HCL Informix SQL User Guide

Version 7.51

Table of Contents

Introduction **12**

In This Introduction 13
About This Guide 13
Organization of This Guide 4
Types of Users 4
Software Dependencies 5
Documentation Conventions 5
Typographical Conventions 5
Icon Conventions 6
On-Line Error Messages 9

Introducing INFORMIX-SQL **11**

In This Chapter 3
What Is INFORMIX-SQL? 4
What Is a Database? 4
How INFORMIX-SQL Stores Data 5
Retrieving Data from the Database 9
Connecting Tables Through Column Joins 10
Getting Started with INFORMIX-SQL 14
The INFORMIX-SQL Main Menu 14
The Current Option 15
The Message Line 15
Moving the Highlight 15
Selecting an Option 16
What the Main Menu Options Mean 16
Using the HELP Menu 19
The INFORMIX-SQL Facilities 20
Creating Tables with the Schema Editor 20
Using a Screen Form 21
The Report Writing Facility 25
Using Structured Query Language 28
The User-Menu System 30
A Complete Database Management System 31

Installing INFORMIX-SQL 31
Order of Installation 32
Establishing the Informix User, Group, and Directory 32
Setting Informix Environment Variables 33
Installing INFORMIX-SQL 35
Preparing to Use INFORMIX-SQL 38
Setting Environment Variables 38
Creating the Demonstration Database and Examples 41
Restricting Database Access 41

Creating a Database 43
In This Chapter 3
Two Ways to Create Databases and Tables 3
Using the Main Menu to Create Databases and Tables 4
Using SQL to Create Databases and Tables 4
Accessing INFORMIX-SQL 5
Using the Main Menu to Create a Database 5
Guidelines for Selecting Database Names 7
The Current Database 8
System Files That Make Up a Database 8
Selecting a Different Current Database 9
Leaving a Current Menu or Screen 10
Using the Main Menu to Create a Table 10
Naming the New Table 12
Using the CREATE TABLE Menu 12
An Overview of Building a Table 13
Naming the Columns 15
Assigning Data Types 15
Assigning a Data Type to the Column 20
Creating Indexes 22
NULL Values 24
Adding the Remaining Columns to the Table 25
Exiting the Schema Editor 26
System Files 28
Changing the Table Specifications 28

Entering Data 31
In This Chapter 3
What Is a Screen Form? 3
Screen Forms in the Demonstration Database 4
What Is PERFORM? 5

Including Data from Multiple Tables 5
Each Table Can Contribute to Several Forms 6
Using a Screen Form 6
The PERFORM Screen 8
Information Lines 9
Screen Form 10
Status Lines 10
The PERFORM Menu Options 11
Adding Data to a Database 12
Entering Data in the Fields 13
Confirming Your Entry 17
Data Checking in PERFORM 17
Special Functions 18
Positioning the Cursor 18
Field Editing 19
Using the Multiline Editor 20
Storing the Row 21
Changing Existing Data 21
Removing a Row 22
Exiting PERFORM 22

Querying a Database 23

In This Chapter 3
What Is a Database Query? 3
Searching for an Exact Match 4
Searching for a Group of Rows 7
Finding All the Rows in a Table 9
How Database Queries Work 10
Searching with Relational Operators 11
Using Wildcard Characters 13
Searching for a Range of Values 14
Searching for the Highest and Lowest Values 15
Searching for Alternative Values 16
Query Operators and Short Fields 16
Searching in SMALLFLOAT and FLOAT Fields 17
Sample Database Queries 18
Moving from Screen to Screen 18
Saving the Results of a Query 19

Using Multiple-Table Forms 27

In This Chapter 3

What Is a Multiple-Table Form? 3
The Active Table 4
Changing the Active Table 4
Join Fields 8
Joins in the orderform Form 9
Verify Joins 11
Lookup Joins 12
The Master-Detail Relationship 12
Selecting the Detail Table 13
Returning to the Master Table 16
Multiple Tables and Current Lists 16
Using the Current Option 16

Creating Your Own Forms 17

In This Chapter 3
What Is a Form Specification? 3
Default Form Specifications 4
How to Create a Default Form 4
How to Modify an Existing Form 4
How to Create a Form 5
Naming the Form 6
Choosing Tables to Include in the Form 6
Compiling the Default Form Specification 7
A Sample Default Form 9
How to Modify an Existing Form 10
What the Form Specification Contains 11
An Example Form Specification 11
The customer Form Specification 14
Comparing the custom and customer Forms 15
Using FORMBUILD Attributes 19
How to Enter Attributes 19
Displaying Field Text in Color 21
Making a Field INVISIBLE 22
Displaying Comments on the Screen 23
Changing Data from Lowercase to Uppercase 24
Changing Default Field Values 24
Requiring User Input 26
Specifying Acceptable Values 26
Verifying Input 28
Joining Fields 29

The INSTRUCTIONS Section 33
Calling C Functions from PERFORM 34

Using SQL 35

In This Chapter 3
Examples Used in This Chapter 3
What Is SQL? 4
Using SQL Statements 4
Accessing SQL 5
SQL Options 5
The SQL SYNTAX Menu 6
Learning to Use SQL 6
Differences Between SQL and PERFORM 7
How to Create a Table with SQL 8
Entering an SQL Statement 8
Using the SQL Editor 9
Using the Use-Editor Option 11
Running an SQL Statement 12
If There Are Errors 12
Saving the Current Statements 14
Using a Command File 15
Formatting SQL Statements 16
Sending Query Results to a Printer or a File 17
Displaying Table Information 18

Creating and Printing Reports 21

In This Chapter 3
What Is a Report Writer? 3
Steps in Creating a Report 5
Using the REPORT Menu 6
A Sample Specification and Report 7
Sections of a Report Specification 9
Creating a Report Specification 12
Generating a Default Report Specification 12
Naming the Report 13
Compiling the Default Report Specification 14
Running the Report 15
Modifying a Report 18
Saving the Report Specification 20
Compiling a Report Specification 21
If the Compilation Is Unsuccessful 21

What a Report Specification Contains 23
Listing the Instructions 24
Including Comments 24
Selecting the Data for a Report 25
Formatting the Report 26
Selecting All Rows and Columns in a Table 26
Selecting Data from More than One Table 27
Reading the Data for a Report 27
Basic Report Formatting 28
The OUTPUT Section 29
The FORMAT Section 29
Creating Page Headers and Trailers 31
Printing Rows of Data 35
Grouping Data 37
Using Calculations in a Report 40
Arithmetic Operators 41
Controlling the Appearance of Numbers 42
Math Functions 45
Group Functions 49
Displaying a Report 52
Printing the Report on a Printer 52
Writing the Report to a File 53
Piping the Report to Another Program 53
Calling C Functions from ACE 53

Database Structure and Integrity 55
In This Chapter 3
Changing the Structure of a Database 3
Removing a Table 4
Removing an Index 4
Removing a Database 5
Renaming a Table 5
Changing the Structure of a Table 6
Using the Schema Editor 6
Transactions 10
Audit Trails 10
Comparing Audit Trails and the Transaction Log 11
Views 11

Notices and Information 1

Index **10**

Introduction

About This Guide 3
Organization of This Guide 4
Types of Users 4
Software Dependencies 5

Documentation Conventions 5
Typographical Conventions 5
Icon Conventions 6
Comment Icons 6
Feature, Product, and Platform Icons 7

In This Introduction

This introduction provides an overview of the information in this guide and describes the conventions it uses.

About This Guide

This guide and its companion volume, the *INFORMIX-SQL Reference Manual*, make up the comprehensive documentation for the INFORMIX-SQL relational database management system.

The guide is an introduction to INFORMIX-SQL. You do not need database management experience or familiarity with basic database management concepts to use this guide. It includes general information about database systems and leads you through the steps necessary to create a database, enter and access database information, and produce printed reports. The first chapter covers important basic information about database systems. Subsequent chapters contain instructions and illustrations that show you how to take advantage of many of the powerful INFORMIX-SQL features. Each chapter also tells you where to look in the *INFORMIX-SQL Reference Manual* for information about advanced features.

The *INFORMIX-SQL Reference Manual* is a complete reference to the programs that make up INFORMIX-SQL. It contains information about everything you can do with INFORMIX-SQL, and it is organized by program name. After you are familiar with INFORMIX-SQL basics, you can use the *INFORMIX-SQL Reference Manual* to learn about advanced features and to quickly locate specific information.

Organization of This Guide

Each chapter in this book begins with a section that describes chapter contents. This section includes a paragraph to help you identify topics appropriate for your needs. The rest of the chapter explains each topic area and demonstrates its use in INFORMIX-SQL.

The *INFORMIX-SQL User Guide* includes the following chapters and appendixes:

- Chapter 1, "Introducing INFORMIX-SQL," introduces important concepts about database management systems. It also includes a section on how to install and prepare to use INFORMIX-SQL.
- Chapter 2, "Creating a Database," explains how to set up a database and describes how to create tables and indexes.
- Chapter 3, "Entering Data," explains how to enter data into database tables, change data, and delete data.
- Chapter 4, "Querying a Database," describes how you can use screen forms to query the database.
- Chapter 5, "Using Multiple-Table Forms," explains how you can use screen forms that include multiple database tables.
- Chapter 6, "Creating Your Own Forms," describes how you can create your own custom screen forms.
- Chapter 7, "Using SQL," describes the SQL query language, which you can use to enter, alter, delete, or retrieve data from the database.
- Chapter 8, "Creating and Printing Reports," explains how to create, display, and print reports.
- Chapter 9, "Database Structure and Integrity," describes how to alter the structure of databases and tables, and how to maintain data integrity.

Types of Users

This guide is written for all INFORMIX-SQL developers. You do not need database management experience or familiarity with relational database concepts to use this guide. A knowledge of Structured Query Language (SQL), however, and experience using a high-level programming language would be useful.

Software Dependencies

This guide is written with the assumption that you are using an Informix database server, Version 14.10 or later.

Documentation Conventions

This section describes the conventions that this guide uses. These conventions make it easier to gather information from this and other volumes in the documentation set.The following conventions are discussed:

- Typographical conventions
- Icon conventions

Typographical Conventions

This guide uses the following conventions to introduce new terms, illustrate screen displays, describe command syntax, and so forth.

Convention	Meaning
KEYWORD	All primary elements in a programming language statement (keywords) appear in uppercase letters in a serif font.
italics ***italics*** `italics`	Within text, new terms and emphasized words appear in italics. Within syntax diagrams and code examples, identifiers or values that you are to specify appear in italics.
boldface ***boldface***	Names of program entities (such as classes, events, and tables), environment variables, file and pathnames, and interface elements (such as icons, menu items, and buttons) appear in boldface.
`monospace` *`monospace`*	Information that the product displays and information that you enter appear in a monospace typeface.

Tip: *When you are instructed to "enter" characters or to "execute" a command, immediately press* RETURN *after the entry. When you are instructed to "type" the text or to "press" other keys, no* RETURN *is required.*

Icon Conventions

Throughout the documentation, you will find text that is identified by several different types of icons. This section describes these icons.

Comment Icons

Comment icons identify three types of information, as the following table describes. This information always appears in italics.

Icon	Label	Description
	***Warning:***	Identifies paragraphs that contain vital instructions, cautions, or critical information
	***Important:***	Identifies paragraphs that contain significant information about the feature or operation that is being described
	***Tip:***	Identifies paragraphs that offer additional details or shortcuts for the functionality that is being described

Feature, Product, and Platform Icons

Feature, product, and platform icons identify paragraphs that contain feature-specific, product-specific, or platform-specific information.

Icon	Description
GLS	Identifies information that relates to the Informix Global Language Support (GLS) feature
IDS	Identifies information or syntax that is specific to Informix Dynamic Server and its editions
SE	Identifies information or syntax that is specific to INFORMIX-SE

These icons can apply to a row in a table, one or more paragraphs, or an entire section. A ♦ symbol indicates the end of the feature-specific, product-specific, or platform-specific information.

