title.

At this point, the format is set up and the query is ready to read the data. The next three statements are similar to the ones used in EXAMP2-1. The OPEN statement opens the PERSONNEL dictionary. The AUTHORITY statement gives the password FOX. This is the password for read-access (the RD PW). This query does not access any item that is protected at a higher level (such as the SALARY item), and the query is not involved in copying or rewriting the data in the file. Therefore, the password for level 10 is the only one necessary. The PRINT statement, which is the last statement in the query, causes the data to be printed in the format you have set up.

The Report:

01/01/79 DRAFT TELEPHONE DIRECTORY PAGE 1

LAST NAME	FIRST NAME	I	TELEPHONE NUMBER
BARKER	WILLIAM	I	(715)109-4664
BROWN	BETTY	J	(534)746-5198
DI VALERA	CHARLES	L	(424)449-9817
MORANDI	ANTHONY	F	(608)377-6062
BRENNER	BRENDA	B	(694)515-7566
BRETTLER	ROBERT	A	(778)525-2159
BARNES	DAVID	M	(278)122-5381
CHAMPION	HELEN	B	(221)791-5091
COREOLIS	CORINNA	D	(788)198-0031
FRANK	ARTHUR	C	(290)848-0240
MORALES	ELENA	M	(383)964-2119
O BRIEN	FREDRICK	N	(338)168-8077
NELMS	KEVIN	S	(449)989-1901
REILLY	SEAN	K	(273)629-4157
HALL	RICHARD	B	(802)101-1548
MEYERS	MELINDA	L	(299)664-8940
GODBY	CLAIR	G	(231)053-0088

01/01/79 DRAFT TELEPHONE DIRECTORY PAGE 2

LAST NAME	FIRST NAME	I	TELEPHONE NUMBER
EVERINGHAM	LLOYD	A	(825)042-0678
ARAOS	LADALIO	P	(086)950-9732
ZARZEWSKA	IRENE	G	(045)211-0912
MELZER	WILLIAM	M	(651)396-6102

(END QUERY PHASE; PRINT FILE IS QL018ELPT)

This query shows how you override some of IQL's defaults for the format of a report. IQL sets up a format automatically, but you can alter this format using statements within the query. You can modify other kinds of defaults as well. EXAMP3-2 shows you how to alter a default for the scope of the report.

3.2 SELECTING RECORDS

The report in EXAMP3-1 contains information about every employee in the corporation. However, you will often want to generate a report that includes information about a group of employees, rather than all of them. EXAMP3-2 shows you how to use IQL to select certain records from the data file.

The data file in these examples contains one record for each employee. Each record contains several items of information, as defined in the dictionary for the file. The previous examples show that any of these items can be printed in a report. Just as IQL can pick out a particular item for printing, it can also pick out a particular item and test it to see if it meets a condition that you specify. Using the IF statement, you can limit the scope of the query to those records that meet the specified condition.

The Problem:

The manager of Department number 26804 requests a report containing the overtime pay rates for the employees in her department. She wants the report to contain the department number, the employee's name and number, and a special highlight in front of the overtime rate.

The Solution:

This problem requires special consideration in one area: limiting the scope of the query. Also, a special format is necessary. However, it is always a good practice to first identify the information that you want to include in the report. In this case, the information is the department number, the employee's name and number, and the overtime pay rate. These items are defined in the PERSONNEL dictionary, so they pose no special problems.

The next thing to consider, when writing a query, is the scope of the report. In this problem, you do not want information about every

employee. The report must include an employee only if he is in Department 26804. In IQL, the IF statement handles this need. Statements that begin with IF are called 'Conditional' statements because they specify that an operation should be performed only if a given condition is true.

Conditional statements are made up of two or sometimes three clauses. The first clause must specify the condition to be met. In this example, the condition is that the employee must be a member of Department 26804. The second clause must specify the operation to perform if the condition is true. Since you want information printed about the employees in the selected department, you must use a PRINT statement in the second clause of your conditional. The third clause, which is optional, specifies the operation to perform when the condition is false. You can ignore the other records in this report, so there is no need for a third clause. If you were writing in English rather than IQL, these three clauses together would form a sentence like, If such-and-such is true, then do so-and-so; otherwise do something else. The way this is actually written in IQL is analyzed following the example.

Your final consideration when writing a query is the format of the report. For this report, you want to highlight the important information, and you want to leave extra spaces around some items to make them stand out clearly. You can do this with the features that have already been discussed.

The Query:

```
<QA>RUN EXAMP3-2
 **EXAMP3-2

HEADING 'OVERTIME ANALYSIS//DEPT 26804' .
OPEN PERSONNEL .
AUTHORITY FOX .
IF DEPT EQ 26804
    PRINT DEPT,5,LNAME,3,EMPNO,'OVERTIME RATE IS:',1,OT-HRLY .
```

This query sets the heading, opens the PERSONNEL dictionary, and gives the password in the same way as the previous examples. Note that IQL works more efficiently when the format is set up in the first few lines, before any data is actually read. Putting format statements, like HEADING or DATE, before the OPEN statement makes it easier and faster for IQL to write the report.

The new feature introduced in this query is the IF conditional statement. The first line of the IF statement is IF DEPT EQ 26804; this clause contains the condition to be tested. DEPT is the department number, and is defined in the dictionary. EQ is a word from the IQL vocabulary, meaning 'equals'. IQL allows many synonyms for 'equals', such as IS, =, and the full word EQUALS. Other arithmetic relations, such as 'greater than', have similar synonyms. Refer to the **IQL User's Guide** for a complete list of these relations and synonyms.

Because of the first clause, IQL performs the second clause on those records that have a DEPT value equal to 26804. As IQL reads each record from the data file, it determines whether the data in the record satisfies the condition. If the condition is satisfied, then the IQL executes the PRINT clause for that record. Since there is no third clause telling IQL what to do if the record fails to meet the condition, this statement ignores the nonqualifying records.

The PRINT clause illustrates two new features that control the format of the line printed. The numbers in the line indicate how many spaces to leave between two items. Using this feature, the number of spaces can be changed for any interval between items. This is slightly different from the HSPACE statement, which sets the same spacing for every interval. The number 5 between DEPT and LNAME sets the interval to five spaces; the 3 after LNAME resets the spacing to three. After an interval is set, it remains in force until it is changed again.

This line also shows a new use for literals. The literal 'OVERTIME RATE IS:' is inserted directly into the PRINT line, and is printed in the report in each line.

The Report:

```
02/13/79                    OVERTIME ANALYSIS              PAGE    1
                               DEPT 26804

DEPT      LAST              EMPLOY                          OVT-TME
          NAME              NUMBER                          RATE

26804     CHAMPION             8    OVERTIME RATE IS:       7.677
26804     COREOLIS             9    OVERTIME RATE IS:      13.852
26804     FRANK               10    OVERTIME RATE IS:      18.921
26804     MORALES             11    OVERTIME RATE IS:       6.633
26804     O BRIEN             12    OVERTIME RATE IS:       7.482
26804     NELMS               13    OVERTIME RATE IS:      10.560
26804     REILLY              14    OVERTIME RATE IS:      17.871
26804     HALL                15    OVERTIME RATE IS:       8.164

(END QUERY PHASE; PRINT FILE IS QL428ELPT)
```

The IF statement in this query is a single statement composed of two clauses. Notice that there is no period on the line containing the word IF; the period comes at the end of the second clause. The period ends the span of control of the IF statement. EXAMP3-3 shows how the span of control of format statements influences a report.

3.3 SPECIAL FORMS

The previous examples assumed that you had no physical limits on the format of your report. However, the use of special business forms often dictates a particular format. The format statements in the other examples may solve many of the problems encountered when you print a report on special forms. A different kind of problem comes up when you want to organize information into groups, such as in an address label. EXAMP3-3 illustrates a solution to this problem.

To write an address label, you need to single-space the lines within each label, but you must skip some lines before you get to the next label. The default for the vertical spacing of an IQL report is a single space. In EXAMP3-1, you used the VSPACE 2 statement to change the spacing. However, that report used double-spacing for the entire report. In this report you need to change the spacing between records. This section shows you how to do this.

SAMPLE PERSONNEL QUERIES

The Problem:

The Corporate Information department wishes to mail out a newsletter to all employees, and they want you to prepare address labels. They have the blank labels on a special form, which is two labels wide.

The Solution:

The information to put on the label is simply the name and address of the employee. Similarly, the scope requires no concern: every employee will get a label. The format of the report requires more attention, because of the special forms.

There is no need for a heading in this type of report, but you must do more than leave out a HEADING statement. IQL automatically prints part of a heading including the page number and the date, at the top of every page. These special forms are not separated into pages like paper from a line printer, so you do not want page breaks at all. The PAGING OFF statement handles this. This statement prevents the default page break from occurring, and thus prevents the printing of page numbers and dates on each page.

A title over each item is also unnecessary. The TITLES OFF statement prevents the title from printing above the column for the item. It is usually used when a report is set up in a format that does not have the data arranged in columns.

Another new statement you need for this query is the ACROSS statement. ACROSS is particularly useful for special forms, because it forces records to be printed side-by-side a specified number of times, as shown in EXAMP3-3.

In this query you must have a single vertical space between the lines on an individual label, but you must skip several lines to get from one label to the next one. To do this you must change the vertical spacing twice within the query. The first setting should force the extra spacing between labels and the second should set the single spacing within a label. As the query executes, it loops through the VSPACE statements executing each of them once for each record. This concept is explained following the query.

The Query:

```
<QA>RUN EXAMP3-3
 **EXAMP3-3

PAGING OFF . TITLES OFF .
ACROSS 2 .
OPEN PERSONNEL .
AUTHORITY FOX .
VSPACE 3 .
PRINT FNAME,1,INIT,1,LNAME,6 .
VSPACE 1 .
PRINT STREET,14 .
PRINT CITY,14 .
PRINT STATE,20 .
PRINT MAILCODE,29 .
```

Notice that the VSPACE statement is used twice. The first time, it sets the spacing to three. This is done right before the PRINT statement that prints the name of the employee. After the name is

printed, the VSPACE 1 statement is executed, and the vertical spacing changes to one. This setting stays in effect until you change it again. To understand this action more clearly, you must learn about the concept of a stage in a query.

3.3.1 Stages

IQL executes statements normally in the order in which they are written, that is, from top to bottom. In the query, this means that the format is set up first, with the first two lines. Then comes the OPEN statement, followed by the password. In the next statement, the vertical spacing is set to three; then the employee's name is printed. The vertical spacing is reset to one in the next statement, and the rest of the address is printed with this spacing.

When IQL reaches the last statement of a query, it returns to the statement following the OPEN statement and starts to execute on the next record in the file. This loop defines an IQL stage. This query is a single-stage query, with the stage running from the OPEN statement to the end of the query. The statements above the OPEN statement are read only once, at the beginning of the query execution. This explains why you should put the overall formatting statements first: they only need to be executed once.

The vertical spacing cannot be set only once, like the other format statements. At different places within the report you need different VSPACE settings. Since the stage of the query is executed once for each record, the two VSPACE statements will generate two settings for each record.

The operation depends on the position of the statement within the loop. As each new record is read (at the OPEN statement) the spacing is set to three. The NAME item is printed while the vertical spacing is at this setting. The spacing is then changed to one. The next PRINT statement, and the remaining ones in the query, are executed with single spacing. Triple-spacing does not come into effect again until the next record is read. Assuming that three vertical spaces will get the next address onto the next label, the labels will be spaced correctly.

The reason VSPACE 3 is ahead of the first statement instead of after the last, is because of the way the vertical spacing works. Vertical spacing takes place when a PRINT statement is executed. When IQL encounters a PRINT statement, it advances the paper the number of lines indicated by the current setting of the VSPACE parameter, and then prints the items listed in the statement. The VSPACE 3 statement can not be at the end of the query, because this would prevent the first label from being spaced down three lines. The setting would not be 3 until the second record was processed. The second record, however, was printed across from the first. If the query was not set up correctly, the label on the left would start on line one of the page, and the label on the right would print on line three.