Documentation Included with INFORMIX-SQL

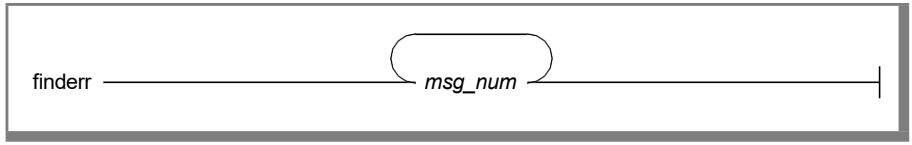
The INFORMIX-SQL documentation set includes the following manuals.

Manual	Description
INFORMIX-SQL Reference Manual	A complete reference to the programs that make up INFORMIX-SQL. It contains information about everything you can do with INFORMIX-SQL and is organized by program name. After you have read the *INFORMIX-SQL User Guide* and are familiar with INFORMIX-SQL basics, you can use the *INFORMIX-SQL Reference Manual* to learn about advanced features and to quickly locate specific information.
INFORMIX-SQLUser Guide	Introduces INFORMIX-SQL and provides the context needed to understand the other manuals in the documentation set. You do not need database management experience or familiarity with basic database management concepts to use this guide. It includes general information about database systems and leads you through the steps necessary to create a database, enter and access database information, and produce printed reports.
Informix Guide to GLS Functionality	Provides full information on the use of Global Language Support (GLS).
Informix Guide to SQL: Reference	Provides full information on the structure and contents of the demonstration database that is provided with INFORMIX-SQL. It includes details of the Informix system catalog tables, describes Informix and common environment variables that should be set, and describes the column data types that Informix database servers support. It also provides a detailed description of all the SQL statements that Informix products support.
Informix Guide to SQL: Syntax	Contains syntax diagrams for all the SQL statements and statement segments that Informix database servers support.
Informix Guide to SQL: Tutorial	Provides a tutorial on SQL as it is implemented by Informix products and describes the fundamental ideas and terminology that are used when you plan and implement a relational database. It also describes how to retrieve information from a database and how to modify a database.

On-Line Error Messages

Use the **finderr** script to display a particular error message or messages on your screen. The script is located in the **$INFORMIXDIR/bin** directory.

The **finderr** script has the following syntax.



msg_num Indicates the number of the error message to display. Error message numbers range from -1 to -32000. Specifying the - sign is optional.

For example, to display the -359 error message, you can enter either of the following:

```
finderr  -359
```

or, equivalently:

```
finderr  359
```

The following example demonstrates how to specify a list of error messages. The example also pipes the output to the UNIX **more** command to control the display. You can also direct the output to another file so that you can save or print the error messages:

```
finderr 233 107 113 134 143 144 154 | more
```

A few messages have positive numbers. These messages are used solely within the application tools. In the unlikely event that you want to display them, you must precede the message number with the + sign.

The messages numbered -1 to -100 can be platform-dependent. If the message text for a message in this range does not apply to your platform, check the operating-system documentation for the precise meaning of the message number.

Chapter

1 Introducing INFORMIX-SQL

In This Chapter 1-3

What Is INFORMIX-SQL? 1-4

What Is a Database? 1-4
- How INFORMIX-SQL Stores Data 1-5
  - A Database Is a Collection of Tables 1-5
  - Tables Are Made Up of Rows and Columns 1-6
- Retrieving Data from the Database 1-9
  - Searching for Rows 1-9
  - Searching for Columns 1-9
  - Searching for Columns Within Rows 1-10
- Connecting Tables Through Column Joins 1-10

Getting Started with INFORMIX-SQL 1-14

The INFORMIX-SQL Main Menu 1-14
- The Current Option 1-15
- The Message Line 1-15
- Moving the Highlight 1-15
- Selecting an Option 1-16
- What the Main Menu Options Mean 1-16
- Using the HELP Menu 1-19

The INFORMIX-SQL Facilities 1-20
- Creating Tables with the Schema Editor 1-20
- Using a Screen Form 1-21
- The Report Writing Facility 1-25
- Using Structured Query Language 1-28
- The User-Menu System 1-30
- A Complete Database Management System 1-31

Installing INFORMIX-SQL 1-31
Order of Installation 1-32
Establishing the Informix User, Group, and Directory 1-32
Setting Informix Environment Variables 1-33
Creating and Using a Setup File 1-34
Installing INFORMIX-SQL 1-35

Preparing to Use INFORMIX-SQL 1-38
Setting Environment Variables 1-38
Creating the Demonstration Database and Examples 1-41
Restricting Database Access 1-41

In This Chapter

This chapter introduces INFORMIX-SQL and considers important database management concepts. The topics discussed include:

- What is INFORMIX-SQL?
- What is a database?
- What are tables, rows, and columns?
- What is a relational database management system?
- What is a screen form?
- What is a report?
- What is a database query?

To make the best use of INFORMIX-SQL, you must understand the basic concepts in this chapter. "What Is a Database?" on page 1-4 is particularly helpful if you have had limited experience with computers or database management systems.

This chapter concludes with the following sections:

- Installing INFORMIX-SQL
- Preparing to use INFORMIX-SQL

What Is INFORMIX-SQL?

INFORMIX-SQL is a computer-based record-keeping system. As a *database management system*, INFORMIX-SQL consists of useful programs or modules that perform data management tasks. INFORMIX-SQL can substantially reduce the amount of time required to organize, store, and retrieve information. It can summarize, group, and format information in a variety of helpful ways.

You can use INFORMIX-SQL to organize and store data, then view it in whatever format is useful to you. For example, you can store all the names, addresses, birthdays, and phone numbers of your friends, then use INFORMIX-SQL to search through the database for all the Hunts, everyone with a birthday in March, all the Sutherlands on Cambridge Avenue with birthdays in May, and so on.

What Is a Database?

A *database* is a collection of information or *data*. The telephone directory is a database; its data is the names, addresses, and telephone numbers of local residents and businesses. Other kinds of databases include:

- Mailing lists
- Inventory lists
- Customer lists

These lists are called databases because they consist of information that is useful to a particular organization, or information that is used for a specific purpose.

A telephone directory is a good example of a database. Here is an example.

Last Name	First Name	Address	Telephone
Hunt	Eloise	15 Emerson	415-987-6543
Hunt	Frank	1102 University	415-123-4567
Hunt	Hubert	95A Sacramento	415-254-1107
Hunt	Inez	99 California Ave	415-786-1111
Hunt	Jason	60 Oakdale	415-454-9087

The telephone directory is organized alphabetically by last name. If you want to call Jason Hunt, you look in the Last Name column under *H*. As long as you know the last name of the person you want to call, this scheme works.

But what about finding phone numbers for everyone who lives on California Avenue? Or the phone number for someone whose last name you do not know? Because the telephone directory is not organized by street addresses or first names, it would take a long time to find these numbers.

This is where INFORMIX-SQL can help.

How INFORMIX-SQL Stores Data

INFORMIX-SQL stores information in tables. A *table* is a collection of rows and columns. Tables, rows, and columns are the building blocks that make up an INFORMIX-SQL database. This section explains these basic elements and how INFORMIX-SQL uses them.

A Database Is a Collection of Tables

A database is made up of tables. A table is a collection of information organized in *rows* and *columns*. A database contains at least one table and can contain as many tables as you need. Some database management systems refer to tables as *files*.

Each table in a database normally contains a different kind of information. For example, you would probably maintain separate tables for the products that you sell, the orders that you take, and the customers that you serve. You need to store different information for products, orders, and customers; that is why you create separate tables. At any time, you can add a new table or delete a table you no longer need.

The INFORMIX-SQL demonstration database, called **stores7**, contains information about a fictitious wholesale sporting goods distributor. This database contains the following tables:

- The **customer** table contains information about the retail sporting goods stores that purchase equipment from the distributor.
- The **orders** table contains information about recent purchases made by each retail sporting goods store.
- The **items** table contains information about the individual items contained in each order placed by the retail sporting goods stores.
- The **stock** table contains information about the items carried in stock by the distributor.
- The **manufact** table contains information about the various manufacturers who supply the distributor with products.
- The **state** table contains the names and postal abbreviations of the 50 states in the United States.
- The **catalog** table describes each item in stock.
- The **cust_calls** table contains information on telephone calls about orders, shipments, or complaints that are logged.

Tip: *The **stores7** demonstration database contains seven tables for INFORMIX-SE and eight tables for Informix Dynamic Server. If you are using INFORMIX-SE as your database server, **stores7** does not contain the **catalog** table.*

Tables Are Made Up of Rows and Columns

INFORMIX-SQL tables consist of rows and columns, just like the tables you see every day in books and newspapers and on the evening news. Some database management systems refer to rows as *records* and to columns as *fields*.

Each row contains all the data about one of the objects the table describes. For example, in the **stores7** demonstration database, the following statements are true:

- Each row in the **customer** table contains all the information about one retail customer.
- Each row in the **orders** table contains all the information about one purchase made by a retail sporting goods store.
- Each row in the **items** table contains all the information about one item in an order placed by a retail sporting goods store.
- Each row in the **stock** table contains all the information about one of the distributor's products.
- Each row in the **manufact** table contains all the information about one manufacturer.
- Each row in the **state** table contains all the information about one state in the United States.
- Each row in the **cust_calls** table contains information about one telephone call from a customer.
- Each row in the **catalog** table contains information about one of the items in stock.

When you create a new table, it contains no rows. It consists only of the column names and contains no data. When you begin to store data in a table, you usually add one row at a time.

Each table includes one or more columns. A column contains a particular type of information, such as a last name, order number, or zip code. When you create a database table, you assign a name to each column. These *column names* are similar to the headings that appear at the top of each column in a printed table. You use the column names to refer to the columns.

For example, in the **stores7** demonstration database, each row in the **customer** table contains 10 columns. Figure 1-1 lists the names of the columns and the information they contain.

Figure 1-1
Columns in the customer Table

This Column	Contains This Information
customer_num	Customer number
fname	Representative's first name
lname	Representative's last name
company	Name of store
address1	Store address, first line
address2	Store address, second line
city	City
state	State
zipcode	Zip code
phone	Phone number

Figure 1-2 shows some of the data from the **customer** table organized into rows and columns.

Figure 1-2
Data in the customer Table

customer_num	fname	lname	company	address1
101	Ludwig	Pauli	All Sports Supplies	213 Erstwild Court
102	Carole	Sadler	Sports Spot	785 Geary St.
103	Philip	Currie	Phil's Sports	654 Poplar
104	Anthony	Higgins	Play Ball!	East Shopping Cntr
105	Raymond	Vector	Los Altos Sports	1899 La Loma Drive

Retrieving Data from the Database

After you store some data in a database, you can use INFORMIX-SQL to ask questions about the data. This process is called *querying* the database. Querying means searching for information that is stored in the database.

You can query an INFORMIX-SQL database in the following ways:

- Search for a particular row or group of rows
- Search for a particular column or group of columns
- Combine both of these techniques to search for specific columns in specific rows

Additionally, you can use a technique called *joining* to retrieve data from more than one table at a time. The ability to retrieve information from multiple tables is one of the many powerful facilities of INFORMIX-SQL.

Searching for Rows

When you search for a row or group of rows in a database, INFORMIX-SQL displays all the information from each row. For example, you might want to show:

- All the information for the customer store named *All Sports Supplies*
- Information for customers who have stores located in Redwood City
- Information for customers who have stores located within the 94020 to 94040 zip code range

Each of these queries asks INFORMIX-SQL to display all the information in one or more rows. Some database management systems call this type of query *selection*.

Searching for Columns

You also can search for one or more columns. For example, you might want to list customer names and phone numbers without including addresses and zip codes. Alternatively, you might want to show the following information:

- The names of all stores and the cities in which they are located
- All store names, addresses, and phone numbers
- The zip codes of all stores

Each of these queries looks for all values in a particular column or group of columns. Some database management systems call this type of query a *projection*.

Searching for Columns Within Rows

More frequently, you combine two techniques to find the data you want. For example, you can show the following information:

- The name and phone number of each customer in the city of San Francisco. (INFORMIX-SQL finds the name and phone number columns from the rows for customers located in San Francisco.)
- The names of customers with stores in the 408 telephone area code. (INFORMIX-SQL finds the name columns from the rows for customers with telephone numbers that begin with the 408 area code.)
- The name and address of the store that has been assigned customer number 101. (INFORMIX-SQL finds the name and address columns from the row of the store assigned customer number 101.)

Each of these queries requests some, but not all, of the information stored in the **customer** table.

Connecting Tables Through Column Joins

INFORMIX-SQL is a *relational* database management system. Unlike a file management system, a relational database management system lets you create direct *relationships* between tables when you query a database.

With INFORMIX-SQL, you do not need to duplicate the same information in different tables. Instead, you use a technique called *joining* to connect data from different tables. Joining lets you look at data stored in several tables as if it were all part of a single table.

The **stores7** demonstration database provides a good example of the use of joins. You might periodically want to print a list showing each customer's name, address, phone number, and a list of each customer's recent orders. You could do this by storing complete customer information and order information in the **orders** table. Doing so, however, would make maintaining the database more time-consuming and prone to errors.

The **customer** table contains information about the name, location, and phone number of each sporting goods customer. If you also stored this information in the **orders** table, you would have the name, location, and phone number stored in two separate tables. This means you would need to update both the **customer** and **orders** tables whenever a customer changed an address or a phone number. Entering the same data in two tables not only is inconvenient, it also creates greater opportunities for data-entry error.

However, duplicate table information is unnecessary with INFORMIX-SQL. You can obtain the list of recent customer orders more easily by joining tables.

Tables can be joined if one column in each table contains the same (or similar) data. This column is called the *join column*, and you use it to create a temporary link—or join—between the tables.

For example, the **customer** and **orders** tables both contain a **customer_num** (customer number) column. You can use this column as a join column. In the **customer** table, this column shows the unique customer number assigned to each sporting goods retailer. In the **orders** table, it shows which customer placed the order described in the row. By joining the two **customer_num** columns, you can obtain the desired list of customer name, address, and phone number (retrieved from the **customer** table) and orders (retrieved from the **orders** table) without duplicating information in the two tables. This joining process is illustrated in Figure 1-3.

customer Table (detail)

customer_num	fname	lname
101	Ludwig	Pauli
102	Carole	Sadler
103	Philip	Currie
104	Anthony	Higgins

orders Table (detail)

order_num	order_date	customer_num
1001	05/20/1998	104
1002	05/21/1998	101
1003	05/22/1998	104
1004	05/22/1998	106

Figure 1-3
Tables Joined by the customer_num Column

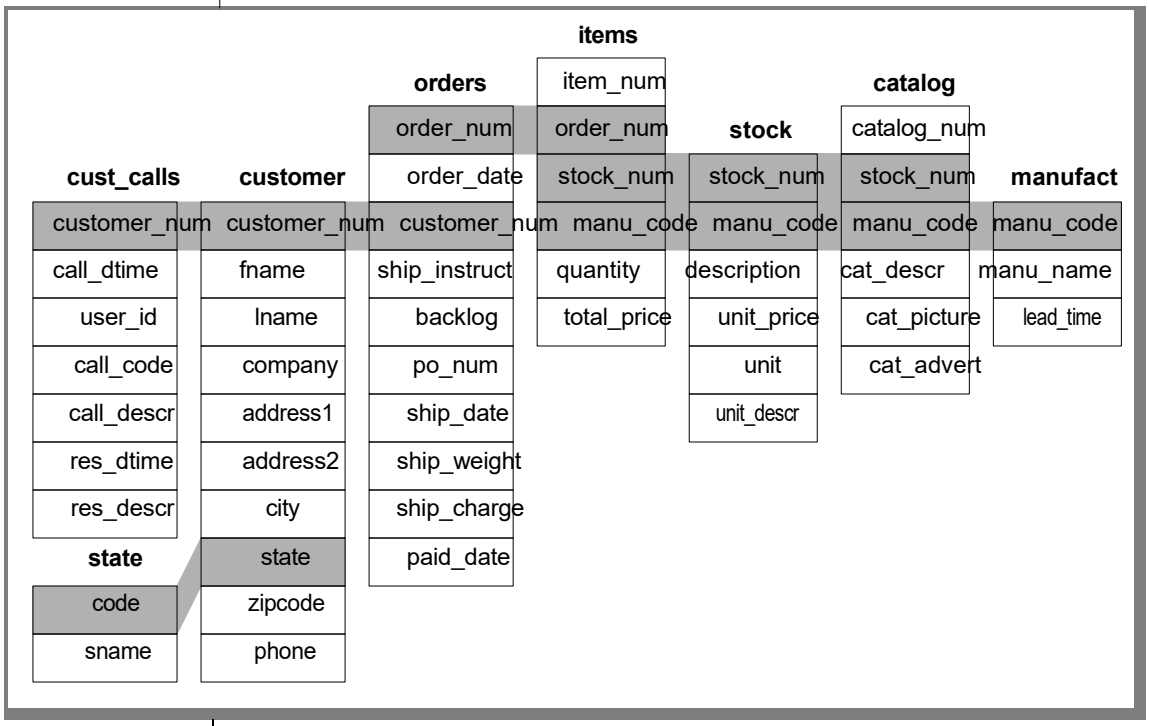
The row in the **customer** table with information about Anthony Higgins contains the number 104 in the **customer_num** column. The **orders** table also contains a **customer_num** column that stores the number of the customer who placed a particular order. According to Figure 1-3, customer 104 (Anthony Higgins) placed two orders because his customer number appears in two rows in the **orders** table.

One of the most important features of a truly relational database management system is the ability to join data from different tables. Instead of storing identical data in several tables, a relational database management system lets you retrieve data from several tables at once and display it as if it were stored in a *single* table.

Joining lets you rearrange your view of a database whenever you want. This flexibility lets you create new relationships between tables without redesigning your database. You can easily expand the scope of a database by adding new tables that join existing tables.

The tables of the **stores7** demonstration database are linked together by potential join columns, as Figure 1-4 shows.

Figure 1-4
Join Relationships in the stores7 Demonstration Database



The *Informix Guide to SQL: Reference* includes a section discussing the join columns that link the tables in the demonstration database.

Getting Started with INFORMIX-SQL

To use INFORMIX-SQL, you need a database with which to work. The examples in this book are based on the **stores7** demonstration database. You can follow the examples if you create a copy of the **stores7** demonstration database, as explained in the *INFORMIX-SQL Reference Manual*.

Before you can start working with INFORMIX-SQL, be sure that your computer is up and running, and that the operating-system prompt appears on your screen.

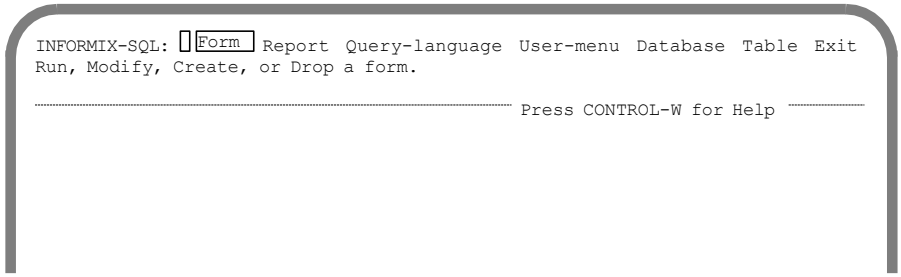
Enter the following command:

```
isql
```

This operation displays "The INFORMIX-SQL Main Menu" as described on page 1-14.

The INFORMIX-SQL Main Menu

The Informix logo is displayed followed by the INFORMIX-SQL Main menu.



The INFORMIX-SQL Main menu displays a series of options across the top of the screen. These options represent the major INFORMIX-SQL functions.

The Current Option

One option is always marked as the *current option*. It is displayed in reverse video (dark letters on a light background) if your terminal can display reverse video, or surrounded by angle brackets < > if it cannot.

The screen illustrations in this manual show the reverse video as a box that surrounds the highlighted part of the screen. In the preceding figure, Form is the current option.

The Message Line

A message appears on the line immediately beneath the list of options. The message describes the current option. In the previous figure, the message:

```
Run, Modify, Create, or Drop a form.
```

describes the activities available to you when you select the Form option on the INFORMIX-SQL Main menu. You can:

- *Run* an existing form. (Use a screen form to enter or retrieve data.)
- *Modify* an existing form. (Alter the appearance or behavior of the screen form.)
- *Create* a new screen form.
- *Drop* (delete) an existing form.

Moving the Highlight

To move the highlight, press the SPACEBAR or the Arrow keys. As you move the highlight, the message displayed on the Message line beneath the options changes. This message always describes the highlighted option. If you highlight the Report option, you see the following message:

```
Run, Modify, Create, or Drop a report.
```

If you cannot remember what an option does, just highlight it and read the message.

Selecting an Option

After you highlight the option you want, you can select it by pressing RETURN. Pressing RETURN tells INFORMIX-SQL you want to work with the option selected. For example, if you want to select, create, or drop a database, move the highlight to the Database option and press RETURN.

You also can select an option by typing the first letter of its name. Typing the first letter selects the option right away; you do not have to move the highlight or press RETURN. For example, no matter where the highlight appears on the Main menu, you can always select the Exit option by pressing the `e` key.

What the Main Menu Options Mean

The options on the Main menu give you access to all the INFORMIX-SQL functions. Select an option on the Main menu, and INFORMIX-SQL displays a new menu of options relating to your choice. For example, if you select the Table option, you see another menu that lets you specify whether you want to create, alter, display information about, or drop a database table.

Here is a summary of the INFORMIX-SQL Main menu options:

Form Displays the FORM menu. Use this menu to work with screen forms. Screen forms are described in Chapter 3 through Chapter 6.

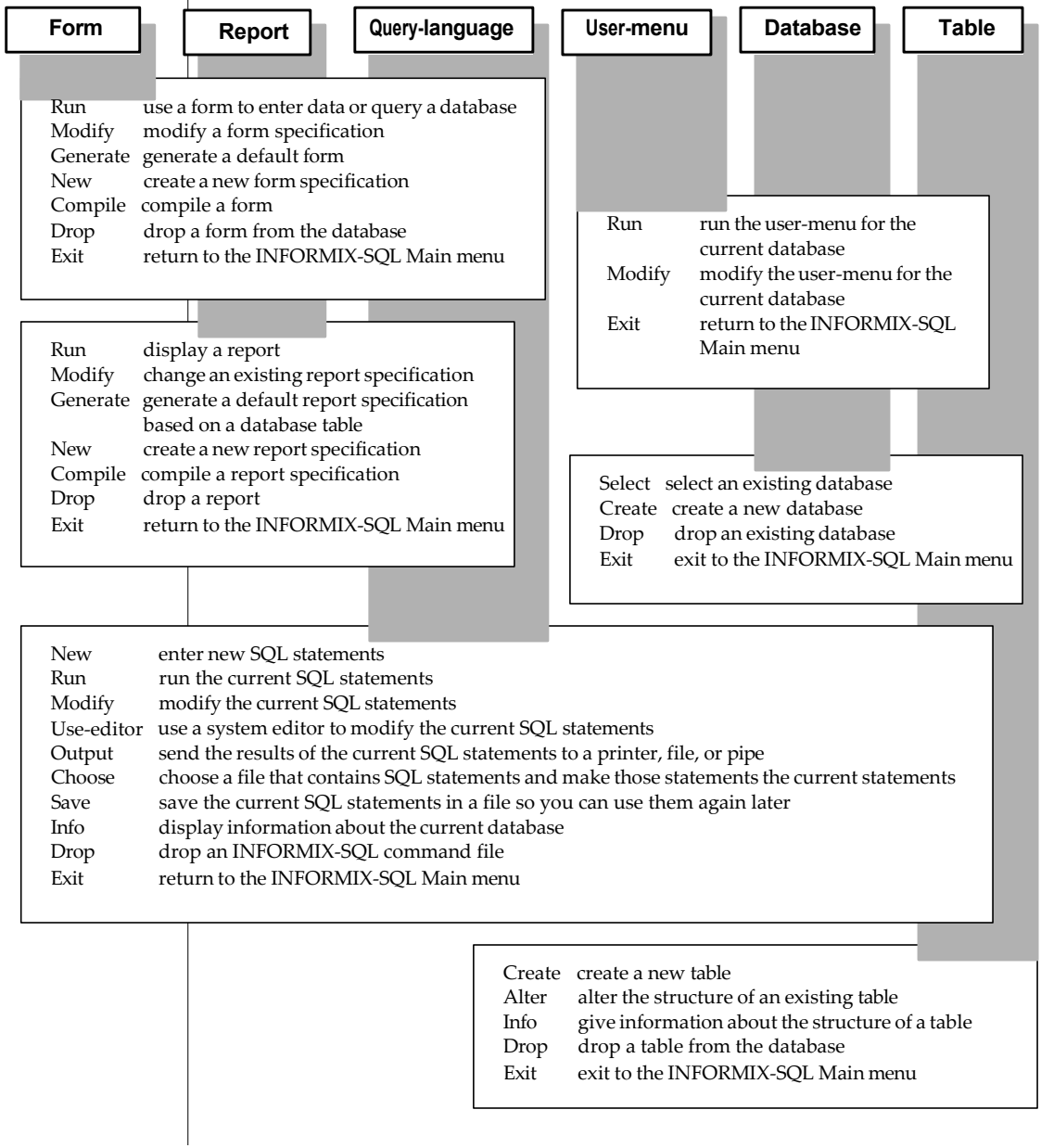
Report Displays the REPORT menu. Use this menu to work with reports. A report selects data from one or more tables in a database and prints it in the format you specify. See Chapter 8, "Creating and Printing Reports," for details.