ACROSS does not affect the vertical spacing. It forces the information from the second record to be printed alongside the first. To print the labels evenly, you must handle the column alignment of the second address yourself. In this example, the width of the label is assumed to be 35 characters. When you add the item lengths of FNAME, INIT, and LNAME (using the length specified in the dictionary) with the single space between them, you find that this printed line occupies 29 spaces. Therefore, at the end of this PRINT statement you must skip six additional spaces so the name of the next employee will

be printed starting in column 36. If you add up the number of characters in the other PRINT statements, you will find that the blank spaces at the end fill the line out to 35 characters.

The Report:

```
WILLIAM     I BARKER                BETTY       J BROWN
COLORADO A&M COLLEGE                184 MAGNOLIA DR
PHOENIX                             WOBURN
ARIZONA                             MASSACHUSETTS
10946                               74651


CHARLES     L DI VALERA             ANTHONY     F MORANDI
65 RETIREMENT LANE                  25 BARBERSHOP RD
ORLANDO                             BURLINGTON
FLORIDA                             CALIFORNIA
44998                               37760


BRENDA      B BRENNER               ROBERT      A BRETTLER
4 WILSON ROAD                       34 LAFAYETTE RD
GALVESTON                           BALBOA
TEXAS                               PANAMA CANAL Z
51575                               52521


DAVID       M BARNES                HELEN       B CHAMPION
672 EL CAMINO ROAD                  37 CLEARWATER ROAD
FARFAN RIVER                        TAMPA
PANAMA CANAL Z                      FLORIDA
12253                               79150


CORINNA     D COREOLIS              ARTHUR      C FRANK
6226 ASPEN WAY                      88 OILWELL DRIVE
COLUMBUS                            AUSTIN
OHIO                                TEXAS
19800                               84802


ELENA       M MORALES               FREDRICK    N O BRIEN
24 BORDEN ST                        230 TEXTILE MILL
DAYTON                              LOWELL
OHIO                                MASSACHUSETTS
96421                               16880

 END MULTIPLE REPORT PHASE

 REPORTS PRINTED-                    1
 LINES PRINTED-                    125
```

The format statements in this query are useful for determining the physical organization of a report. The logical organization of a report, on the other hand, usually involves reorganization of the records before they are sent to the page. Section 3.4 addresses the problem of the reorganization of data.

3.4 SORTING AND TALLYING

Reports often reflect the data exactly as it is organized in the file. In the previous examples, all the reports have been written this way. However there are many occasions when you may want to reorder the records, or group them into meaningful categories. EXAMP3-4 shows how to group the records into categories, and also shows some of the things you can do when the records have been rearranged.

The Problem:

For upcoming wage negotiations, the Salary Committee requests information about the employees of one of your subsidiary companies, Riverview Mfg. The employees must be separated into two wage categories, hourly and salaried, with a count of employees in each category. Within the categories, the employees should be listed in alphabetical order. The report should indicate the date of the last raise for each employee.

The Solution:

The first thing to determine is the information to include in the report. You want a report containing the name of the employee, his wage category, and the date of his last raise. These are all items defined in the dictionary (CAT is wage category, DATE-LR is the date of the last raise). The total number of people in each category, however, is not included in your file. You must find some way to generate this tally within your report.

This is the first example requiring information not already in your file. The first step in writing a query is to determine what information you want in the report, because this information might not actually be in your data file. If you have to create the data within the query, you must know this before proceeding to any other steps.

Tallying employees within a wage category can best be accomplished if the records are separated into the categories before being counted. This logical reorganization must be done to divide the report into two parts, corresponding to the two wage categories. The wage category (CAT) is therefore the controlling item of this logical organization.

You use the SORT statement to reorganize records from a file. You can sort a file on any item, or items, in the dictionary. The result of a sort is a temporary file that has the records rearranged in ascending ASCII order on the value of the controlling item. (Ascending order for numeric items is 0 through 9; for alphanumeric items, the collating sequence begins with the special characters, proceeds through the numerals, and ends with the letters of the alphabet, A through Z.)

A SORT statement must name the dictionary of the data being sorted and the item(s) that controls the sort. You can use more than one item to control the sort. If you use two controlling items the records are categorized according to the first item, and then they are ordered within the category according to the second item.

Once you have sorted the records, you can use the TALLY statement to count the employees in each category. If you want to tally the number of records in a group, you specify the item that defines the group in the TALLY statement. This item is known as the break item. When the value of the break item changes, the count of all the records in the category that just ended is printed in your report. If you use a TALLY statement, you must be sure that the records are already categorized properly. Only the controlling item of a previous SORT can be used as the break item of a TALLY.

You have now generated the information to include in the report. The next step is to determine the scope of the query. In this example, the report must include the employees of Riverview Mfg. (where the value of CO is 50). You use the IF statement to establish the scope of your query.

Whenever you are sorting a file, and also limiting the scope of your report, place the SORT statement in the second clause of the conditional. Sorting consumes more computer time than most other IQL operations, so you should limit the number of records sorted if this is possible. You use the IF statement to limit the number of records sorted.

A SORT statement also divides the query into stages. The statements ahead of the SORT are performed on every record that is read by the OPEN; the statements after the SORT are performed only on the records that were sorted. Only the records that are selected by the IF statement are sorted, so these are the only records that are used in the rest of the query.

The information and the scope have now been determined. The logical organization has also been set up; only the organization of the page is left. The page format can largely be set up using the IQL default settings. You still must print the data from each category on separate pages. IQL can handle this easily, using two new words: NEWPAGE and NEWGRPV.

NEWPAGE is a one-word IQL statement. Whenever IQL encounters this statement in a query, the report advances to the next page before the next line is printed. In the report you are writing, you want to start a new page when the report moves into a new wage category. This is where you use NEWGRPV. NEWGRPV is shorthand for 'new group value', and you must specify the item that defines the group (again, the item that controls a NEWGRPV should always be a sort key item.) NEWGRPV is meaningful only in the first clause of an IF statement, because NEWGRPV is a test that can be either true or false. When the value of the controlling item changes, the NEWGRPV test is true and the second clause of the IF statement is executed. In this example the second clause is the NEWPAGE statement, so the report continues on a new page when the wage category changes.

The Query:

```
<QA>RUN EXAMP3-4
 **EXAMP3-4

HEADING 'RIVERVIEW MFG. EMPLOYEES//ALPHABETICALLY//WITHIN CATEGORY'.
OPEN PERSONNEL .
AUTHORITY FOX .
IF CO EQ 50
    SORT PERSONNEL BY CAT, NAME .
TALLY NAME BY CAT .
IF NEWGRPV OF CAT THEN NEWPAGE .
PRINT NAME,4,CAT,DATE-LR .
```

After the dictionary is opened, the first operation in this query is the IF statement that controls the SORT. Only employee records that are part of Riverview Mfg. (CO = 50) are sorted. First they are sorted on the wage category; then, within each category, the records are put in alphabetical order by employee name. After this, the only records that the query works with are those records that have been selected and sorted by the IF statement.

The TALLY statement counts the employees. You must specify an item to tally: NAME was chosen because every employee has a name. CAT is the break item of the tally. The first record to pass through the TALLY with a new value in CAT causes the cumulative tally of employees to be printed in the report.

The NEWPAGE statement is also executed when the value of the CAT item changes. Note that the use of an item value as a control is useful only when the records have already been sorted on that item. If you had not sorted the records on CAT, the two wage categories would be mixed together randomly; tallies and new pages would show up every few records. Since you sorted the records by category, all the records for the same category are grouped together.

The Report:

03/07/79 RIVERVIEW MFG. EMPLOYEES PAGE 1
ALPHABETICALLY
WITHIN CATEGORY

FULL NAME			CAT	LAST RAISE YY-MM-DD
ARAOS	LADALIO	P	H	73-06-08
AWASTHI	BIPIN	N	H	73-03-06
BARFIELD	CHARLES	R	H	71-02-05
BRAITHWAITE	ESTELLE	U	H	71-03-26
BROWN	BETTY	J	H	72-03-20
CHAMPION	HELEN	B	H	71-03-26
FEMINO	SALVATORE	H	H	71-04-24
FORBES	JOAN	E	H	72-02-08
FRANK	ARTHUR	C	H	71-07-21
GOERNER	DUVALL	W	H	73-05-27
MANFIELD	MELISSA	R	H	72-04-09
MCNATT	BELINDA	B	H	72-07-03
MEYERS	MELINDA	L	H	73-04-22
MORALES	ELENA	M	H	73-03-19
NELMS	KEVIN	S	H	73-04-22
NESBIT	DAN	F	H	73-04-05
PAYNE	PHILIP	Z	H	71-07-20
REILLY	SEAN	K	H	73-06-23
RYAN	DEBBIE	P	H	73-03-10
SCHLAFKE	NANCY	W	H	71-07-28
ZIMMERMANN	DAVID	A	H	71-02-02

CAT H FULL NAME TALLY: 21

03/07/79 RIVERVIEW MFG. EMPLOYEES PAGE 2
ALPHABETICALLY
WITHIN CATEGORY

FULL NAME			CAT	LAST RAISE YY-MM-DD
AARNASEN	GEIRMUNDERA		S	71-07-06
BALLARD	PHYLLIS	S	S	73-03-30
BARKER	WILLIAM	I	S	71-07-27
BARNES	DAVID	M	S	73-03-12
BRENNER	BRENDA	B	S	71-03-09
BRETTLER	ROBERT	A	S	71-04-19
COREOLIS	CORINNA	D	S	72-07-04
CRUZ	HENRY	Z	S	71-04-05
DI VALERA	CHARLES	L	S	72-05-03
EVERINGHAM	LLOYD	A	S	72-04-27
FALK	MARY	R	S	72-02-05
FORBES	JAMES	H	S	73-04-14
GODBY	CLAIR	G	S	71-03-29
HALL	RICHARD	B	S	71-04-08
JARAMILLO	CARLOS	A	S	71-07-30
KOURANEY	OSCAR	P	S	71-04-31
LEE	LAUREN		S	72-07-31
MELZER	WILLIAM	M	S	71-02-03
MORANDI	ANTHONY	F	S	71-05-21
MUCKELROY	ROBERT	H	S	73-08-09
O BRIEN	FREDRICK	N	S	73-04-06
PRATT	PEARL	B	S	72-06-06
SIMMS	MARY	I	S	73-03-17
SLOUGH	SELENA	S	S	73-03-16
SMITH	ROBERT	H	S	72-06-24
TARR	RITA	M	S	72-05-26
WAGNER	PAUL	A	S	71-03-26
WEBSTER	WANDA	W	S	71-06-27
ZARZEWSKA	IRENE	G	S	72-05-15

CAT S FULL NAME TALLY: 29

(END QUERY PHASE; PRINT FILE IS QL211ELPT)

You must write the query statements in the proper order. Remember that TALLY and NEWPAGE statements look for a new value in the CAT item. This occurs when IQL processes the Aarnasen record. When this record passes through the TALLY statement, IQL immediately prints the tally of the previous group. This tally prints at the bottom of the first page. Once the tally is printed, the IF NEWGRPV condition is tested; since S (salaried) is a 'new group value' for CAT, the NEWPAGE clause is executed. Finally, the Aarnasen record passes through the PRINT statement, and the desired information is printed in the right place.

The line in the report containing the tally information identifies the item that controls the tally. To generate this line, IQL reads the top and bottom titles from the dictionary. Since NAME was tallied by CAT, the tally line also tells you which category has been tallied (S or H). EXAMP3-5 shows many other summaries that you can generate within a query.

3.5 SUMMARIZING

Reports often require summaries. The query in EXAMP3-4 uses a particular kind of summary, TALLY, which counts the number of records that the query processes. Other kinds of summaries, notably those that handle numeric operations, are also available in IQL. This section introduces some of them.