Query-language Displays the SQL menu. Use this menu to work with the Informix structured query language. You use this language to search a database for information, add information to a database, and change the structure of a database. See Chapter 7, "Using SQL," for details.

User-menu	Displays the USER-MENU menu. Use this menu to work with custom screen menus. These menus are created through the procedures described in the *INFORMIX-SQL Reference Manual*.
Database	Displays the DATABASE menu. Use this menu to select, create, or drop a database.
Table	Displays the TABLE menu. Use this menu to create or alter a table.
Exit	Exits INFORMIX-SQL and returns to the operating system.

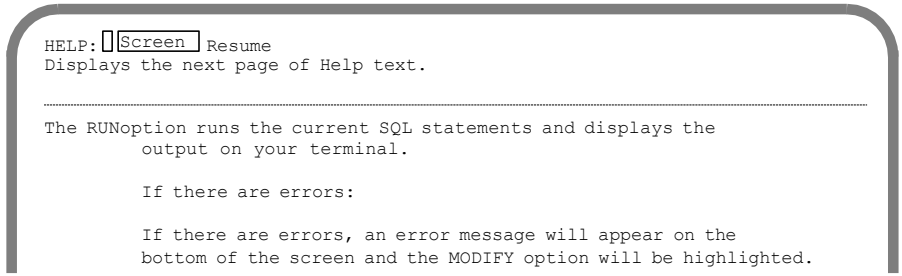
Figure 1-5 illustrates the INFORMIX-SQL menu structure.

Figure 1-5
The INFORMIX-SQL Menu Structure



Using the HELP Menu

Every INFORMIX-SQL screen has a HELP menu associated with it. The HELP menu contains information about your current options and suggests appropriate actions. Whenever you want information about what to do next, press the CONTROL-W key, and INFORMIX-SQL displays the HELP menu.



You can read as much or as little text as you need. Selecting the Screen option on the HELP menu displays the next page of the HELP text; selecting the Resume option lets you continue your work.

You use the HELP menu in the same way you use other INFORMIX-SQL menus. Press the SPACEBAR or arrow keys to move the highlight to the option you want and then press RETURN. You also can enter the first letter of an option.

The INFORMIX-SQL Facilities

INFORMIX-SQL provides a complete set of database management facilities that allow you to perform the following tasks:

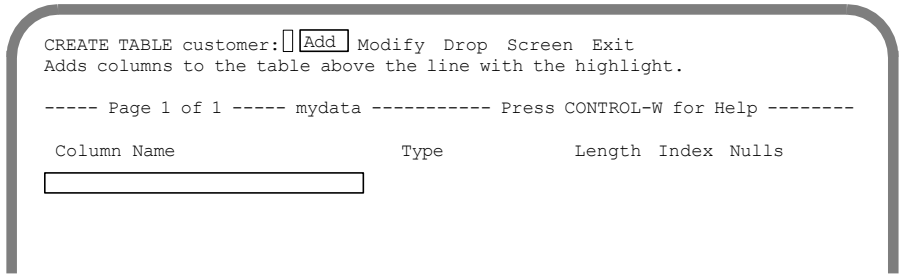
- Create and modify tables using the menus provided with the *schema editor*
- Enter and retrieve database information using *screen forms*
- Sort, combine, format, and display data with *reports*
- Enter, modify, and retrieve database information using the *query language*
- Access INFORMIX-SQL through special-purpose, custom *menus*

Each of these features is demonstrated in this section.

Creating Tables with the Schema Editor

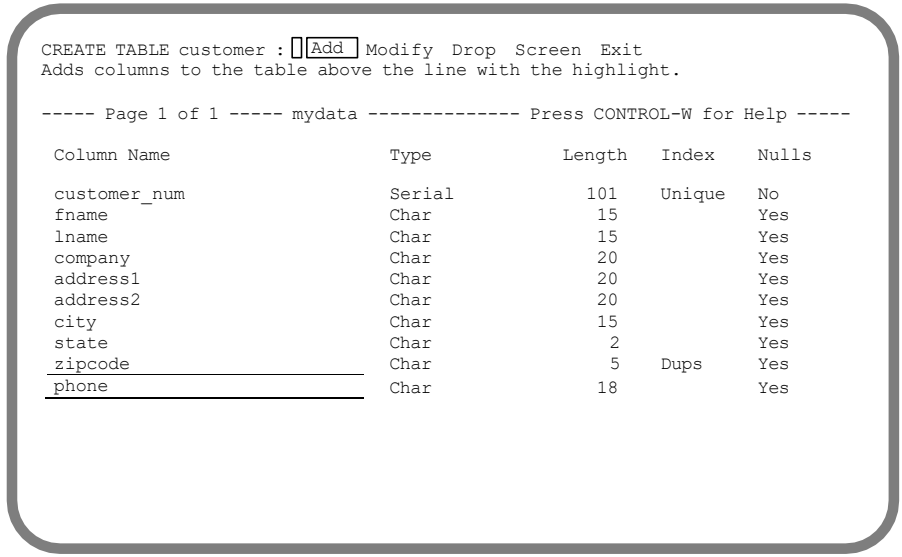
A database is made up of one or more tables. The INFORMIX-SQL schema editor provides menus that guide you through the steps necessary to create each table in your database.

The CREATE TABLE menu lets you create the *schema* for a table on the bottom portion of the screen.



A schema defines the structure of a database table.

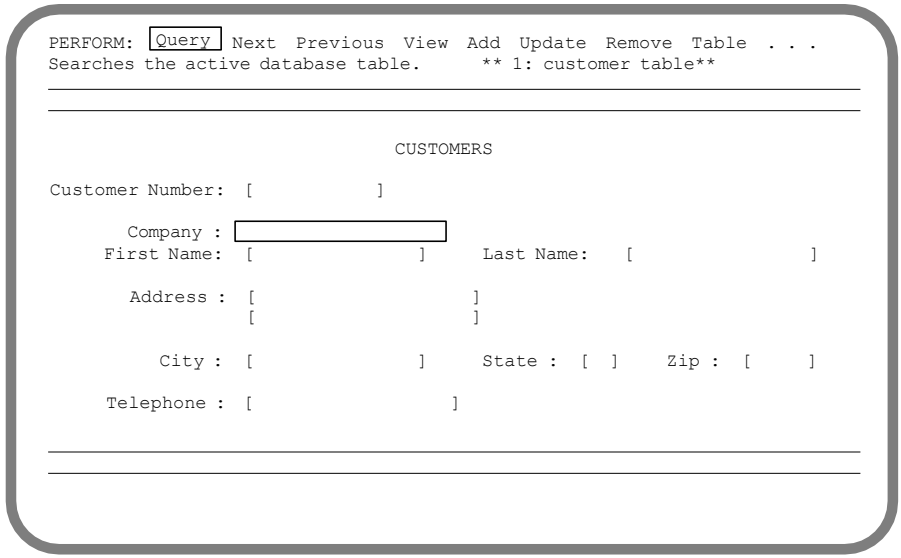
The finished table schema for the **customer** table (included in the demonstration database) appears as follows.



Using a Screen Form

Once you have created a table, you can store data in it. The easiest way to enter the data you want to store is to use a *screen form*. Much like a printed form, a screen form contains blank spaces that you can fill in. The blank spaces on a screen form are called *fields*.

Included with the demonstration database application materials is a screen form named **customer** that you can use to enter data into the **customer** table. Here is what the **customer** form looks like.



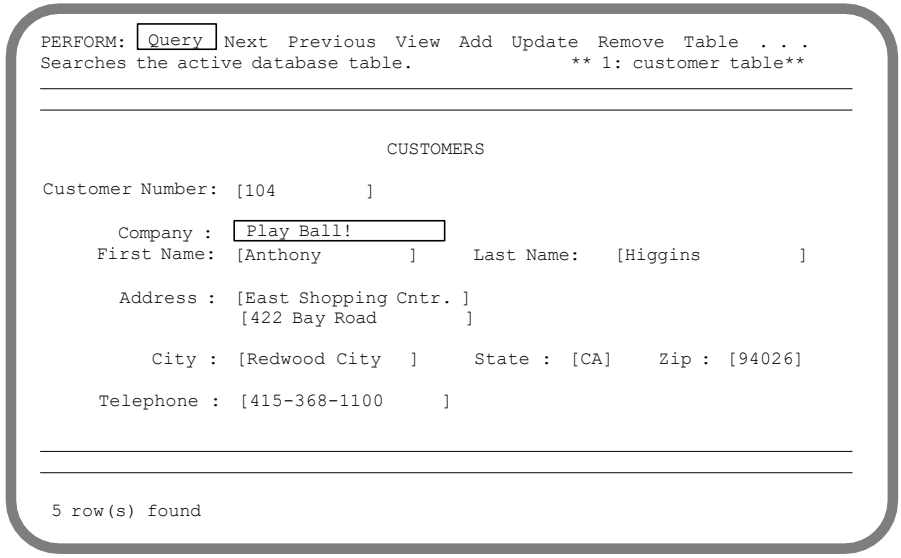
You add data to a table by entering it into the fields on a form. Each field normally corresponds to a column in one of your database tables. The data you enter into a field is stored in the corresponding column.

You can also use a screen form to query the database and retrieve information. For example, to find the rows for all customers with stores in Redwood City, you would complete the **customer** form as follows.

```
QUERY: ESCAPE queries.   INTERRUPT discards query.  ARROW keys move cursor.
Searches the active database table.          ** 1: customer table**
_____________________________________________________________________________
_____________________________________________________________________________

                                CUSTOMERS

 Customer Number: [           ]

        Company :
     First Name: [               ]     Last Name:   [               ]

       Address : [                   ]
                 [                   ]

          City : [Redwood City   ]     State : [ ]     Zip : [      ]

     Telephone : [                 ]

_____________________________________________________________________________
_____________________________________________________________________________
```

This screen instructs INFORMIX-SQL to search for rows in which the **city** column contains "Redwood City."

Here is what the **customer** form looks like when you execute the query.



The message at the bottom of the screen indicates that INFORMIX-SQL found five customers with locations in Redwood City. The data about the first customer appears on the screen.

The Report Writing Facility

A report consists of database information that is arranged and formatted according to your instructions and then displayed on the screen, printed on paper, or stored in a file for future use. You can select all or some of the information stored in a database and display it in a report. Alternatively, you can retrieve and display data from an ASCII input file.

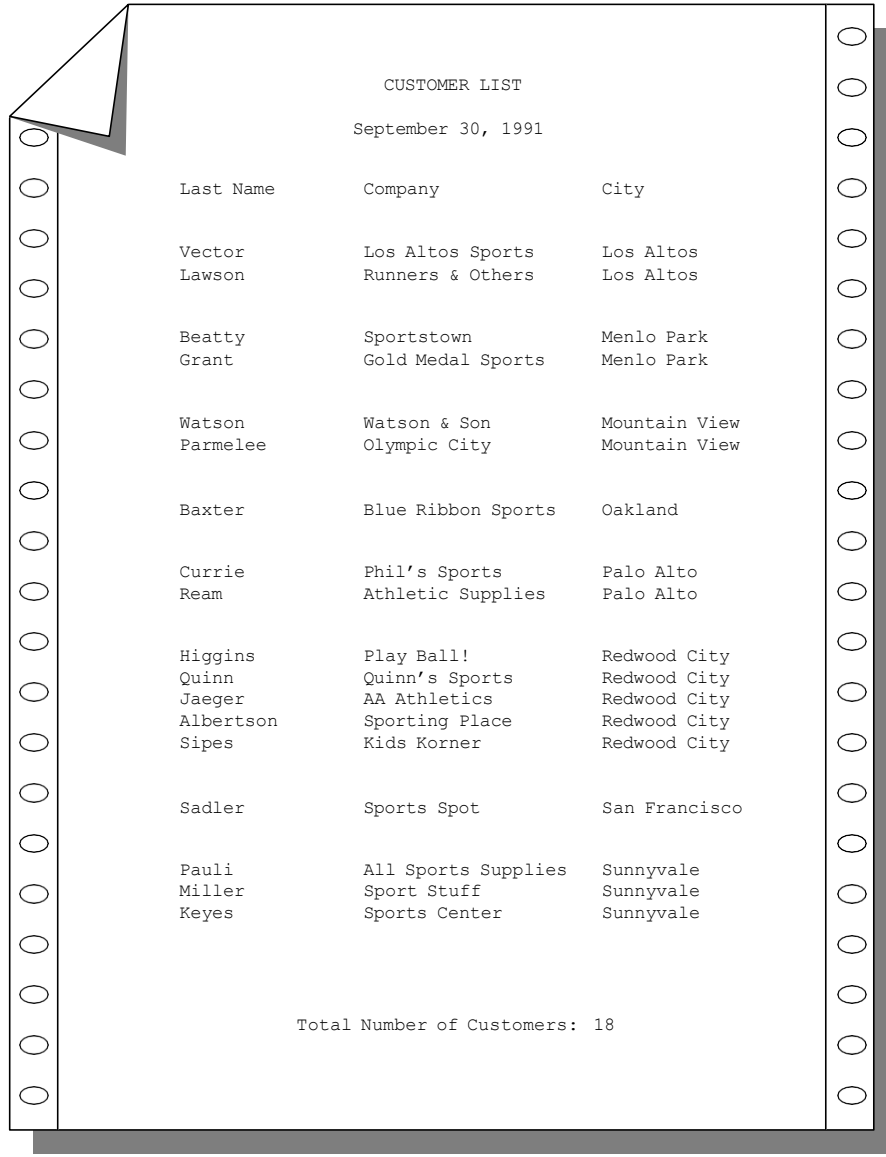
The INFORMIX-SQL report writer retrieves data from a database or input file, organizes and formats it according to your instructions, adds the headings or column titles you specify, performs calculations on number columns, and then displays the report.

You can combine information from several tables in a database or file to create whatever kind of report you need. Some of the reports INFORMIX-SQL can produce are as follows:

- Mailing labels
- Form letters
- Payroll summaries
- Medical histories
- Accounting records
- Inventory lists
- Telephone directories
- Project schedules
- Sales and marketing reports
- Payroll checks

The demonstration database application materials include a variety of reports that illustrate the uses of the INFORMIX-SQL report writing facility. The report in Figure 1-6 on page 1-26 lists the last name, company, and city for all customers. It groups customers by city and calculates the total number of customers.

Figure 1-6
A Sample Report



CUSTOMER LIST

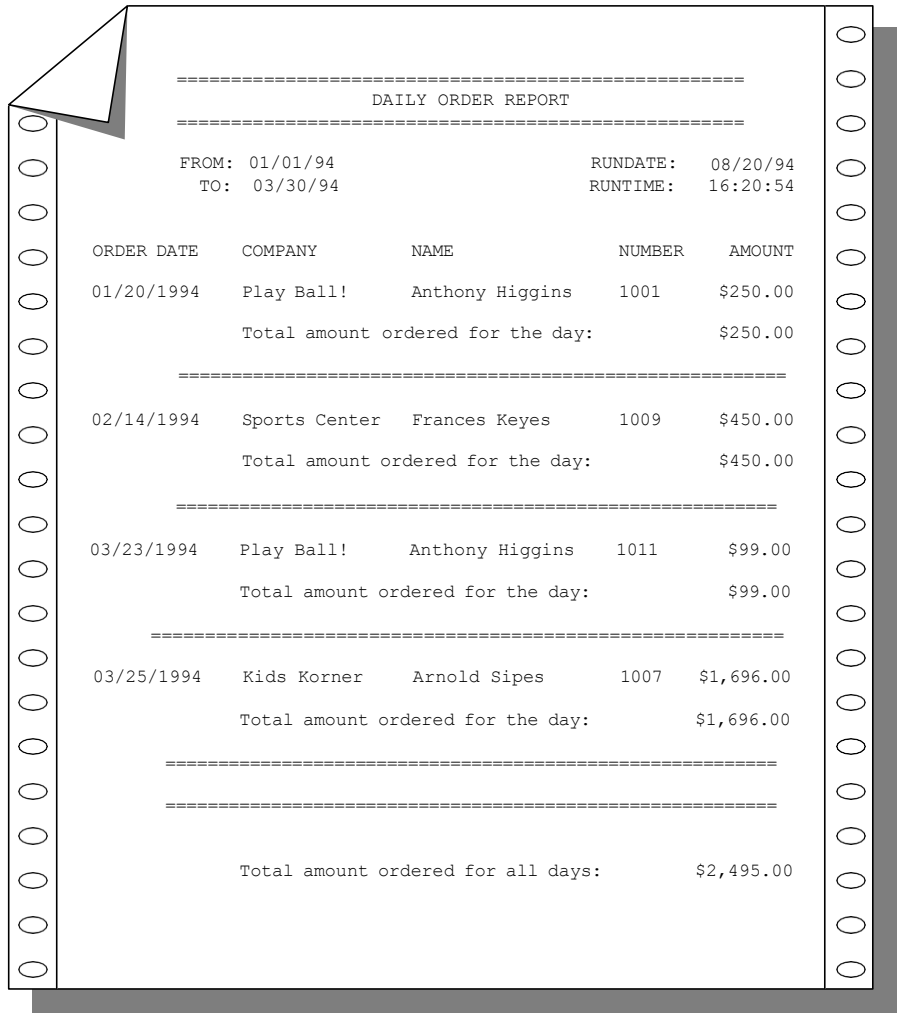
September 30, 1991

Last Name	Company	City
Vector	Los Altos Sports	Los Altos
Lawson	Runners & Others	Los Altos
Beatty	Sportstown	Menlo Park
Grant	Gold Medal Sports	Menlo Park
Watson	Watson & Son	Mountain View
Parmelee	Olympic City	Mountain View
Baxter	Blue Ribbon Sports	Oakland
Currie	Phil's Sports	Palo Alto
Ream	Athletic Supplies	Palo Alto
Higgins	Play Ball!	Redwood City
Quinn	Quinn's Sports	Redwood City
Jaeger	AA Athletics	Redwood City
Albertson	Sporting Place	Redwood City
Sipes	Kids Korner	Redwood City
Sadler	Sports Spot	San Francisco
Pauli	All Sports Supplies	Sunnyvale
Miller	Sport Stuff	Sunnyvale
Keyes	Sports Center	Sunnyvale

Total Number of Customers: 18

You also can prepare *interactive* reports with INFORMIX-SQL. The report in Figure 1-7 asks the user for beginning and ending dates and prepares a listing of daily orders. It then automatically computes and prints subtotals and totals.

Figure 1-7
Another Sample Report



```
                    DAILY ORDER REPORT

          FROM:  01/01/94                      RUNDATE:    08/20/94
            TO:  03/30/94                      RUNTIME:    16:20:54

ORDER DATE    COMPANY         NAME               NUMBER     AMOUNT

01/20/1994    Play Ball!      Anthony Higgins    1001       $250.00

              Total amount ordered for the day:             $250.00

02/14/1994    Sports Center   Frances Keyes      1009       $450.00

              Total amount ordered for the day:             $450.00

03/23/1994    Play Ball!      Anthony Higgins    1011        $99.00

              Total amount ordered for the day:              $99.00

03/25/1994    Kids Korner     Arnold Sipes       1007     $1,696.00

              Total amount ordered for the day:           $1,696.00

              Total amount ordered for all days:          $2,495.00
```

Using Structured Query Language

SQL is an English-like, interactive query language that you can use when working with databases. The SQL supplied with INFORMIX-SQL is an enhanced version of the industry-standard Structured Query Language (SQL) originally developed by HCL. With SQL, you can perform a wide variety of database management tasks, including:

- Creating and dropping tables
- Entering and deleting data
- Querying a database
- Renaming tables and columns

To use SQL, you enter one or more SQL *statements*. A statement is simply an instruction that tells INFORMIX-SQL what you want to do. For example, to create a table, you use the CREATE TABLE statement; to query a database, you use the SELECT statement.

The following statement was used to create the **stock** table in the demonstration database:

```
create  table  stock

(    stock_num      smallint,
     manu_code      char(3),
     description  char(15),
     unit_price     money(6),
     unit           char(4),
     unit_descr     char(15)
);
```

This statement tells INFORMIX-SQL to create a table named **stock** and specifies the name of each column and the type of data (numbers, words, money, or dates) each column will contain. Column names appear on the left; data types are indicated on the right.

The next example illustrates how database queries are easily achieved with SQL. In this example, all stock items relating to the sport of baseball are selected from the database, and a five percent increase in their unit price is calculated.

```
select stock_num, stock.manu_code,
   manu_name, description, unit_price,
   unit_price * 1.05 newprice
   from stock, manufact
   where stock.manu_code
   = manufact.manu_code and
   description matches "baseball*"
   order by stock.manu_code
```

The next figure shows the results of this query.

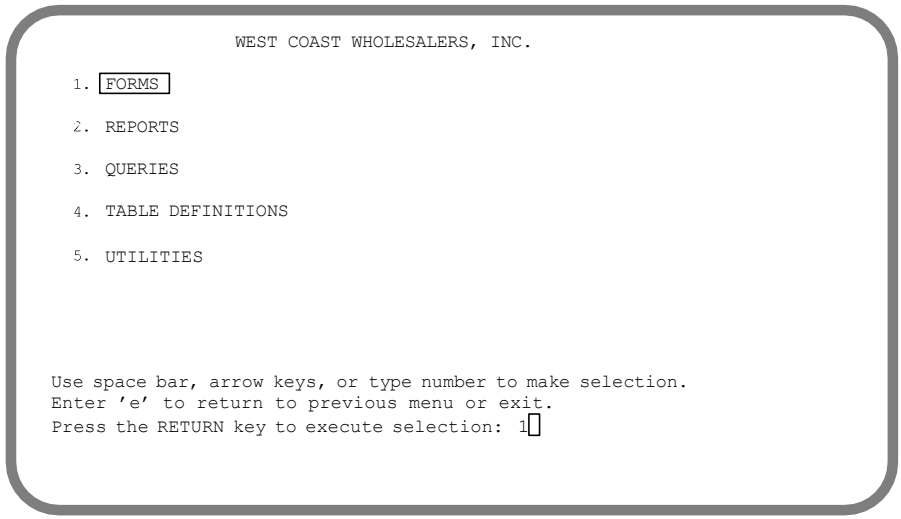
```
SQL: New  Run  Modify  Use-editor  Output  Choose  Save  Info  Drop  Exit
Run the current SQL statements.

------------------------------ stores7 ------------------------------ Press CONTROL-W for Help-------------------------------

stock_num  manu_code manu_name    description       unit_price   newprice

        1  HRO       Hero         baseball   gloves   $250.00   $262.5000
        2  HRO       Hero         baseball             $126.00   $132.3000
        1  HSK       Husky        baseball   gloves   $800.00   $840.0000
        3  HSK       Husky        baseball    bat    $240.00    $252.0000
        1  SMT       Smith        baseball gloves     $450.00   $472.5000




5 row(s) retrieved.
```

The User-Menu System

You can create custom menus that allow individuals with limited experience to use sophisticated INFORMIX-SQL database management applications. Use the INFORMIX-SQL user-menu facility to create and run custom menus. The options on a user-menu can call:

- submenus on the custom menu system.
- INFORMIX-SQL programs, such as reports, screen forms, and SQL queries.
- other programs or sets of programs in your software library.
- operating-system utilities.