The Problem:

The Salary Committee wants some specific salary data for the Inorganic Division of Riverview Mfg. For each employee, they need the salary and the date of hire. They need figures on the money being spent on salaries and the average salaries, for the whole division and for each department within the division.

The Solution:

In many ways, this problem is similar to the preceding one. You can select the records for the Inorganic Division (those with the value of DIV equal to 470) and sort them into groups by department. However, two new summary statements are necessary to generate the information requested. These are TOTAL and AVERAGE. Also, you must keep track of two summaries. The scope of one summary is the entire division, and the scope of the other is a single department. This involves a slightly different use of the summarizing statements.

TOTAL and AVERAGE operate like TALLY, but they can be used only on numeric items (TALLY can be used on any type of item). If you assign a break item to these statements, as you did in EXAMP3-4, the summaries are printed when the value of the break item changes. Since you need a breakdown of averages and totals by department, DEPT must be the break item for these interim summaries.

This report requires overall summaries, in addition to the interim ones. To generate summaries over the entire report, use a summarizing statement with no controlling item. You still must name the item that you want summarized, but omit the phrase 'BY item'. When no controlling item is specified, IQL generates an overall summation and prints it at the end of the report.

The Query:

```
<QA>RUN EXAMP3-5
 **EXAMP3-5

HEADING 'SALARY ANALYSIS//RIVERVIEW MFG.//INORGANIC DIVISION'.
OPEN PERSONNEL.
AUTHORITY TIGER.
IF CO = 50 AND DIV = 470
    THEN SORT PERSONNEL BY DESCENDING DEPT, SALARY .
TALLY NAME BY DEPT .
TOTAL SALARY BY DEPT .
AVERAGE SALARY BY DEPT .
TALLY NAME .
TOTAL SALARY .
AVERAGE SALARY .
PRINT DIV,DEPT,LNAME,EMPNO,SALARY,DATE-HR.
```

The IF statement in this query contains two options that have not yet been discussed. AND is allowed in any conditional clause in an IF statement. Other logical operands such as NOT and OR are also allowed. Refer to the **IQL User's Guide** for rules governing their use. The other new option is in the SORT phrase, where DESCENDING is specified. Normally, items are sorted in ascending order. This order can be reversed using the DESCENDING modifier, so you can list numbers with the highest value first.

Although the report does not require a count of employees, this query includes the TALLY statement so that you can see the similarity among all the summarizing statements. The first TALLY statement in this query is the same as the one from the preceding query, except that the tally is controlled by DEPT instead of CAT. The department value also controls the AVERAGE and TOTAL statements. Thus, at the end of each group of employees in a single department the report will contain the count of employees, the average salary, and the total salary figure for that department.

The second TALLY statement, and the TOTAL and AVERAGE statements which follow it, do not have a controlling item. These three statements produce the summaries for the entire company that are printed at the end of the report.

The Report:

03/08/79 SALARY ANALYSIS PAGE 1
RIVERVIEW MFG.
INORGANIC DIVISION

DIV	DEPT	LAST NAME	EMPLOY NUMBER	SALARY WEEKLY	DATE HIRED	
470	27264	AARNASEN	48	516.20	65-07-17	
470	27264	LEE	49	559.72	41-07-18	
DEPT 27264 FULL NAME				TALLY:		2
DEPT 27264 SALARY WEEKLY				TOTAL:		1075.92
DEPT 27264 SALARY WEEKLY				AVG:		537.96
470	27172	MCNATT	40	561.48	69-07-18	
470	27172	SMITH	44	445.36	48-06-14	
470	27172	FEMINO	46	327.60	47-04-11	
470	27172	FORBES	42	148.88	64-02-05	
470	27172	FORBES	43	271.44	59-04-09	
470	27172	WAGNER	45	229.84	45-03-08	
470	27172	BRAITHWAITE	47	218.28	45-03-07	
470	27172	FALK	41	150.36	67-02-05	
DEPT 27172 FULL NAME				TALLY:		8
DEPT 27172 SALARY WEEKLY				TOTAL:		2353.24
DEPT 27172 SALARY WEEKLY				AVG:		294.16
470	27080	PAYNE	36	539.68	51-07-17	
470	27080	MANFIELD	34	268.76	63-04-09	
470	27080	SIMMS	32	198.04	56-03-07	
470	27080	ZIMMERMANN	38	136.40	69-02-04	
470	27080	SCHLAFKE	33	514.84	43-07-16	
470	27080	WEBSTER	35	437.48	44-06-14	
470	27080	GOERNER	37	394.40	46-05-13	
470	27080	NESBIT	39	295.36	68-04-10	
DEPT 27080 FULL NAME				TALLY:		8
DEPT 27080 SALARY WEEKLY				TOTAL:		2784.96
DEPT 27080 SALARY WEEKLY				AVG:		348.12
470	26988	PRATT	28	467.48	66-06-15	
470	26988	CRUZ	26	300.96	66-04-10	
470	26988	BARFIELD	30	123.04	66-02-04	
470	26988	MUCKELROY	31	592.20	64-08-19	
470	26988	KOURANEY	29	306.20	40-04-10	
470	26988	BALLARD	25	188.12	43-03-06	
470	26988	SLOUGH	27	179.24	57-03-06	
DEPT 26988 FULL NAME				TALLY:		7
DEPT 26988 SALARY WEEKLY				TOTAL:		2157.24
DEPT 26988 SALARY WEEKLY				AVG:		308.18

03/08/79 SALARY ANALYSIS PAGE 2
RIVERVIEW MFG.
INORGANIC DIVISION

OVERALL FULL NAME	TALLY:	25
OVERALL SALARY WEEKLY	TOTAL:	8371.36
OVERALL SALARY WEEKLY	AVG:	334.85

(END QUERY PHASE; PRINT FILE IS QL312ELPT)

This query, like the one that preceded it, uses a SORT statement to organize the records before it begins to summarize them. The order of the statements is also important in this query, just as it has been in the other queries in this chapter. Chapter 4 shows how you can alter the order of execution within the query, to complement the natural order of execution of IQL. It also shows how you can generate data in addition to the summaries that IQL provides.

3.6 REVIEW

In this chapter, you have learned the fundamental rules for writing a query, and have seen the interactions among many of the basic IQL statements. Now you can review some of the basic techniques you have learned.

When writing a query, the first thing you should do is determine what information you need to include in your report. This information is usually in your data file and your dictionary. When you need data that is not part of your file, you must decide how to create it before you proceed.

Once you have determined the information you want, you should determine which records you want. This is what has been referred to as the scope of the query. You can control the scope of the query, or of any single statement within the query, with the IF statement.

The third step in preparing a query is to determine the format of the report. You may need to reorder the records, using a SORT statement before you extract the data for printing. Finally you set up the page layout using the page formatting statements, such as LMARGIN and VSPACE.

The three steps above are in reverse order from the order in which you actually write the query. When you write the query, you should put the global format statements (those that control the whole query) at the top, followed by the statements that control the scope of the query, and finally you write the statements that produce the output such as PRINT and TALLY.

This brings up the notion of stages. The global format statements come before the OPEN statement, which initiates the first stage. In the first stage, you should include any statements that are needed to define the scope of the query. Assuming that you are sorting the file, you should perform the SORT after the scope has been defined. Below the SORT, you enter the second stage of the query. In this stage, you should include all the statements that actually generate the information you want in your report.

You may find these recommendations useful when writing queries. However, you often have to make allowances for special considerations. In EXAMP3-3, for instance, you had to imbed format statements within the second stage so that they could be changed while the records were being read. These local format statements must be placed within the query itself, so that they are executed each time a record is processed.

A query executes from the top down within each stage. On the first pass, the global format statements are processed. When IQL encounters the OPEN statement, the query checks the dictionary and opens the data file named in it. The first record in the file is read, and the query moves on to the next statement. IQL processes this record through all the statements that precede a SORT statement; if this is one of the

records that is to be sorted, it is dropped into a temporary file. Since the first stage ends at the SORT, IQL returns to the OPEN statement. The next record from the data file is read and processed, and passed to the sort. This loop continues until all the records in the data file have been read.

Once the original file has been exhausted, IQL begins the second stage if there is one. The temporary file is sorted according to the controlling items, and IQL uses the sorted file as the input to the next stage. As the new stage begins, the first record from the sorted file is read and processed in the same fashion as the first stage.

You must take account of the order of execution if you are going to control formats within the query, as illustrated in EXAMP3-3. When you change a format setting during execution, you must explicitly change it back again if the second record is to be processed the same way as the first. To determine the correct placement of a statement you must know when you want it to take effect, and what the effect will be. Refer to the **IQL User's Guide** for the discussion of each statement.

CHAPTER 4

SAMPLE MARKETING QUERIES

This chapter describes the use of IQL in a marketing application. The queries here are not intended to show every use that a marketing department could make of IQL. They illustrate some of the more advanced features of IQL that can be used in any application.

The analysis following each query explains the new features that are introduced in the query. The statements and concepts introduced in Chapter 2 are not explained further, although some are used in new ways. Among the new features shown for the first time in this chapter are the GO TO statement, the FIND statement, the COMPUTE statement, and using multiple dictionaries in a single query.

The first queries address the CUSTOMERS dictionary. The next few address the SALESMEN dictionary, and the final group addresses both dictionaries. The CUSTOMERS dictionary is shown here:

```
<QA>ITEMS CUSTOMERS

DICT        FILE    FILE-IN                    REC  BLK  KEY KY KY RD CP RW
NAME        TYPE    NAME        DIRECT         LEN  FAC  LOC LN TP PW PW PW
----        ----    ----        ------         ---  ---  --- -- -- -- -- --

CUSTOMERS   SQ DSK7 CUSTMRSEQ                  145    0    0  0

   ITEM     TOP          BOTTOM      1ST  NO.  T S PRINTING        SCAN
ID NAME     TITLE        TITLE       CHAR CHAR Y C PICTURE         GNNS PT
-- -------- ---------- ---------- ---- ---- - - -------------- ---- --

DD CUSTNO   CUSTOMER'S VENDOR NO.     1    5 N 0 SZZZZ
DD CNAME    CUSTOMER     NAME         6   30 A 0
DD BUYER    BUYER        NAME        36   26 A 0
DD STREET   CUST         STREET      62   25 A 0
DD CITY     CUST         CITY        87   25 A 0
DD STATE    CUST         ST         112    2 A 0
DD ZIP      CUST         ZIP        114    5 N 0 SZZZZ9
DD CYSALES  YR TO DATE   PURCHASES  119    9 N 2 $$,$$$,$$$.$$
DD CYPAID   Y-T-D        PAID       128    9 N 2 $$,$$$,$$$.$$
DD CLIMIT   CREDIT       LIMIT      137    9 N 2 $$,$$$,$$$.$$
DD SLSMAN   SALESMAN     NUMBER       2    4 N 0 SZZZ9

 (END LIST OF ITEMS)
```

4.1 VARIABLES

In EXAMP3-4 and EXAMP3-5, you needed information in your report that did not appear in your data file. To obtain this information, you used IQL's ability to tally information during the execution of a query. This section shows you how to use IQL to do more general computations. You can perform many kinds of computations and use the results within the execution of the query.

The Problem:

The Credit Department requires regularly updated reports about customers who are at, or have exceeded their credit limit. The report must contain the amount of purchases, the amount paid so far this year, and the credit limit. You should list the customers with the highest credit limit first.

The Solution:

By looking at the dictionary, you can determine that the information the report must contain is in your data file. The amount of purchases for the year is in the data item CYSALES, the amount paid is in CYPAID, and the credit limit is in CLIMIT. These items, and the name of the customer, are the only entries the report should contain.

When you consider the scope of the report, however, you can see that there is no single item that identifies customers who are over their credit limit this year. But the outstanding balance can be obtained by finding the difference between the amount of purchases and the amount of payment. The result of this simple arithmetic operation can then be compared against the credit limit. The report can thus be limited to those customers who have exceeded their limit.