The main level of the user-menu included with the demonstration database application materials follows.



A Complete Database Management System

The INFORMIX-SQL facilities equip you to perform all your database management tasks. You can:

- easily add new tables to a database.
- modify existing tables with the schema editor.
- add, update, and delete data with screen forms.
- use the SQL query language to enter and retrieve database information and to modify the structure of the database.
- display database information in a variety of formats with reports.
- create custom menus that provide all users with easy access to INFORMIX-SQL programs.

Installing INFORMIX-SQL

This section explains how to install your INFORMIX-SQL software. It includes the following sections:

- Order of installation
- Establishing the Informix user, group, and directory
- Setting Informix environment variables
- Installing INFORMIX-SQL

INFORMIX-SQL includes a demonstration database called **stores7**. For information about how to build this sample database and examples, see "Creating the Demonstration Database and Examples" on page 1-40.

If you are also installing a database server, you will need to install your database server after you have installed INFORMIX-SQL. For more information, check the *Administrator's Guide* for your database server.

Order of Installation

If you install more than one Informix product, you must install them in a specific order, as follows:

1. Application development tools

   Install application development tools in order, from oldest to newest versions.

2. SQL application programming interfaces (APIs)

   Install SQL APIs in order, from oldest to newest versions.

3. Database servers

   Install database servers in order, from oldest to newest versions. Generally, a database server should be installed in the same directory as any SQL API or application development tools that use it. Therefore, verify that **$INFORMIXDIR** is set to the common directory before you install these types of products.

4. Additional products

   Install remaining interface and documentation products in order, from oldest to newest versions.

Establishing the Informix User, Group, and Directory

If this is the first time that you will install an Informix product on your computer, you need to create new user and group accounts and create a new directory. Before you begin, keep the following considerations in mind:

- Know who you are logged in as. Typically, you need to be able to log in as the following users:
  - Informix: to run the Informix database servers and software.
  - Yourself: to perform administrative work on your computer, such as copying or untarring files.
  - Root: to edit system files and install the Informix software.
- The user **informix** is the database equivalent of the UNIX **root** account, so anyone logged in as **informix** can completely access any Informix products and databases. Make sure you keep the password for user **informix** confidential.

- Informix products use group **informix** internally to control database access. Make user **informix** the only member of group **informix**. If you make a user of an Informix product a member of group **informix**, you can cause unintended and uncontrolled database access.
- If you are using a network, make sure you propagate the new user name to all the systems on the network. For example, you (or the network administrator) might need to run the **ypmake** utility.
- Although Informix uses **/usr/informix** as the directory path and name, you can install the software in any directory. If you decide to use a different directory, substitute that directory path and name throughout this guide whenever you see **/usr/informix**.

To establish the Informix user, group, and directory

1. Create a new group called **informix** in the **group** file (**/etc/group** on most UNIX-based systems).

   Generally, you should provide an unused group number greater than or equal to 100. If necessary, see your system administrator or operating system manual for assistance.
2. Add a new user called **informix** to the **/etc/passwd** file and assign the user to the group **informix**.

   Generally, you should provide a user ID number greater than or equal to 100 for that user and include a password for user **informix**.
3. Create a new directory for your INFORMIX-SQL software by entering a command such as:

   ```
   mkdir /usr/informix
   ```

Setting Informix Environment Variables

Before you can use any Informix product, set the following environment variables correctly:

- **INFORMIXDIR** specifies the location of the client programs, library files, message files, header files, and other Informix software components. For example:

  ```
  setenv INFORMIXDIR /extra/isql
  ```

- **PATH${PATH}:${INFORMIXDIR}/bin** specifies the location of the binaries for your Informix software. For example:

```
setenv  PATH  ${PATH}:/extra/isql/bin
```

- **INFORMIXSERVER** specifies the arbitrary, logical name you supplied for your Informix database server. This name is used to establish connectivity in client/server configurations. For example:

```
setenv  INFORMIXSERVER  cardtrick_ol
```

- **INFORMIXSQLHOSTS** sets the path to the **etc/sqlhosts** file within your Informix database software. For example:

```
setenv   INFORMIXSQLHOSTS   /extra/isql/etc/sqlhosts
```

- **LD_LIBRARY_PATH** sets the path to the library file that the Informix database software uses:.

```
setenv  LD_LIBRARY_PATH
$INFORMIXDIR/lib:4INFORMIXDIR/lib/
esql:$INFORMIXDIR/lib/tools
```

For more information about how to set Informix environment variables, see the *Informix Guide to SQL: Reference*. If you need to modify the **termcap** file, modify the file in another directory and reference that directory.

Creating and Using a Setup File

You can create a setup file that stores the environment variables. You must run the environment variables in the window from which you are running an Informix product. You can then source the file to apply the environment variables.

To create and use a setup file

1. Create a setup file using either the **Text Editor** or **vi** editor.
2. Enter your current environment variables into the setup file.

   For example, a setup file might include the following environment variables:

```
setenv  INFORMIXDIR  /extra/isql
setenv PATH ${PATH}:/extra/isql/bin
setenv  INFORMIXSERVER  cardtrick_ol
setenv   INFORMIXSQLHOSTS   /extra/isql/etc/sqlhosts
```

3. Save the file.

4. Each time you open a new shell window, source the setup file. For example:

```
source setup
```

5. View the environment variables to make sure they are set correctly.

   It is always a good idea to check the environment variables. Sometimes UNIX might not allow you to source a file. If this occurs, you might have to log in as another user to source the file.

```
env
```

Installing INFORMIX-SQL

This section contains instructions for loading and installing INFORMIX-SQL. If you are also installing a database server, review the installation and configuration information in the *Administrator's Guide* for your version of the database server before you proceed with the steps in this section.

Before you begin, you will need:

- INFORMIX-SQL software media
- Serial number keycard
- Your device name

Your Informix product materials include a serial number keycard and CD-ROM or other electronic media that contain all the product files. Both are necessary for installation. If you do not have the serial number keycard or the proper media, contact your supplier or Informix sales representative.

You will need the serial number keycard to perform the installation procedure that follows. Your serial number keycard provides the correct command for copying the Informix files onto the hard disk of your computer. It lists a version of the **tar** command similar to the following example:

```
tar xvf[b] devicename [20]
```

where *devicename* refers to the full pathname to that device. Devices are commonly in **/dev**, so the name is normally **/dev/***devicename*.

To install INFORMIX-SQL

1. Log in as root.
2. Load the CD-ROM or other electronic media supplied with your software into the appropriate drive in your computer.
3. If you are not currently in the **$INFORMIXDIR** directory, enter the following command:

```
cd $INFORMIXDIR
```

4. Copy the software files from the CD-ROM or other electronic media to the current directory by entering the **tar** command listed on your serial number keycard.

   For example:

```
./tar  -xzf  isql-7.20.UD1.tar
```

   After you issue the **tar** command, you should see a set of directories similar to the following example:

```
axis%  ls

bin                     gls                     lib
demo                    incl                    msg
etc                     installsql              release
forms                   isql-7.20.UD1.tar.gz    tar
```

5. Enter the installation command:

```
./installsql
```

6. The following message appears on the screen:

```
Installation  Script

This installation procedure must be run by root (super-
user). It will change the owner, group, and mode of all
files of this package in this directory. There must be
a user "informix" and a group "informix" known to the
system.
Press RETURN to continue,
or the interrupt key (usually CTRL-C or DEL) to abort.
```

7. Press RETURN to continue the installation procedure.

   The following prompt appears:

```
Enter your serial number (e.g., INF#X999999) >
```

8. Enter the 11-character serial number located on your serial number keycard.

   The serial number is 3 uppercase letters, followed by a pound sign (#), followed by 1 uppercase letter and 6 digits.

   The following prompt appears:

   ```
   Enter your serial number KEY (uppercase letters only)>
   ```

9. Enter the 6-letter key from the serial number keycard.

   You then see a message similar to the following example:

   ```
   WARNING!
   This software, and its authorized use and number of
   users, are subject to the applicable license agreement
   with Informix Software, Inc. If the number of users
   exceeds the licensed number, the excess users may be
   prevented from using the software. UNAUTHORIZED USE OR
   COPYING MAY SUBJECT YOU AND YOUR COMPANY TO SEVERE
   CIVIL AND CRIMINAL LIABILITIES.
   ```

   If your software is licensed for use by an unlimited number of simultaneous users, you see a message to that effect.

10. Press RETURN to begin the installation.

    A series of messages appears on the screen as each directory is installed:

    ```
    Installing directory .
    Installing directory bin
    Installing directory lib
              .
              .
              .
    ```

    The following message tells you that your Informix product is fully installed:

    ```
    Installation of INFORMIX-SQL complete
    ```

    After this message appears, you see the shell prompt, indicating completion of installation.

11. Repeat steps 4 through 8 until each file has been executed.

12. Log out as **root** after you complete product installation.

Preparing to Use INFORMIX-SQL

This section describes the steps you must follow before running INFORMIX-SQL, summarizes the conventions for using your terminal with the program, and explains how to use the demonstration database provided as part of INFORMIX-SQL. It also explains how INFORMIX-SQL uses operating-system file and directory permissions to control access to databases, and the ways in which operating-system constraints can affect the performance of the program.

Setting Environment Variables

Before you can use INFORMIX-SQL, be sure that your computing environment is set up as follows:

- Set the **INFORMIXDIR** environment variable so that INFORMIX-SQL can locate its programs.
- Set the **TERM** environment variable so that INFORMIX-SQL recognizes the kind of terminal you are using.
- Set the **TERMCAP** or **TERMINFO** environment variable so that INFORMIX-SQL can communicate with your terminal.
- Set the **INFORMIXTERM** environment variable to specify whether INFORMIX-SQL should use the **termcap** or **terminfo** file to locate terminal capability information.
- Set the **PATH** environment variable so that the shell searches the correct bin for executable INFORMIX-SQL programs.

Some of these variables might already be set if you have just finished installing INFORMIX-SQL.

You can set these environment variables at the system prompt or in your **.profile** (Bourne shell) or **.login** (C shell) file. If you set these variables at the system prompt, you will have to assign them again the next time you log onto the system. If you set these variables in your **.profile** or **.login** file, the operating system assigns them automatically every time you log in.

For more detailed information on environment variables, refer to the *Informix Guide to SQL: Reference*.

The following set of instructions assumes that INFORMIX-SQL resides in the directory **/usr/informix**. If INFORMIX-SQL has been installed in another directory, substitute the new directory name for **/usr/informix**.

1. Log onto the system.
2. Set the **INFORMIXDIR** environment variable.

   If you are using the Bourne shell, enter:

   ```
   INFORMIXDIR=/usr/informix
   export INFORMIXDIR
   ```

   If you are using the C shell, enter:

   ```
   setenv  INFORMIXDIR  /usr/informix
   ```

3. Set the **TERM** environment variable.

   To do this, you need to obtain the code for your terminal from the system administrator.

   If you are using the Bourne shell, enter:

   ```
   TERM=name
   export TERM
   ```

   where *name* is the code for the terminal you are using.

   If you are using the C shell, enter:

   ```
   setenv  TERM  name
   ```

4. Set the **TERMCAP** or **TERMINFO** environment variable.

   If the code for your terminal is listed in the **TERMCAP** or **TERMINFO** files included with INFORMIX-SQL, follow these instructions. Otherwise, consult your system administrator about the procedure for your system.

   If you are using the Bourne shell, enter:

   ```
   TERMCAP=/usr/informix/etc/termcap
   export TERMCAP
   ```

   or

   ```
   TERMINFO=/usr/informix/lib/terminfo
   export TERMINFO
   ```

   If you are using the C shell, enter:

   ```
   setenv  TERMCAP  /usr/informix/etc/termcap
   ```

   or

   ```
   setenv  TERMINFO  /usr/informix/lib/terminfo
   ```

5. Set the **INFORMIXTERM** environment variable.

   You can set this variable to **termcap** or **terminfo**, depending on which you are using to define your terminal capabilities. If you are using the Bourne shell, enter:

   ```
   INFORMIXTERM  filename
   export  INFORMIXTERM
   ```

   where *filename* is either **termcap** or **terminfo.**

   If you are using the C shell, enter:

   ```
   setenv  INFORMIXTERM  filename
   ```

   If **INFORMIXTERM** is unset, INFORMIX-SQL uses the **termcap** file.

   The use of **termcap** and **terminfo** files is described in detail in the *INFORMIX-SQL Reference Manual*.

6. Include the directory that contains INFORMIX-SQL in your **PATH** environment variable.

   If you are using the Bourne shell, enter:

   ```
   PATH=$PATH:/usr/informix/bin
   export PATH
   ```

   If you are using the C shell, enter:

   ```
   setenv  PATH  ${PATH}:/usr/informix/bin
   ```

7. If you placed these environment settings in your **.login** or **.profile** file, log out and then log back in. (This procedure allows the shell to read the changes you have made.) The new settings take effect each time you log in.

   If you entered these environment settings from the system prompt, the new settings remain in effect until you log out. You need to re-enter them every time you log onto the system.

In addition to the environment variables listed in this section, you can assign other environment variables by following the instructions in the *Informix Guide to SQL: Reference*.

Creating the Demonstration Database and Examples

INFORMIX-SQL includes several examples and a demonstration database called **stores7** that contains information about a fictitious wholesale sporting-goods distributor. You can create the **stores7** database in any directory you like by changing to the directory and entering the following command:

```
isqldemo
```

Many (but not all) examples in the INFORMIX-SQL documentation set are based on the **stores7** database. This database is described in detail in the *Informix Guide to SQL: Reference*. The examples are installed with your software in the **$INFORMIXDIR/demo/sql** directory. For U.S. English, go to the **en_us/0333** subdirectory; for other languages, go to the appropriate subdirectory under the **fgl** directory.

Restricting Database Access

INFORMIX-SQL includes a set of statements used to ensure database security and data integrity. You can control database and table privileges independently, allowing only selected individuals to access certain tables (or parts of tables) and excluding other users from the database entirely. The SQL statements used for these purposes are described in the *Informix Guide to SQL: Tutorial*.

For INFORMIX-SQL to access a database, the current user must have execute (search) permission for each directory in the database pathname. Check with your system administrator for more information about file permissions.

Chapter

2

Creating a Database

In This Chapter 2-3

Two Ways to Create Databases and Tables 2-3
- Using the Main Menu to Create Databases and Tables 2-4
- Using SQL to Create Databases and Tables 2-4

Accessing INFORMIX-SQL 2-5

Using the Main Menu to Create a Database 2-5
- Guidelines for Selecting Database Names 2-7
  - Number of Characters 2-7
  - Characters You Can Use 2-7
  - Reserved Words 2-8
- The Current Database 2-8
- System Files That Make Up a Database 2-8
- Selecting a Different Current Database 2-9
- Leaving a Current Menu or Screen 2-10

Using the Main Menu to Create a Table 2-10
- Naming the New Table 2-12
- Using the CREATE TABLE Menu 2-12
- An Overview of Building a Table 2-13
  - Defining Each Column in the Table 2-13
  - Correcting Mistakes 2-14
- Naming the Columns 2-15
- Assigning Data Types 2-15
  - CHAR Columns 2-16
  - Number Columns 2-16
  - SERIAL Columns 2-18
  - DATE Columns 2-19
  - MONEY Columns 2-19

DATETIME Columns 2-19
INTERVAL Columns 2-20
Assigning a Data Type to the Column 2-20
Creating Indexes 2-22
When to Index Columns 2-22
NULL Values 2-24
Adding the Remaining Columns to the Table 2-25
Exiting the Schema Editor 2-26
System Files 2-28
Changing the Table Specifications 2-28

In This Chapter

This chapter explains how to use INFORMIX-SQL to set up a database and describes how to create both a table and an index. The following topics are discussed:

- How to create a database
- How to identify the current database
- How to create a table
- What are data types
- How to modify the structure of a table
- When to create an index
- How to create an index
- What are NULL values

Before you can store data using INFORMIX-SQL, you must create a database and one or more tables. If you intend to use only databases that someone else has created, you can skim this chapter.

Two Ways to Create Databases and Tables

INFORMIX-SQL provides two methods for creating databases and tables: selecting the Database and Table options on the INFORMIX-SQL Main menu or using the Structured Query Language (SQL).

Using the Main Menu to Create Databases and Tables

As described in this chapter, the easiest way to create a database or table is to select the Database and Table options on the INFORMIX-SQL Main menu. Menus are provided at each step, and you are prompted to enter each piece of necessary information. HELP menus also are available for added assistance.

Using SQL to Create Databases and Tables

SQL provides a flexible and efficient method for performing a variety of functions. This section introduces SQL. The *Informix Guide to SQL: Tutorial* describes in further detail how to use SQL statements to create databases and tables.

SQL is a set of English-like statements you can use to perform the following functions:

- Create and drop tables and indexes
- Enter and delete data
- Query a database
- Select a different current database
- Display information about one or more tables
- Rename tables and columns
- Check and repair tables by passing these functions to the database server
- Grant and revoke database and table privileges

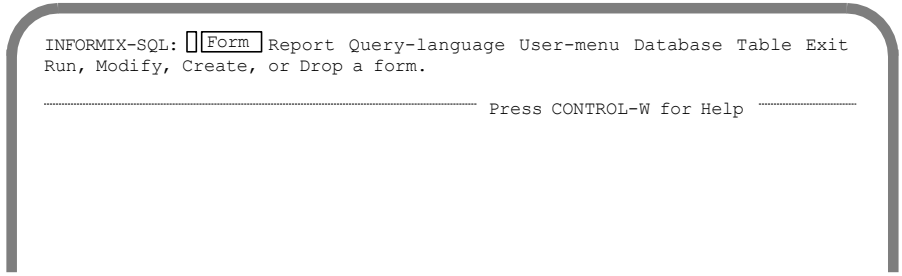
For more information about SQL statement syntax and usage, see the *Informix Guide to SQL: Syntax*.

Accessing INFORMIX-SQL

Before working with INFORMIX-SQL, be sure that your computer is running properly and that the operating-system prompt appears on the screen. Then enter:

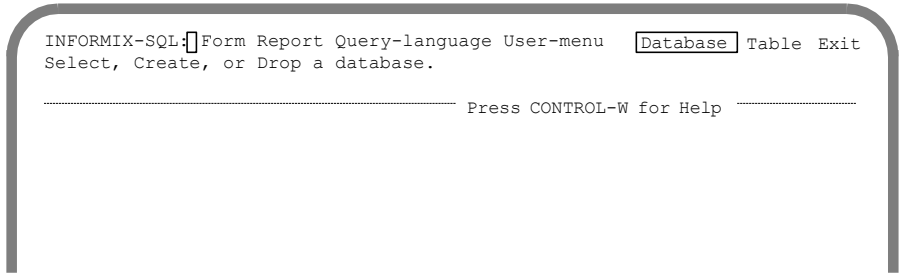
```
isql
```

The Informix logo appears, followed by the INFORMIX-SQL Main menu.

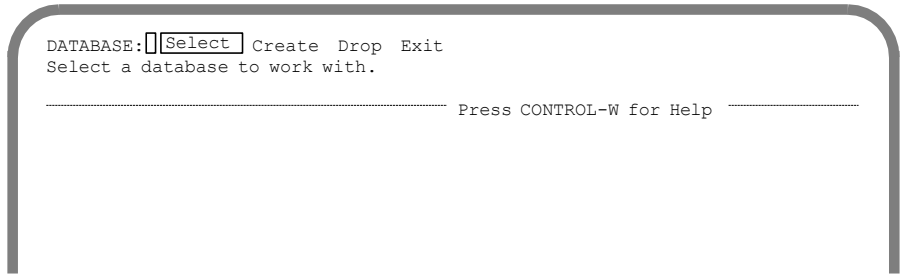


Using the Main Menu to Create a Database

To create a database, select the Database option on the Main menu.



INFORMIX-SQL then displays the DATABASE menu.



The following four options appear on the DATABASE menu:

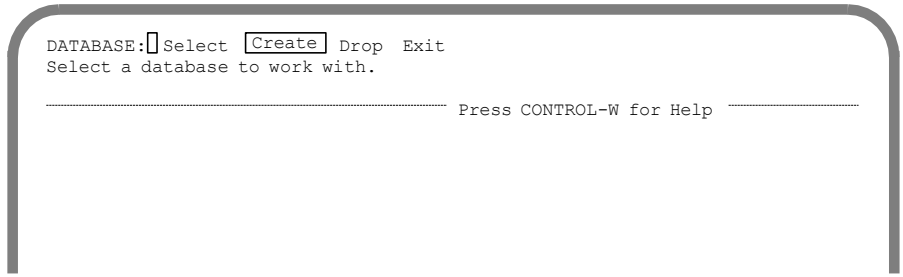
Select Selects an existing database

Create Creates a new database

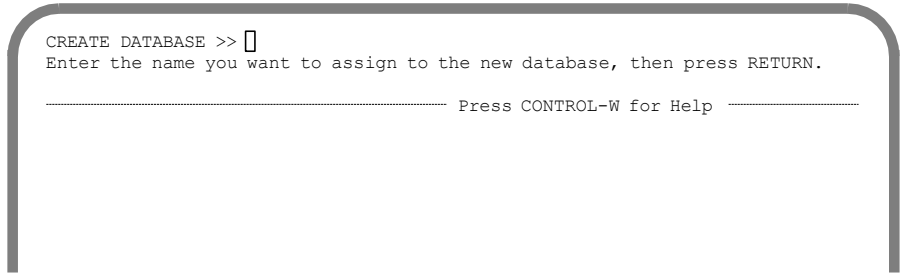
Drop Drops an existing database

Exit Exits to the INFORMIX-SQL Main menu

To create a database, select the Create option.



The CREATE DATABASE screen appears and prompts you to name the new database.



Type the new database name (this chapter uses **mydata**) and press RETURN. You can assign any name you want, as long as you follow the guidelines in the next section. If you accidentally enter the name of an existing database, INFORMIX-SQL displays the message:

```
330: Cannot create database.
```

Use another name for the new database.

Guidelines for Selecting Database Names

When choosing a database name, you are limited in the number and types of characters you can use. Also be aware that certain words might cause ambiguities. If you store more than one database in a directory, each database name must be unique.

Number of Characters

The name you assign to the database can be up to 10 characters long.

Characters You Can Use

The database name must begin with a letter. The rest of the name can consist of any combination of letters, numbers, and underscores (_). You can use uppercase and lowercase letters, but INFORMIX-SQL makes no distinction between them. Therefore, **tahiti**, **Tahiti**, and **TAHITI** all refer to the same database.

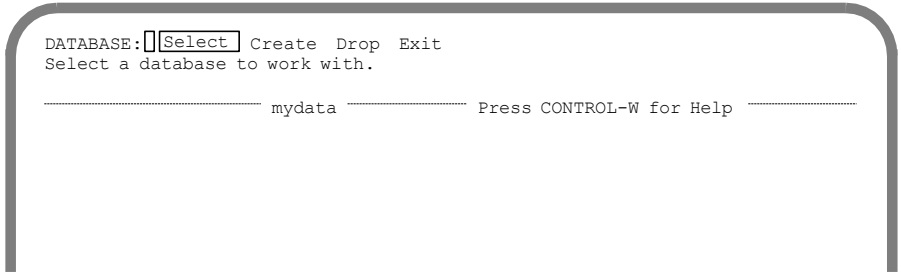
Reserved Words

Certain *reserved words* might cause ambiguities when used as names of tables, columns, databases, or other identifiers. For a list of reserved words, see the *INFORMIX-SQL Reference Manual*.

The Current Database

The database you are currently working with is called the *current database*. INFORMIX-SQL displays the name of the current database in the middle of the broken line that separates the top and bottom of the screen. When you create a new database, it automatically becomes the current database.

After you have named the new database, the DATABASE menu displays its name as the current database.