You must store the result of the computation in a variable so that you can work with it. In IQL, you declare a variable as soon as you use it; you do not have to make any other provisions. Numeric variables always start with an X, as shown in EXAMP4-1. Whenever IQL encounters a word that begins with an X inside a query, it knows that this is a variable (you cannot define an item name that begins with an X, because the DEFINE command looks for this and tells you that it is illegal).

Now that you have defined a storage variable, you need to perform the computation. You use the COMPUTE statement to tell IQL that you are doing arithmetic. The items named in a COMPUTE statement must be defined as numeric items, and you must leave spaces around your arithmetic operators (+, -, *, and /).

The Query:

```
<QA>RUN EXAMP4-1
 **EXAMP4-1

HEADING 'LIST OF CUSTOMERS OVER CREDIT'.
OPEN CUSTOMERS.
COMPUTE X = CYSALES - CYPAID .
IF X GEQ CLIMIT
    SORT CUSTOMERS BY DESCENDING CLIMIT.
PRINT CNAME,CYSALES,CYPAID,CLIMIT.
```

The query is very straightforward. The computation is performed following the HEADING and OPEN statements. The name of the variable

in this example is simply "X". For each record processed, X contains the outstanding balance. Then, if the outstanding balance is greater than the credit limit, the record is passed on to be sorted. The records are printed in descending order of the credit limit.

The Report:

02/13/79 LIST OF CUSTOMERS OVER CREDIT PAGE 1

CUSTOMER NAME	YR TO DATE PURCHASES	Y-T-D PAID	CREDIT LIMIT
BARBAZON HOTELS, INC	$100,000.00	$75,000.00	$25,000.00
WILDCAT OIL INC.	$27,211.25	$18,711.00	$5,000.00
GALACTIC EXPLORATION INC.	$21,347.98	$18,112.00	$3,000.00
FLORENTINE TILE SERVICE	$7,500.00	$3,000.00	$3,000.00
DEL RAY PROPERTIES, INC	$5,000.00	$.00	$2,500.00
BRABSTEIN MOTOR HOMES INC.	$8,694.50	$2,119.80	$2,500.00
WESTERN BROASTED CHICKEN INC	$4,500.00	$3,000.00	$1,500.00
F & F INC.	$9,000.00	$162.00	$1,000.00
LAKESIDE HOMES, INC	$1,211.00	$.00	$500.00
SOLVANG SAILS, INC	$25.00	$25.00	$.00
GARLAND TRAVEL TOURS	$45.00	$45.00	$.00
REALIZATION STUDIOS, INC	$1,000.00	$500.00	$.00

(END QUERY PHASE; PRINT FILE IS QL128ELPT)

4.2 ALTERING THE ORDER OF EXECUTION

Chapter 2 explains stages in a query. The SORT statement separates a query, so that the statements above it are executed on all records read by the OPEN statement, and all statements below it are executed on those records that are sorted. This ability to execute certain statements on a select group of records is useful, but using SORT is time-consuming and often unnecessary. The GO TO statement provides a method for altering the order of execution of a query without terminating the current stage of the query.

The Problem:

The Marketing department is analyzing a strategy for generating increased revenue. Because this revenue will be used to offset the added costs of heating factories in New England, the proposed strategy increases prices by 10 percent to customers in the New England states. You must prepare a report on the increased revenue, showing each customer's current purchases and the projection of purchases after the increase.

The Solution:

Your report must include the following pieces of information: the customer's name, the amount of purchases made this year, the cost of those purchases at the proposed higher price, and the increase in revenue. Summary totals of these three items should also be included.

CYSALES contains the amount of purchases so far this year, but the other two pieces of information are not in your file. You must create them within the query, using numeric variables.

Having determined the information you need, you must now consider the scope of the report. In this example you want to limit the report to customers in the northern New England states. The IF conditional statement is used to limit the scope to these records. You must also determine what to do with these records in the second clause of the IF statement.

The GO TO statement transfers the execution of the query to a statement you specify. You specify the statement with a two digit number. In EXAMP4-2, you need to transfer control to the COMPUTE statement. Therefore, you label the COMPUTE statement with a 10 in the first two columns; when the query executes the GO TO 10 statement, control passes to the statement labeled 10.

An IF statement does not eliminate records from the query, but merely executes the second clause on those records that satisfy the condition of the first clause. For example, a customer record from Ohio would fail to satisfy the condition that it is from northern New England, but it would still be passed to the COMPUTE statement in the normal execution of the query.

To make the IF statement an effective screen, you must use another GO TO statement in the third clause of the IF statement. You use a third clause in an IF statement to specify an operation to perform on those records that do not satisfy the test condition. This clause is introduced by the word ELSE. The statements following the word ELSE are executed only on those records that do not pass the test condition.

This report should contain information only about customers in northern New England. Thus, the query should proceed immediately to the next record if the current record does not qualify. You can do this with another form of the GO TO statement, GO TO NR (NR is shorthand for Next Record). This form of the statement tells IQL that you want to ignore the current record and immediately begin to process the next one. Control of the query passes back to the first statement following the OPEN (or to the top of the current stage of the query).

The scope of the query is thus limited to those customers from the northern New England states. In the query itself, there are some additional statements that help to format the report. These are discussed in the analysis following the query.

The Query:

```
<QA>RUN EXAMP4-2
 **EXAMP4-2

  HEADING 'ANALYSIS OF//PRICE CHANGE//IMPACT' .
  TITLES XSALES = 'NEW PRICE//SALES', XDIFF = 'INCREASED//SALES' .
  PICTURE XSALES = 'ZZZ,ZZZ.99', XDIFF = 'ZZZ,ZZZ.99' .
  OPEN CUSTOMERS .
  IF STATE EQ 'NH','VT','MA','ME'
      THEN GO TO 10
      ELSE GO TO NR .
10COMPUTE XSALES = CYSALES * 1.10 .
  COMPUTE XDIFF = XSALES - CYSALES .
  TOTAL CYSALES . TOTAL XSALES . TOTAL XDIFF .
  PRINT CNAME,1,CYSALES,XSALES,XDIFF,STATE .
```

This query contains some formatting features that have not yet been discussed. First, look at the TITLES and PICTURE statements. These refer to the working variables, XSALES and XDIFF, which you use later in the query. Remember that a variable name must start with X, and can be up to eight characters long.

This TITLES statement has a slightly different form than you have seen before, because it sets up two titles with a single statement. Note that if you do not assign a title to a working variable the report uses the name of the variable as the column title.

The PICTURE statement is useful when you wish to print working variables in a specific format. IQL prints the variable item in the format you specify with the PICTURE statement. The rules for the use of pictures are the same in IQL as they are in COBOL. Refer to the **IQL User's Guide** for the complete set of rules. For the variables in this query, the pictures specify that there are two decimal places (.99) and up to six numerals with the leading zeros suppressed (ZZZ,ZZZ).

The operation of the query proceeds as follows. Once the format is set up and the dictionary is opened, the conditional statement is executed on the record. This conditional shows a new kind of test. The value of the STATE item is tested against the four literals, NH, VT, MA, and ME. If the value in the item matches any of these codes (separating the literals with commas implies OR), then the condition is true and the second clause is executed for this record. Otherwise, the third clause is executed. The third clause must always be prefaced by the word ELSE. In this query, the second clause is prefaced by the word THEN. THEN is optional, but it is included here to improve clarity. The indentation of the THEN and ELSE clauses is also optional, and is used in this manual to improve readability.

Note that the label 10 is in the first two columns of the text, and that all the other lines are indented two spaces. A label, such as 10, must fall in these first two columns, but the text of a statement can start anywhere on the line. All the statements in this query start in the third column of text.

The statement labeled 10 is the COMPUTE statement. This statement is executed only on those records that pass the test of the IF statement, because all other records force IQL to proceed to next record in the file. The records of all customers from the northern New England states are passed to the two COMPUTE statements, which generate the data you need for the report.

The Report:

03/28/79 ANALYSIS OF PRICE CHANGE IMPACT PAGE 1

CUSTOMER NAME	YR TO DATE PURCHASES	NEW PRICE SALES	INCREASED SALES	CU ST
SCARLOTTI VIOLIN CO.	$1,233.67	1,357.03	123.36	MA
GLOBALEX	$2,500.00	2,750.00	250.00	MA
LAKESIDE HOMES, INC	$1,211.00	1,332.10	121.10	NH
LITCHFIELD MOTORS	$9,000.00	9,900.00	900.00	MA
HUBERT OIL COMPANY	$600.00	660.00	60.00	NH
BEE DRILL SERVICE	$1,000.00	1,100.00	100.00	ME
INVESTMENTINC.	$364.75	401.22	36.47	MA
ENERGY RESOURCES INC.	$8,000.00	8,800.00	800.00	VT
FISHERIES OF N.E. INC.	$9,164.00	10,080.40	916.40	MA
MERRIMACK VALLEY SALES	$6,000.00	6,600.00	600.00	MA
APPLE ORCHARDS INC.	$671.00	738.10	67.10	NH
CRAIG PUBLISHING	$5,115.50	5,627.05	511.55	MA
JOHN JONES	$2,324.79	2,557.26	232.47	MA

OVERALL YR TO DATE PURCHASES	TOTAL:	$371,640.15
OVERALL NEW PRICE SALES	TOTAL:	408,804.17
OVERALL INCREASED SALES	TOTAL:	37,164.02

(END QUERY PHASE; PRINT FILE IS QL140ELPT)

4.3 MULTIPLE REPORTS

This section shows you how multiple reports are generated in a single pass through the file. The multiple report feature of IQL is useful when you need more than one report based on the same files. The reports need not contain the same data, or even the same records. The multiple report feature can help you save computing time because it is always more efficient to deal with each record only once.

The Problem:

You must verify the data in your customer files. To do so, you must print every item in each record. However, this would produce too long a line for your output device. To circumvent this problem, you can print certain portions of the record on one page, then the rest on subsequent pages.

The Solution:

You must use the multiple report feature because a single report will not fit on a single page. The multiple report feature allows IQL to read a record once, and write information to more than one report.

You use the REPORT statement to create a multiple report query. The statements following REPORT 1 comprise the first report, the statements following REPORT 2 comprise the second report, and so on.

REPORT places no constraints on what you can do in each segment of the query. You can specify different formats for each report but you must

account for the fact that a format statement stays in effect until it is changed. Following each REPORT statement you must specify the format for the current report. EXAMP4-3 shows the basic layout of a multiple report query. Section 4.4 explains how the format can be changed between reports.

The Query:

```
<QA>RUN EXAMP4-3
 **EXAMP4-3

OPEN CUSTOMERS.
REPORT 1 .
HEADING 'CUSTOMER DATA//LIST NO. 1'.
PRINT CUSTNO,1,CNAME,BUYER .
REPORT 2 .
HEADING 'CUSTOMER DATA//LIST NO. 2' .
PRINT CUSTNO,1,STREET,CITY,STATE,ZIP .
REPORT 3 .
HEADING 'CUSTOMER DATA//LIST NO. 3' .
PRINT CUSTNO,1,CYSALES,CYPAID,CLIMIT .
```

You can think of the REPORT statement as a switch that sets the destination of the output. Statements that generate output include PRINT, HEADING, and the TALLY class statements. IQL sends the output to an area on disk in a file created and named by IQL. This continues until IQL encounters the REPORT 2 statement. At this point, IQL sets up a new destination and starts sending the output there. REPORT 3 has the same effect. After IQL completes processing the current record, control passes back to the top of the stage. In this query, the top of the stage is the REPORT 1 statement. With this statement, the destination is set back to the original disk file and the processing of the next record begins.

The Reports:

```
02/13/79                        CUSTOMER DATA                    PAGE     1
                                  LIST NO. 1

CUSTOMER'S   CUSTOMER                              BUYER
VENDOR NO.   NAME                                 NAME

     1 SCARLOTTI VIOLIN CO.                  ANTHONY SCARLOTTI
     2 WILDCAT OIL INC.                      ROBERT LEE BALLARD
     3 BARBAZON HOTELS, INC                  RICHARD J. HOLLISTER
     4 MILITARY SUPPLY INC.                  LT COL HARLAND BIRD
     5 FORESTMAN NURSERIES                   ELROND WILLIAMS, JR.
     6 DEL RAY PROPERTIES, INC               PETER VAN ALLSTEIN
     7 CONSOLIDATED MARINAS, INC             THEODORE M HOOK
     8 SOLVANG SAILS, INC                    STANFORD ELIASSEN
     9 NAVAJO PRODUCTIONS                    RICHARD LITTLE BEAR
    10 SONJA L MARTIN INC                    SONJA L MARTIN
    11 GALACTIC EXPLORATION INC.             JOHN ARMSTRONG
    12 INTERNATIONAL MOTORS CO.              WILLIAM OROURKE
    13 MARVELL MOTORCYCLES INC               EVEL HONDA JR.
    14 WESTERN BROASTED CHICKEN INC          COL. HARLAND V N JACKSON
    15 ELECTRONICS CONCEPTS, INC             HORIGAN HAMILTON
    16 SCOTTISH HABERDASHERS LTD             IAN MACGILVERAY III
```

```
 17 GLOBALEX                          EDWARD W NADEAU
 18 LAKESIDE HOMES, INC               JEROME SILAS
 19 GARLAND TRAVEL TOURS              GLORIA G GARLAND

(END QUERY PHASE; PRINT FILE IS QL328ELPT)
```

02/15/79 CUSTOMER DATA PAGE 1
LIST NO. 2

1	233 COMMONWEALTH AVE.	BOSTON	MA	10872
2	43 MAIN STREET	EL PASO	TX	43211
3	BARBAZON TOWERS	CHICAGO	IL	43996
4	432 DALE MABRY HIGHWAY	TAMPA	FL	23560
5	911 MENASHA ST	APPLETON	WI	44074
6	1 MALIBU WAY	MALIBU	CA	87460
7	HARBOR ISLAND	SAN DIEGO	CA	87211
8	SOLVANG BLD	SOLVANG	CA	87101
9	TRIBAL CHAMBERS	NAVAJO RESERVATION NORTH	OK	67453
10	43 MADISON AVE	NEW YORK	NY	23111
11	47 ORBITAL WAY	HOUSTON	TX	34001
12	PONTIAC TURNPIKE	PONTIAC	MI	47345
13	4 MECHANICS ROAD	PIERRE	SD	76093
14	WESTERN TOWERS	NASHVILLE	TN	33091
15	12111 EL CAMINO REAL	PALO ALTO	CA	88666
16	5 AVE OF THE AMERICAS	NEW YORK	NY	21333
17	23 BROADWAY	LAWRENCE	MA	18037
18	WINNEPESAUKEE BLDG	MELVIN VILLAGE	NH	03886
19	211 JOPPA RD	BALTIMORE	MD	21133

02/15/79 CUSTOMER DATA PAGE 1
LIST NO. 3

1	$1,233.67	$867.23	$1,000.00
2	$27,211.25	$18,711.00	$5,000.00
3	$100,000.00	$75,000.00	$25,000.00
4	$233.50	$233.50	$1,000.00
5	$4,678.99	$4,100.00	$1,000.00
6	$5,000.00	$.00	$2,500.00
7	$1,250.47	$1,250.47	$2,500.00
8	$25.00	$25.00	$.00
9	$111,111.11	$111,111.11	$10,000.00
10	$467.75	$333.00	$210,000.00
11	$21,347.98	$18,112.00	$3,000.00
12	$22,775.00	$19,870.00	$30,000.00
13	$1,000.00	$1,000.00	$1,000.00
14	$4,500.00	$3,000.00	$1,500.00
15	$8,788.00	$8,688.00	$1,500.00
16	$2,544.00	$1,772.00	$2,000.00
17	$2,500.00	$2,500.00	$2,500.00
18	$1,211.00	$.00	$500.00
19	$45.00	$45.00	$.00

```
END MULTIPLE REPORT PHASE

REPORTS PRINTED-                2
LINES PRINTED-                184
```

Note that the reports shown here are facsimiles of the terminal session you experience when you run the query. At the terminal, you receive two messages during a multiple report query. The first is the standard message, END OF QUERY PHASE, telling you where the print file is located. Another message, END MULTIPLE REPORT PHASE, appears after the final report.

A multiple report query produces two print files. One contains the first report, and is identified in the first message. The second file contains all subsequent reports (in this example, reports 2 and 3 are in the same disk file). You may print the subsequent reports together as one file (the pages will of course be separated by the line printer), or you may separate the reports into distinct files using a text editor.

4.4 REPORT FORMATS

The reports of a multiple report query can have different formats. The format of each report must be set up immediately following the REPORT statement. There is no limitation on the variety of formats you can design within a single query. EXAMP4-4 demonstrates how you can change the format between reports.

The Problem:

You need to produce two monthly reports based on your customer data file. One report is the summary, by state, of year-to-date sales. The other is a set of mailing labels for all your customers.

The Solution:

This problem can be solved using the techniques covered earlier in this manual. You will need to make some of the format statements explicit, however.

When writing a multiple report query, the first thing to do is determine what each query would contain if written separately. From this you can determine which statements and format settings they have in common. At the very least, they should have the same OPEN statement. In this problem, the two queries have another statement in common; both reports must be sorted by state.

The statements that apply to both queries are written first, before the REPORT 1 statement. This procedure is different from the way other queries are written, because in other queries you put most of your format statements ahead of the OPEN statement. Generally, you should put global format statements at the beginning of the query; those that have limited span of control should come at the appropriate place later in the query.

Thus, the first two statements in this query should be the OPEN and SORT statements. You write the REPORT 1 statement next, and then the format statements that control the first report. When you are done with all the statements that apply to Report 1, you should define Report 2. The new format should be set up for the new report, followed by the statements that execute the report.

The Reports:

```
03/28/79                        YTD PURCHASES                        PAGE  1
                                  BY STATE

CUSTOMER                                    YR TO DATE
NAME                                         PURCHASES

COOK INC.                                      $164.00

CUST ST AR YR TO DATE PURCHASES                 TOTAL:                $164.00

HAWTHORNE PUBLISHING CO.                        $87.50
DESERT PROPERTIES                            $4,567.89

CUST ST AZ YR TO DATE PURCHASES                 TOTAL:              $4,655.39

SOLVANG SAILS, INC                              $25.00
CONSOLIDATED MARINAS, INC                    $1,250.47
DEL RAY PROPERTIES, INC                      $5,000.00
FIBERGLASS ASSOCIATES                       $47,000.00
ELECTRONICS CONCEPTS, INC                    $8,788.00
HAMPSTEAD,INC.                              $27,000.00
JOSEPHS INC.                                 $9,950.00

CUST ST CA YR TO DATE PURCHASES                 TOTAL:              $99,013.47

(END QUERY PHASE; PRINT FILE IS QL340ELPT)
```

```
COOK INC.
BRUCE JOHNSON
181 MESSINA
LITTLE ROCK
AR      72201

HAWTHORNE PUBLISHING CO.
MURIEL DEVONSHIRE
67 EAST 80TH
PHOENIX
AZ       8503

SOLVANG SAILS, INC
STANFORD ELIASSEN
SOLVANG BLD
SOLVANG
CA      87101

CONSOLIDATED MARINAS, INC
THEODORE M HOOK
HARBOR ISLAND
SAN DIEGO
CA      87211

DEL RAY PROPERTIES, INC
PETER VAN ALLSTEIN
1 MALIBU WAY
MALIBU
CA      87460

FIBERGLASS ASSOCIATES
JAMES T. SLOAN
SLOAT WAY
MONTEREY
CA      87699

HAMPSTEAD,INC.
TED STEPHEN
37 DINSMORE
HAWTHORNE
CA      94620

JOSEPHS INC.
GEORGE RAMSBOTTOM
1885 PACIFIC AVE.
LOS ANGELES
CA      95441

 REPORTS PRINTED-                2
 LINES PRINTED-                174
```

4.6 MULTIPLE BREAK ITEMS

Many reports require summaries of small groups within larger groups. To create these summaries, it is not enough to use the smaller group as a break item. A problem would arise if the name of the last small group in one region was the same as the name of the first small group in the next region. Because the name itself does not change, IQL would not know that this in fact was two separate groups. Summaries can be written with more than one break item so summary will break when either item value changes.

The Problem:

You must prepare a report on the performance of a certain group of salesmen for each region and district in the western states. You are interested in the low-quota salesmen, and the report should reflect the performance of all these salesmen in each region and in each district within a region. The report must include the total and average sales in each group.

The Solution:

The group of salesmen is limited to those from the western states (where SREGION is 6 or higher) who have a sales quota of $250,000 or less. This group of salesmen must be sorted by their region, and by the district within the region. Your query therefore must have a SORT statement controlled by an IF statement in the first stage. The second stage contains the statements that generate the report.

These statements must include a TOTAL and an AVERAGE statement. Since you must generate the total and average for both the region and the district, these two statements should have two break items. When writing a summary statement with more than one break item, you must write the major group first, followed by the minor one. This is the same order of items as in a SORT statement. A region is a major group in this query, and the districts are minor groups.

You can separate the total summaries from the average summaries using REPORT statements. Notice that in EXAMP4-5, REPORT 1 appears twice. IQL allows you to redirect information from one report to another at any point in the query.

The Query:

```
<QA>RUN EXAMP4-5
 **EXAMP4-5

HEADING 'SALES PERFORMANCE//LOW QUOTA SALESMEN//WESTERN REGION' .
OPEN SALESMEN .
IF SREGION GR 5 AND SQUOTA LEQ 250000
   SORT SALESMEN BY SREGION, SDIST .
REPORT 1 .
TOTAL SSALES BY SREGION, SDIST .
IF NEWGRPV OF SREGION PRINT ' ' .
REPORT 2 .
AVERAGE SSALES BY SREGION, SDIST .
IF NEWGRPV OF SREGION PRINT ' ' .
REPORT 1 .
TOTAL SSALES . AVERAGE SSALES .
```

When more than one break item controls a summary, the summary value of the minor item is printed when either of the item values changes. Each time the value of SDIST changes, IQL prints the accumulated summary for the district. When the value of SREGION changes, IQL prints the accumulated summary for SDIST, as well as the summary of SREGION.

The value of SREGION also controls the PRINT statements. These statements force blank lines between data about different regions. The literal in these statements is a space. Notice in the report itself, that three blank lines appear between regions. IQL always inserts an extra blank line between lines generated by PRINT statements and lines generated by summary statements. Of the three, the middle one is the line generated by the PRINT statement; it consists of a single blank space.

The Report:

```
05/09/79                    SALES PERFORMANCE                    PAGE  1
                          LOW QUOTA SALESMEN
                            WESTERN REGION

DISTRICT 2 Y-T-D SALES                          TOTAL:        $17,356
DISTRICT 3 Y-T-D SALES                          TOTAL:        $65,031
REGION 6 Y-T-D SALES                            TOTAL:        $82,387


DISTRICT 1 Y-T-D SALES                          TOTAL:        $27,250
DISTRICT 2 Y-T-D SALES                          TOTAL:        $48,605
DISTRICT 3 Y-T-D SALES                          TOTAL:        $27,671
REGION 7 Y-T-D SALES                            TOTAL:       $103,526


DISTRICT 1 Y-T-D SALES                          TOTAL:        $44,682
DISTRICT 2 Y-T-D SALES                          TOTAL:        $37,682
DISTRICT 3 Y-T-D SALES                          TOTAL:         $1,472
REGION 8 Y-T-D SALES                            TOTAL:        $83,836


OVERALL Y-T-D SALES                             TOTAL:       $269,749
OVERALL Y-T-D SALES                             AVG:          $13,487

(END QUERY PHASE; PRINT FILE IS QLB16ELPT)


05/09/79                                                     PAGE  1

DISTRICT 2 Y-T-D SALES                          AVG:          $17,356
DISTRICT 3 Y-T-D SALES                          AVG:          $10,839
REGION 6 Y-T-D SALES                            AVG:          $11,770


DISTRICT 1 Y-T-D SALES                          AVG:          $27,250
DISTRICT 2 Y-T-D SALES                          AVG:          $12,151
DISTRICT 3 Y-T-D SALES                          AVG:          $13,836
REGION 7 Y-T-D SALES                            AVG:          $14,789


DISTRICT 1 Y-T-D SALES                          AVG:          $14,894
DISTRICT 2 Y-T-D SALES                          AVG:          $18,841
DISTRICT 3 Y-T-D SALES                          AVG:           $1,472
REGION 8 Y-T-D SALES                            AVG:          $13,973

 END MULTIPLE REPORT PHASE

 REPORTS PRINTED-                  1
 LINES PRINTED-                   15
```

4.7 COMPUTATION

The COMPUTE statement is described in Chapter 2. Because COMPUTE has more general application than summary statements, this section expands on the format of this statement. In general, the COMPUTE statement tells IQL that an arithmetic equation is coming next. IQL pulls out its library of arithmetic subroutines and processes the equation for each set of values that is passed to it.

The Problem:

You must create a report on the projected bonuses of all salesmen in your department, based on the performance in January. The bonus is figured as the salesman's annual salary times the ratio by which he exceeded his quota. Since the weekly salary is kept in the SALESMAN file, you must convert this value to annual for the purposes of the calculation.

The Solution:

For this to be a complete report, you need the saleman's name, assignment, quota, and the bonus projected on the basis of sales in the month of January. A total bonus projection for all salesmen should appear at the end of the report.

The figure for the bonus must be calculated using the COMPUTE statement. This bonus can be stored in a numeric variable, which you could call XBONUS. The equation is composed of the salesman's January sales, times 12 (assume that the SSALES item contains one month's sales, and this is February); this yields the projected sales for the year. Next, you must subtract the sales quota from this, so that you now have the amount by which the sales exceed the quota. This must be multiplied by the ratio, which is SSAL times 52, divided by the quota. This computation generates the projected bonus.

This equation generates a negative number if the salesman's projected sales does not meet his quota. Obviously this negative bonus is not real, so you must avoid this result. The best way to do this is to check for negative results after the equation and change these results to zero. You can change any variable to a constant using the SET statement. EXAMP4-6 illustrates the use of this statement.

The Query:

```
<QA>RUN EXAMP4-6
 **EXAMP4-6

HEADING "BONUS PROJECTION//1979//BASED ON JANUARY PERFORMANCE" .
TITLES XBONUS = "PROJECTED//BONUS" . PICTURE XBONUS = "$$$,$$9" .
OPEN SALESMEN .
COMPUTE XBONUS = ( ( SSALES * 12 ) - SQUOTA ) * 52 * SSAL / SQUOTA .
IF XBONUS LEQ 0 SET XBONUS TO 0 .
PRINT SNAME, SASSIGN, SQUOTA, XBONUS .
TOTAL XBONUS .
```

Look first at the format of the COMPUTE statement. The variable that holds the results of the equation must stand alone on the left side of the equal sign. The entire computation appears on the right side of the equation. When IQL processes an equation, it follows the normal order of precedence for arithmetic operators.

IQL processes multiplications and divisions first, and then additions and subtractions. Leftmost operations are performed first. You may group expressions within parentheses to alter the order of execution. Innermost parentheses are analysed first.

When writing the IQL COMPUTE statement, be sure to leave a space on either side of each arithmetic symbol. Otherwise, IQL assumes that the symbol is a part of the word beside it.

The Report:

05/09/79 BONUS PROJECTION PAGE 1
1979
BASED ON JANUARY PERFORMANCE

SALESMAN NAME			ASSIGNMENT	QUOTA	PROJECTED BONUS
CATALDO	ALBERT	A	08-01-003	$200,000	$3,919
DONOVAN	ARTHUR	B	08-02-001	$250,000	$3,343
MILLER	FLORENCE	C	08-02-002	$300,000	$2,893
BUTLER	CARL	F	08-02-003	$175,000	$0
PORTER	THOMAS	T	08-03-001	$150,000	$0
WHITE	SANDRA	M	07-01-001	$225,000	$5,893
LUDWIG	RICHARD	C	07-01-002	$275,000	$3,460
WILLS	EDWARD	W	07-02-001	$150,000	$0
STEWART	LEONARD	R	07-02-002	$175,000	$0
DOUCETTE	ERNEST	J	07-02-003	$175,000	$1,262
MURRAY	FRANK	M	07-02-004	$275,000	$4,821
COSTA	CARLOS	E	07-03-002	$250,000	$3,082
DOWNING	MAURICE	T	06-01-001	$350,000	$0
ROBERTS	VINCENT	J	06-02-001	$275,000	$4,821
KELLY	MICHAEL	P	06-03-004	$175,000	$0
HUBERT	MICHAEL	J	06-03-005	$275,000	$0
POWERS	MARK	S	06-03-006	$200,000	$3,965
ROWE	JEFFREY	L	05-01-002	$200,000	$36,806
JOHNSTON	DANIELLE	M	05-02-001	$175,000	$0
PRATT	MARGARET	H	04-01-001	$175,000	$0
NADEAU	WAYNE	J	04-02-001	$330,000	$2,718
BATES	ROBERT	H	04-02-003	$200,000	$3,965
WILDE	ANDREW	R	02-01-001	$150,000	$0
CARNEY	PETER	I	01-01-001	$400,000	$2,022
OVERALL PROJECTED BONUS				TOTAL:	214,614.65

(END QUERY PHASE; PRINT FILE IS QLE16ELPT)

4.8 REWRITING FILES

You can use IQL to create a new file based on the records in your original input file. You can copy a subset of the records, sort the input file and copy the records in the new order, or rewrite the original file after changing the information in some records.

The COPY statement performs these functions. COPY creates a sequential output file and copies the records you choose. You can copy records out of an ISAM file, but note that the output file will be sequential. You cannot use COPY on database files.

The Problem:

Because of low sales potential, you must consolidate some of your sales regions and districts. The file on salesmen contains an assignment field made up of region, district and territory. The specific changes you must make are: in Region 6, consolidate Districts 1 and 2 into District 1, making the old territories into new territories. In addition, you must consolidate Regions 1, 2 and 3 into Region 2 with a single district, 1, and make the old territories new distinct territories under District 1.

The Solution:

IQL can copy records to a file. You can use this feature to make systematic changes to a data file. For this problem, you must make changes to certain records and then copy all the records to an output file.

You can specify the name of the output file to which the records are written. If you do not specify a name, the records are written to a new file with the same name as the input file, but with an extension of .OUT. It is a wise practice to allow IQL to use this default file as the output file, rather than to specify the input file as the output file (which would cause the input file to be rewritten). This allows you the opportunity to verify the new file without destroying the original input file. When you have verified the new file, you can rename it to replace the original file.

If you want to make a new copy of the entire file (as this example requires), make sure that the COPY statement is executed on every record in the input file, and not only on those that have been changed. If you want the output file to contain a subset of the original, place the COPY statement under control of an IF statement. You could also place it below a SORT statement, where it applies only to the records in that stage.

The Query:

```
<QA>RUN EXAMP4-7
 **EXAMP4-7

OPEN SALESMEN .
IF SREGION = 6 AND SDIST = 2
    COMPUTE SDIST = 1
    COMPUTE STERRTY = STERRTY + 1 .
IF SREGION = 3, 2, 1,
    COMPUTE STERRTY = SREGION
    COMPUTE SREGION = 2 COMPUTE SDIST = 1 .
COPY RECORD .
```

From EXAMP4-7, you get only a single line, since there are no statements that would produce an actual report.

```
END QUERY RUN.  NUMBER OF RECORDS COPIED ONTO SLSMEN FILE IS    52
```

You could, however, write the query to report some information on the changes made, as in EXAMP4-8.

The Revised Query:

```
<QA>RUN EXAMP4-8
 **EXAMP4-8

TITLES OFF .
OPEN SALESMEN .
PRINT ' ' .
PRINT SLNAME,'OLD ASSIGNMENT:', SASSIGN .
IF SREGION = 6 AND SDIST = 2
    COMPUTE SDIST = 1
    COMPUTE STERRTY = STERRTY + 1 .
IF SREGION = 3, 2, 1,
    COMPUTE STERRTY = SREGION
    COMPUTE SREGION = 2 COMPUTE SDIST = 1 .
PRINT 18,'NEW ASSIGNMENT:',3,SASSIGN .
COPY RECORD .
```

In this report a pair of lines is printed for every salesman, showing the assignment before and after the changes are made. Although this confirms that the query performed the operations you needed, you may want to do some additional verification of the file itself.

There are two ways to do this verification using IQL. One way is to query the SALESMEN dictionary in Immediate mode right away, and record the results of the session. These results reflect the data in the original SLSMEN.SEQ file. Then exit IQL, rename SLSMEN.OUT to be SLSMEN.SEQ, and return to IQL to perform the same series of Immediate mode queries and compare the results.

The other way is to use the Deferred mode. Write a query to produce a report containing the data that you changed, using the normal form of the OPEN statement. After running this query (which addresses the original SLSMEN.SEQ), go back and edit the line containing the OPEN statement so it includes the name of the output file, in addition to the name of the dictionary. The new line should look something like this:

```
OPEN SALESMEN 'SLSMEN.OUT' .
```

Now run the query again. This time, the query reads the output file instead of the original file. You now have reports for both the 'before' and 'after' image of the data, which you can compare to verify the execution of the COPY statement. When you are satisfied that the output file contains the data as you want it, you can rename the output file to conform with the file name specified in the dictionary (that is, the name of the original input file).

This second method takes advantage of two features of IQL. The first is that the output file created by the COPY statement can be interrogated with the same dictionary as the input file; the records remain in exactly the same format. The second feature is an option under the OPEN statement, which allows you to specify the name of the file you want the dictionary to address. You do this by specifying the file name within quotes as a part of the OPEN statement.

The Report:

```
04/09/79                                                    PAGE  1

JOHNSON          OLD ASSIGNMENT:   10-05-001
                 NEW ASSIGNMENT:   10-05-001

HANRATTY         OLD ASSIGNMENT:   10-05-002
                 NEW ASSIGNMENT:   10-05-002

THOMAS           OLD ASSIGNMENT:   10-05-003
                 NEW ASSIGNMENT:   10-05-003

THORNTON         OLD ASSIGNMENT:   10-04-001
                 NEW ASSIGNMENT:   10-04-001

GILLIAM          OLD ASSIGNMENT:   10-04-002
                 NEW ASSIGNMENT:   10-04-002

DOWNING          OLD ASSIGNMENT:   06-01-001
                 NEW ASSIGNMENT:   06-01-001

ROBERTS          OLD ASSIGNMENT:   06-02-001
                 NEW ASSIGNMENT:   06-01-002

BECKER           OLD ASSIGNMENT:   06-02-002
                 NEW ASSIGNMENT:   06-01-003

PRICE            OLD ASSIGNMENT:   06-03-001
                 NEW ASSIGNMENT:   06-03-001

SMITH            OLD ASSIGNMENT:   03-01-001
                 NEW ASSIGNMENT:   02-01-003

WILDE            OLD ASSIGNMENT:   02-01-001
                 NEW ASSIGNMENT:   02-01-002

CARNEY           OLD ASSIGNMENT:   01-01-001
                 NEW ASSIGNMENT:   02-01-001

(END QUERY PHASE; PRINT FILE IS QL418ELPT)
```

4.9 MULTIPLE DICTIONARIES

Now that you have queried the Salesman and Customer dictionaries extensively, you can query them both together. You may use up to three dictionaries in a single query.

Often when you refer to two dictionaries, you find information from the first file and use that information to find related information in the second file. The first dictionary is called the primary dictionary and the others are called secondary dictionaries.

You address the primary dictionary as if it were the only dictionary. The query reads the records, one after another, at the top of the first stage. Using the information in these records, you must control the secondary files yourself. This involves the use of the FIND statement, which is illustrated in EXAMP4-9.

SAMPLE MARKETING QUERIES

The Problem:

You need a report that matches salesmen up with customers. The report should contain the customer's name, in order by customer number, the salesman's name with his number, and the quota for that salesman.

The Solution:

If you look back at the dictionary for customers, you can see that the only clue about the salesman for each customer is the salesman's number, which is part of the customer record. Looking at the SALESMEN dictionary, you can see that there is no information about customers in those records. As a result, you must match the salesman number from the customer record with the salesman number from the salesman record. Using this link between records, you can obtain the information you need for your report.

When IQL is working with only one dictionary, only one record at a time is available for your query. But when two dictionaries are opened, IQL can work with two records at once: one from each data file. As a result, you can name items from the two records in a single PRINT statement (or in any other IQL statement).

This can create confusion if the same item name appears in both dictionaries. You can avoid this ambiguity in two ways. One way is never to use the same name in two dictionaries. This is the method used in the SALESMEN dictionary (all item names are preceded by an extra 'S'). The other method is used when the dictionaries are already defined and ambiguity is actually a problem. You prefix the item name with the dictionary name when it is used in a multiple dictionary query. Using this method, CNAME becomes CUSTOMER-CNAME.

The thing to remember in a multiple-dictionary queries is to match the secondary record with the primary record. First, you must have a current record from the primary file. Since the primary file is treated according to the normal access rules, you can use any of the methods for reading a file described thus far (such as the OPEN or SORT statements), as well as some that are shown in EXAMP4-9.

In this example, the customer record must be the primary one because it is the record that identifies the salesman. (Since the salesman's record does not identify a customer, it cannot be used to find the customer's record.) Once you have a customer's record as the current record, you can look at the salesman number (SLSMAN) and use this as the key for finding that salesman's record. You can do this within the query using the FIND statement.

The FIND statement is slightly different for sequential files, for ISAM files, and for database files. When you are finding a record in a secondary file, you must use the first variety because the secondary files must be sequential. (In those cases where you want an ISAM or DBMS file as your secondary file, take note that you can preprocess these files with the COPY statement; this will create a sequential file that can be referenced with the original dictionary.)

You use the FIND statement in this query to find a record from the secondary file, using the value of the SLSMAN item from the primary file. When you write a FIND statement of this form, identify the item you are searching for (from the secondary file) on the left side of an equal sign and the item that is the basis of the search (from the primary file) on the right side. EXAMP4-9 shows how to write the FIND statement.

The Query:

```
<QA>RUN EXAMP4-9
 **EXAMP4-9

HEADING "CUSTOMER/SALESMAN//TERRITORY//QUOTAS" .
TITLES CUSTNO = "CUST//NUMB"
       SSLSMN = "SMAN//NUMB" .
OPEN CUSTOMERS, SALESMEN .
FIND SSLSMN = SLSMAN FROM BEGINNING .
PRINT CUSTNO,1,CNAME,SSLSMN,SLNAME,SQUOTA .
```

The OPEN and FIND statements illustrate points of interest. The OPEN statement names both dictionaries used in this query. The first is the primary dictionary and the secondary one follows. Because the customer record is the one with the key information in it, the CUSTOMERS dictionary is named as the primary one.

The key is the item named SLSMAN. This key is used in the FIND statement. In the FIND statement, SLSMAN is on the right side of the equal sign. On the left side is the name of the item you are looking for, which is SSLSMN in this example. You also need to use the qualifying phrase, FROM BEGINNING, (which is explained below) so the query searches the SALESMEN file from top to bottom for the correct record. If IQL does not find the record, an all-blank record is returned.

In this query, a record is read from the CUSTOMERS file first. This happens as part of the OPEN statement. In the FIND statement, you named the SLSMAN item as the basis for the search. IQL extracts the value of this item from the customer record. Next IQL checks the other item you have named: SSLSMN. From the dictionaries, IQL knows that this item is a part of the SALESMEN file.

IQL now searches the SALESMEN file. Each record is read in turn, and the value of the SSLSMN item is compared to the value of the SLSMAN item being looked for. When IQL finds a match, it holds both records and executes the rest of the stage. In this query, the customer is matched up with the salesman. From this point on, the statements in the query can address either record, or both. EXAMP4-9 contains a PRINT statement that addresses items from both records.

When IQL finds the right record from the secondary file, it leaves the secondary file positioned at that record. Because they are sequential, secondary files are read from the beginning until the proper record is found. The next time this file is searched, the search starts where the previous search left off. This is what is meant by the position of the file.

If you want the search to start at the beginning of the file on the second and subsequent searches, you must specify FROM BEGINNING in the FIND statement. If this phrase is present, IQL repositions the file before it begins to search for a record the next time around. If you use this phrase, you can be sure that IQL reads the entire secondary file when looking for a match.

If IQL cannot find a record that contains the value it is searching for, it substitutes a record that contains all blanks. This happens for customer number 26 in the report. All items printed from the SALESMEN record are blank, because IQL could not find a salesman record that matched the value from the customer's record.

The Report:

04/10/79 CUSTOMER/SALESMAN TERRITORY QUOTAS PAGE 1

CUST NUMB	CUSTOMER NAME	SMAN NUMB	SALESMAN LAST NAME	QUOTA
1	SCARLOTTI VIOLIN CO.	1	JOHNSON	$200,000
2	WILDCAT OIL INC.	2	HANRATTY	$200,000
3	BARBAZON HOTELS, INC	3	THOMAS	$250,000
4	MILITARY SUPPLY INC.	4	THORNTON	$250,000
5	FORESTMAN NURSERIES	5	GILLIAM	$250,000
6	DEL RAY PROPERTIES, INC	6	BELL	$300,000
7	CONSOLIDATED MARINAS, INC	7	HARRISON	$300,000
8	SOLVANG SAILS, INC	8	FREDRICKS	$200,000
9	NAVAJO PRODUCTIONS	9	RIBELIO	$350,000
10	SONJA L MARTIN INC	10	MARTINELLI	$225,000
11	GALACTIC EXPLORATION INC.	11	ELLINGTON	$175,000
12	INTERNATIONAL MOTORS CO.	12	FEENEY	$150,000
13	MARVELL MOTORCYCLES INC	13	HAYES	$175,000
14	WESTERN BROASTED CHICKEN INC	14	LOWE	$200,000
15	ELECTRONICS CONCEPTS, INC	15	MAHONEY	$225,000
16	SCOTTISH HABERDASHERS LTD	16	TAYLOR	$175,000
17	GLOBALEX	17	STEVENSON	$200,000
18	LAKESIDE HOMES, INC	18	BENSON	$175,000
19	GARLAND TRAVEL TOURS	19	CATALDO	$200,000
20	CERTIFIED SOFTWARE PACKAGES	20	DONOVAN	$250,000
21	REALIZATION STUDIOS, INC	21	MILLER	$300,000
22	JONES AND HILDEBRAND	22	BUTLER	$175,000
23	DEEPSEA DIVERS, INC	23	PORTER	$150,000
24	CHARLESGATE ASSOCIATES	24	WHITE	$225,000
25	JEFFERSON ENTERPRISES	25	LUDWIG	$275,000
26	FIBERGLASS ASSOCIATES	0		$
27	DESERT PROPERTIES	27	STEWART	$175,000
28	LITCHFIELD MOTORS	28	DOUCETTE	$175,000

CHAPTER 5

EXAMPLE ACCOUNTING QUERY

You may want to write a report that contains information from more than one source. The multiple-file capability of IQL, discussed in Chapter 4, provides the ability to do this. The example in this chapter expands on this ability. This example demonstrates how to chain three files together, using the information from one to categorize and define another.

One of the most important factors that affects data processing is the efficient definition of data records. Records that are too large slow down processing each time the file is read. You should define a record so that it includes only items that are useful whenever the record is processed. If some of the items are relevant only part of the time, these items are like dead weight the rest of the time. If you create multiple data files that can be linked together during processing, you can avoid this problem.

The example in this chapter illustrates how three sequential files can be linked together to facilitate efficient processing. The three dictionaries for these files are shown in Section 5.1. These dictionaries are named BALANCES, RANGES, and CHARTS. They can be used together to generate a daily balance sheet in an accounting application.

5.1 THE DICTIONARIES

The three dictionaries used in this example are shown below.

```
<QA>ITEMS BALANCES

DICT        FILE     FILE-IN                    REC   BLK   KEY KY KY RD CP RW
NAME        TYPE     NAME         DIRECT        LEN   FAC   LOC LN TP PW PW PW
----        ----     ----         ------        ---   ---   --- -- -- -- -- --

BALANCES    SQ DSK7  BANKS SEQ    <IQL30-DIST>  80      0     0  0

   ITEM      TOP          BOTTOM       1ST  NO.  T S PRINTING          SCAN
ID NAME      TITLE        TITLE        CHAR CHAR Y C PICTURE           GNNS PT
-- --------- ------------ ------------ ---- ---- - - ----------------- ---- --

DD B-ACCT    ACCOUNT        NO           1    6 N 0 SZZZZZ9
DD B-BAL     CURRENT      BALANCE        8    6 N 0 ZZZ,ZZZ
DD B-MTD        MTD       AVERAGE DB    15    6 N 0 ZZZ,ZZZ
DD B-BUD     CURRENT      MO BUDGET     22    6 N 0 ZZZ,ZZZ
DD B-AVG     LAST YR MO   AVERAGE DB    29    6 N 0 ZZZ,ZZZ

 (END LIST OF ITEMS)

<QA>ITEMS RANGES

DICT        FILE     FILE-IN                    REC   BLK   KEY KY KY RD CP RW
NAME        TYPE     NAME         DIRECT        LEN   FAC   LOC LN TP PW PW PW
----        ----     ----         ------        ---   ---   --- -- -- -- -- --

RANGES      SQ DSK7  RANGESSEQ                    80     0     0  0

   ITEM      TOP          BOTTOM       1ST  NO.  T S PRINTING          SCAN
ID NAME      TITLE        TITLE        CHAR CHAR Y C PICTURE           GNNS PT
-- --------- ------------ ------------ ---- ---- - - ----------------- ---- --

DD R-FROM    ACCOUNT        FROM         1    6 N 0 SZZZZZ9
DD R-TO      ACCOUNT        TO           8    6 N 0 SZZZZZ9
DD R-LINE     LINE        NUMBER        15    3 N 0 SZZ9

 (END LIST OF ITEMS)

<QA>ITEMS CHARTS

DICT        FILE     FILE-IN                    REC   BLK   KEY KY KY RD CP RW
NAME        TYPE     NAME         DIRECT        LEN   FAC   LOC LN TP PW PW PW
----        ----     ----         ------        ---   ---   --- -- -- -- -- --

CHARTS      SQ DSK7  CHARTSSEQ                    80     0     0  0

   ITEM      TOP          BOTTOM       1ST  NO.  T S PRINTING          SCAN
ID NAME      TITLE        TITLE        CHAR CHAR Y C PICTURE           GNNS PT
-- --------- ------------ ------------ ---- ---- - - ----------------- ---- --

DD C-LINE     LINE        NUMBER         1    3 N 0 SZZ9
DD C-TITLE   TITLE        CODE           5    2 A 0
DD C-DESC    CHART OF     DESCRIPTON     8   30 A 0

 (END LIST OF ITEMS)
```

Notice that the records in each file, as defined in these dictionaries, are limited in scope. A record in the BALANCES file

contains an account number and four pieces of financial information. This data is useful in a number of applications, such as preparing statements and projecting cash flow. Since all financial information is contained in this file, it can be used in a variety of contexts. However, you need more information to prepare a summary balance sheet.

The RANGES file contains some of the information that can make BALANCES usable. Note that the BALANCES dictionary has the account number defined as a data item. Since account numbers are assigned according to a standard system, it is useful to group them together and look at all of the accounts within a given range. The RANGES dictionary contains these groupings. For each range of account numbers, there is a line number. This line number, which is held in the R-LINE item, is a pointer into the records defined in the CHARTS dictionary.

The CHARTS dictionary contains three items: the line number (matching the item from RANGES), a title code, and a 35 character description. Each entry in the RANGES file has a corresponding description in the CHARTS file. An entry in the RANGES file corresponds to a number of entries in the BALANCES file. Using these links between records, you can generate a complete summary balance sheet.

5.2 THE QUERY

The object of this query is a report listing balances and other financial information. The report must summarize the accounts within various line items, as defined by the value of the account number. The report must show subtotals of assets, liabilities, and equity, and these ledgers must be checked for balance.

Given the structure of the data records, the logic of the query can be expressed in simple terms. You first need to get a description from the CHARTS file and decide how to treat it. The chart lines fall into a number of categories, identified by the title code. The title code determines how this particular chart line should be handled.

If the title code indicates that this is the title of a range, the range must be identified. The range identifies which records from the BALANCES file should be totaled within this group. The records from the BALANCES file are read at this point, and the financial information is totaled up until the range is exhausted. When this occurs, it is time to print the description and the accumulated totals. Finally the next chart record is read, and the process repeats itself.

If the code indicates that the title refers to an overall summary, then the accumulated summary for a group of ranges must be printed. These values are accumulated in variables during processing, and can be printed like any other item. Note that the SUMPRINT OFF statement prevents IQL from printing its own total line. Instead, the totals are stored in variables so you can print them when and how you want.

EXAMPLE ACCOUNTING QUERY

```
*************************************************************************
* QUERY TO PRODUCE SUMMARY BALANCE SHEET FOR ALL ACCOUNTS IN BANKS.SEQ
***************
* THIS QUERY PROVIDES FOR ITS OWN TITLES AND SUMMARIES
***************
  HEADING "DEPOSITOR'S TRUST// //DAILY BALANCE SHEET//
          //TOTAL COMPANY" .
  SUMPRINT OFF .
  TITLES OFF .
  RMARGIN 80 .
***************
* XL CLASS IS ACCOUNT LINES (CODE = D1 AND D2), TOTALED IN ROUTINE #10
* XT CLASS IS SUBTOTALS (CODE = A1), TOTALED IN ROUTINE #40
* XF CLASS IS GROUP TOTALS (CODE = A2 AND A3), TOTALED IN ROUTINE #40
* XA CLASS IS ASSETS ONLY (LINE NUMBER IS 50), TOTALED IN ROUTINE #60
***************
  PICTURE XL-BAL = "ZZZ,ZZZ", XL-MTD = "ZZZ,ZZZ", XL-BUD = "ZZZ,ZZZ" .
  PICTURE XT-BAL = "ZZZ,ZZZ", XT-MTD = "ZZZ,ZZZ", XT-BUD = "ZZZ,ZZZ" .
  PICTURE XF-BAL = "ZZZ,ZZZ", XF-MTD = "ZZZ,ZZZ", XF-BUD = "ZZZ,ZZZ" .
  PICTURE XA-BAL = "ZZZ,ZZZ", XA-MTD = "ZZZ,ZZZ", XA-BUD = "ZZZ,ZZZ" .
*************************************************************************

  OPEN BALANCES, CHARTS, RANGES .
  IF NOT FIRSTIME GO TO 08 .

***************
* ROUTINE #05 AT TOP OF PAGE ONLY
***************
05VSPACE 1 .
  PRINT 39,"CURRENT",4,"   MTD   ","CURRENT" .
  PRINT 5,"DESCRIPTION",23,"BALANCE",4,"AVERAGE","BUDGET" .
  GO TO 20 .
08IF LASTIME GO TO 40 .

***************
* TOTAL UP THE DATA WITHIN THE CURRENT RANGE
* AT END OF RANGE GO PRINT THE RESULTS
***************
10IF B-ACCT >  R-TO GO TO 40 .
  IF B-ACCT = R-FROM TOTAL B-BAL ( = XL-BAL )
                     TOTAL B-MTD ( = XL-MTD )
                     TOTAL B-BUD ( = XL-BUD ) .
* IF STILL WITHIN RANGE, GET NEXT ACCOUNT FROM BANKS.SEQ
  GO TO NR .

***************
* GET A NEW CHART LINE AND DEAL WITH IT
***************
20FIND C-LINE = NEXT .
  IF C-TITLE = "T2","T3", VSPACE 2
                          PRINT C-DESC
                          GO TO 20 .
  IF C-TITLE = "T1" GO TO 40 .
  IF C-TITLE = "A1" GO TO 50 .
  IF C-TITLE = "A2","A3" GO TO 60 .
* FALL THROUGH ON D1 AND D2
```

```
***************
* GET A NEW RANGE
***************
30FIND R-FROM = NEXT .
* IF RANGE MATCHES CURRENT CHART, TOTAL THE ACCOUNTS
  IF R-LINE = C-LINE GO TO 10 .
* SKIP THIS CHART-ITEM IF THE RANGE HAS PASSED IT
  IF R-LINE > C-LINE GO TO 20 .
* SKIP THIS RANGE IF IT ISN'T IN THE CHART
  GO TO 30 .

***************
* PRINT TOTALS WHEN THE RANGE IS EXHAUSTED
***************
40VSPACE 1 .
  PRINT C-DESC,4,XL-BAL,XL-MTD,XL-BUD .
* ACCUMULATE SUBTOTALS IN XT AND GROUP TOTALS IN XF
  TOTAL XL-BAL ( = XT-BAL ) .
  TOTAL XL-MTD ( = XT-MTD ) .
  TOTAL XL-BUD ( = XT-BUD ) .
  TOTAL XL-BAL ( = XF-BAL ) .
  TOTAL XL-MTD ( = XF-MTD ) .
  TOTAL XL-BUD ( = XF-BUD ) .
* ZERO OUT XL AND MOVE ON TO THE NEXT CHART LINE
  RESET XL-BAL, XL-MTD,XL-BUD .
  GO TO 20 .

***************
* PRINT SUBTOTALS WHEN CODE IS A1
***************
50VSPACE 1 .
  PRINT C-DESC,4,XT-BAL,XT-MTD,XT-BUD .
  RESET XT-BAL,XT-MTD,XT-BUD .
  GO TO 20 .

***************
* TAKE CARE OF A2 AND A3 CODES
***************
* CREATE XL VALUES FOR EQUITY LINE; GO PRINT THEM
60IF C-LINE = 092 COMPUTE XL-BAL = XF-BAL - XA-BAL
                  COMPUTE XL-MTD = XF-MTD - XA-MTD
                  COMPUTE XL-BUD = XF-BUD - XA-BUD
                  GO TO 40 .
* IF THIS IS LAST CHART LINE, PRINT DEBIT TOTALS; STOP EXECUTION
  VSPACE 2 .
  IF C-LINE = 094
        PRINT C-DESC,4,XA-BAL,XA-MTD,XA-BUD
        GO TO XT .
* CHART LINE IS NOW 50, 82, OR 88
  PRINT C-DESC,4,XF-BAL,XF-MTD,XF-BUD .
* STORE TOTAL ASSETS IN XA; SET UP PAGE TO PRINT LIABILITIES
  IF C-LINE = 50 COMPUTE XA-BAL = XF-BAL
                 COMPUTE XA-MTD = XF-MTD
                 COMPUTE XA-BUD = XF-BUD
                 NEWPAGE GO TO 05 .
* CHART LINE IS 82 OR 88
  IF C-LINE NOT = 088  RESET XF-BAL,XF-MTD,XF-BUD .
  RESET XT-BAL,XT-MTD,XT-BUD,XL-BAL,XL-MTD,XL-BUD .
  GO TO 20 .
```

5.3 THE REPORT

The report is shown below. Notice that the descriptions printed on the left side of the page, are the C-DESC items. Since this report requires these descriptive headings, in addition to standard column headings, the headings are stored in a file. However, the headings are not stored in the same record as the ranges, even though they correspond. Rather, the two records are held separately so that each record is more efficient. If you had to process the RANGES file without the headings, it would be a much simpler operation this way. The BALANCES file is also unencumbered by cosmetic information, such as the name of the account holder. That information could be stored in another data file, to be matched up with the BALANCE file whenever the two are needed together.

05/24/79 DEPOSITOR'S TRUST PAGE 1

DAILY BALANCE SHEET

TOTAL COMPANY

DESCRIPTION	CURRENT BALANCE	MTD AVERAGE	CURRENT BUDGET
ASSETS			
CASH & DUE FROM BANKS			
CASH & RESERVE	2,487	2,460	2,580
FLOAT	3,711	3,297	1,646
DUE FROM BANKS	4,711	4,746	3,910
TOTAL CASH & DUE FROM BANK	10,909	10,503	8,136
SECURITIES			
U. S. GOVT	18,785	18,785	5,795
AGENCIES	4,000	4,000	5,000
STATE & POLITICAL	24,928	24,937	23,700
OTHER	5	5	5
TOTAL SECURITIES	47,718	47,727	34,500
LOANS			
DEMAND AND TIME	29,882	29,873	37,301
TAX FREE	888	893	388
INSTALLMENT	25,587	25,586	29,864
BANKAMERICARD	3,250	3,254	3,460
REAL ESTATE	12,871	12,875	11,000
CONSTRUCTION	7,371	7,329	7,500
ACCEPTANCES FUNDED			
TOTAL LOANS	79,849	79,810	89,513
PREMISES AND EQUIPMENT-NET	2,548	2,548	2,794
OTHER ASSETS	3,051	2,514	1,309
TOTAL ASSETS	144,075	143,102	136,252

EXAMPLE ACCOUNTING QUERY

05/24/79 DEPOSITOR'S TRUST PAGE 2

DAILY BALANCE SHEET

TOTAL COMPANY

DESCRIPTION	CURRENT BALANCE	MTD AVERAGE	CURRENT BUDGET
LIABILITIES & STOCKHOLDERS' EQUITY			
DEMAND DEPOSITS			
INDIVIDLS, PRTNRSHPS & CORPS	42,678	42,458	41,765
DUE TO BANKS	215	215	230
U. S. GOVERNMENT	1,392	914	1,412
STATE & POLITICAL	987	995	1,518
OTHER DEMAND	1,373	1,383	1,378
TOTAL DEMAND DEPOSITS	52,244	51,027	50,406
TIME DEPOSITS			
PASSBOOK & STATEMENT SAVINGS	26,364	26,229	24,135
SAVINGS CERTIFICATES	20,073	26,229	24,135
OTHER CONSUMER SAVINGS	4,034	4,010	4,541
C/D'S OVER $100M	23,532	23,557	21,000
OTHER TIME	1,310	1,310	2,120
TOTAL TIME DEPOSITS	75,313	81,335	75,931
TOTAL DEPOSITS	266,033	270,402	258,486
NET MONEY POSITION	7,000	7,125	3,042
OTHER LIABILITIES	4,198	4,206	4,438
TOTAL LIABILITIES	11,198	11,331	7,480
RESERVE FOR LOAN LOSSES	1,257	1,259	1,300
STOCKHOLDER'S EQUITY	131,620	130,512	127,472
TOTAL LIABILITIES, RES & EQUITY	144,075	143,102	136,252

INDEX

TOPS-10/TOPS-20
Introduction to
Interactive Query Language
AA-0947B-TK

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify)_____

Name_____ Date_____

Organization_____

Street_____

City_____ State_____ Zip Code_____
or
Country

Please cut along this line.

Do Not Tear - Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE PUBLICATIONS
200 FOREST STREET MR1-2/E37
MARLBOROUGH, MASSACHUSETTS 01752

Do Not Tear - Fold Here and Tape

Cut Along Dotted Line