System Files That Make Up a Database

When you create a database, INFORMIX-SQL creates a directory to store the information about the database and places the new directory in the current directory. INFORMIX-SQL names the directory with the database name you selected and appends the extension **.dbs**. The following example shows a database directory:

```
/usr/andrea/schedule.dbs
```

The database directory initially contains 22 files. The files are associated with the *system catalog* tables. INFORMIX-SQL uses the system catalog tables to keep track of the tables, columns, indexes, views, synonyms, and permissions in each database. The system catalog tables are described in detail in the *Informix Guide to SQL: Reference*.

Selecting a Different Current Database

The Database option on the Main menu lets you select a different current database. When you use the DATABASE menu, your screen looks something like this.

```
DATABASE: [Select] Create  Drop  Exit
Select a database to work with.

                        mydata                  Press CONTROL-W for Help
```

To select a different database, move the highlight to the Select option and press the RETURN key, or type `s`. The SELECT DATABASE screen appears.

```
SELECT DATABASE >>
Select a database with the Arrow Keys, or enter a name, then press RETURN.

                        mydata                  Press CONTROL-W for Help

 mydata

  stores7
```

You select a database by typing its name or by using the Arrow keys to move the highlight to the name of the database you want. Press RETURN, and the name you typed or highlighted becomes your current database. For example, to select the **stores7** database in the previous menu, type or highlight `stores7`, and then press RETURN.

To leave the SELECT DATABASE screen without selecting a new database, press the Interrupt key (CONTROL-C).

Leaving a Current Menu or Screen

You always have the option of leaving the current menu or screen and moving up a level in the menu hierarchy. On most menus, this is accomplished by selecting the Exit menu option.

As you saw with the SELECT DATABASE screen, sometimes you do not have an Exit option. Many screens do not include this option, or you might be working with a feature of INFORMIX-SQL that does not use menus. You can always exit a screen, without making any choices or changes, by pressing the Interrupt key (CONTROL-C). Pressing this Interrupt key moves you out of the current screen and up a level in the menu hierarchy.

Using the Main Menu to Create a Table

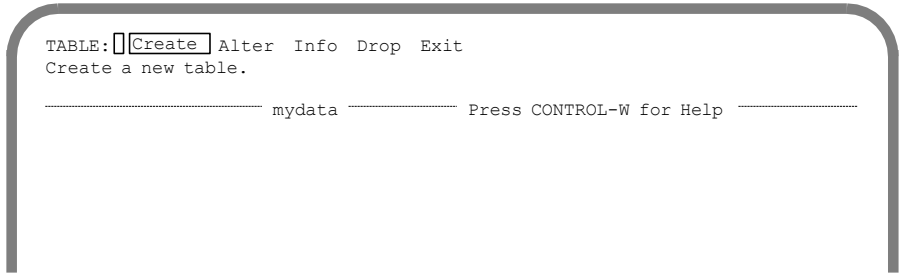
After you create a database, the next step is to create the tables you want included in the database.

INFORMIX-SQL does not limit the number of tables in a database; the limit is determined by the amount of disk space available on your computer.

Any tables created by INFORMIX-SQL are placed in the current database. Before you create a table, be sure that the current database is the one in which you want the table to exist. The name of the current database is displayed on the third line from the top of the screen.

The following examples indicate you are working with the **mydata** database. If **mydata** is not your current database, use the options available with the DATABASE menu and select (or create) **mydata**.

To begin creating a table, select the Table option on the INFORMIX-SQL Main menu. The TABLE menu appears.



The TABLE menu displays the following five options:

Create Creates a new table

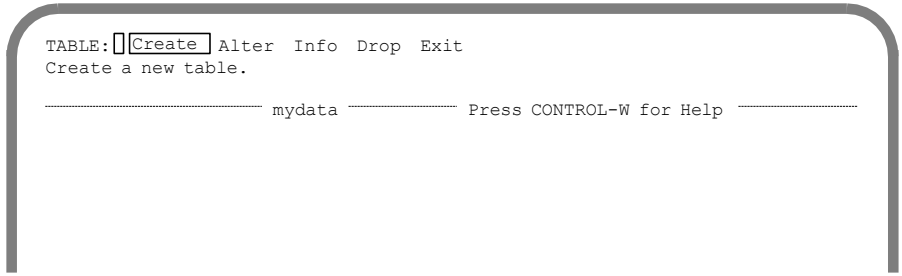
Alter Alters the structure of an existing table

Info Gives information about the structure of a table

Drop Drops a table from the database

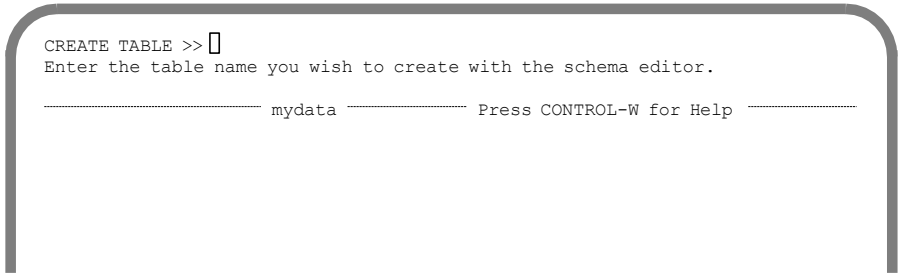
Exit Exits to the INFORMIX-SQL Main menu

To create a table, select the Create option on the TABLE menu.



Naming the New Table

The CREATE TABLE screen appears and prompts you to name the new table.

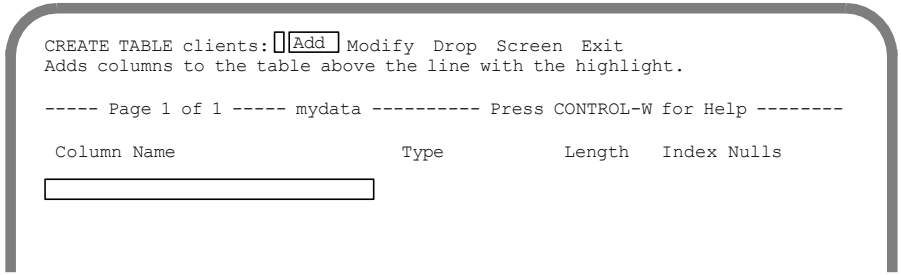


You must follow the same guidelines that apply to database names (see "Guidelines for Selecting Database Names" on page 2-7) except that table names can be up to 18 characters long.

Enter a table name (this chapter uses the name **clients**) and press RETURN. INFORMIX-SQL displays the CREATE TABLE menu.

Using the CREATE TABLE Menu

The CREATE TABLE menu guides you through the steps necessary to create a table.



The CREATE TABLE menu displays the following five options:

Add	Adds a new column to the table
Modify	Modifies the structure of an existing column
Drop	Drops an existing column from the table
Screen	Scrolls down the screen and displays new text
Exit	Exits to the TABLE menu

An Overview of Building a Table

The CREATE TABLE menu displays the *schema* for the table on the bottom portion of the screen. A schema is essentially a blueprint for a table. Just as a blueprint of a building provides instructions for how the building is constructed, a table schema defines the columns in a database table.

Each horizontal line in the schema represents one column, with the name of the column at the left. Each of the five headings in the schema presents some information about the column, such as the length of the column and the type of data it will store.

The CREATE TABLE menu works with the *schema editor* to design the table schema. Whenever you select the Create or Alter option on the TABLE menu to create or alter a table schema, you use the schema editor.

Defining Each Column in the Table

In creating a table schema, you define each column in the table, one at a time. As you define each column, INFORMIX-SQL prompts you for the information it needs. Most of the questions INFORMIX-SQL asks are in the form of menus, so you can select the appropriate response quickly and easily. Some of the questions, such as the name of the column, require you to enter something other than a menu selection. As you answer these questions, INFORMIX-SQL places your answers in the schema. You move from left to right across the screen as you define each column, and from top to bottom as you define additional columns.

Correcting Mistakes

If you make a mistake when entering information about a column, you can back up and correct it as long as you are still defining the same column (still working on the same line). Use the [←] and [→] keys to move the highlight back and forth under the headings. If you notice a mistake in one table column after you have moved on to another, refer to "Changing the Table Specifications" on page 2-27.

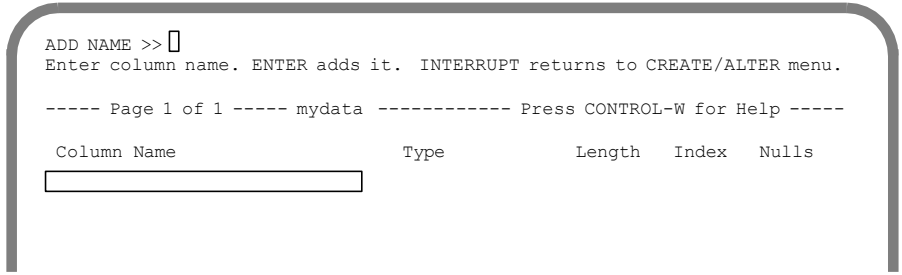
After you finish building a schema for the **clients** table, the screen appear as follows.

```
CREATE TABLE clients : [Add] Modify  Drop  Screen  Exit
Adds columns to the table above the line with the highlight.

----- Page 1 of 1 ----- mydata ---------- Press CONTROL-W for Help --------

 Column Name                    Type            Length   Index   Nulls

 customer_num                   Serial             101   Unique  No
 fname                          Char                15           Yes
 lname                          Char                15           Yes
 company                        Char                20           Yes
 address1                       Char                20           Yes
 address2                       Char                20           Yes
 city                           Char                15           Yes
 state                          Char                 2           Yes
 zipcode                        Char                 5   Dups    Yes
 phone                          Char                18           Yes
```

The **clients** table created in this chapter is identical to the **customer** table included in the **stores7** demonstration database.

Naming the Columns

Select the Add option on the CREATE TABLE menu, and INFORMIX-SQL displays the ADD NAME screen and prompts you for the column name.

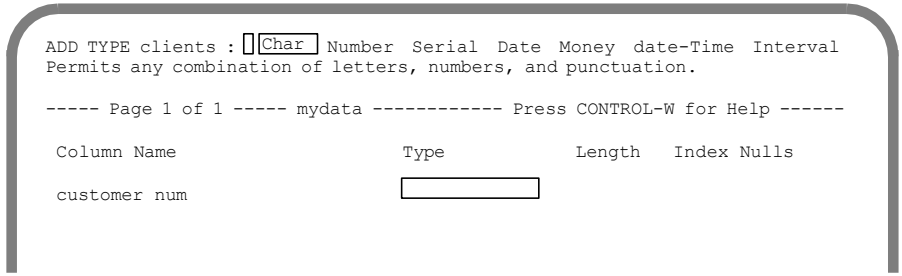


In selecting column names, you must follow the same guidelines that apply to table names (see "Naming the New Table" on page 2-11). You must use a different name for each column within a single table so that INFORMIX-SQL can identify each column uniquely.

Enter `customer_num` for the column name and press RETURN.

Assigning Data Types

The ADD TYPE menu appears next.



You can store many different types of data in a table: dates, zip codes, names, part numbers, part descriptions, salaries, costs—the list is endless. You need to choose the *data type* that best suits each type of data. The data type indicates the kind of information you intend to store in each column.

Data types fall into the following categories:

Character CHARACTER columns store any combination of letters, numbers, and symbols.

Number Number columns store six different data types (including the SERIAL type described next). Each is designed to store a different kind of number data.

Serial SERIAL columns store sequential integers assigned by INFORMIX-SQL.

Date The DATE data type stores calendar dates.

Money MONEY columns store currency amounts.

Datetime DATETIME columns hold values that represent a moment in time.

Interval INTERVAL columns hold values that represent a span of time.

IDS

You can use BYTE and TEXT data types only with Informix Dynamic Server. ♦

CHAR Columns

CHAR (short for CHARacter) columns store any combination of letters, numbers, and symbols. You normally use CHAR columns to store names, addresses, social security numbers, job titles, project names, and other information that combines letters, numbers, and symbols.

You must specify the length of each CHAR column. This length is the total number of characters you want to reserve for the name, address, job title, or whatever data you want to store in the column.

CHARACTER is a synonym for CHAR. You can use CHARACTER in place of CHAR anytime you need to specify the CHAR data type.

Number Columns

There are six data types for numbers. Each type is designed for a different kind of number. You cannot store characters or symbols in any number column, but you can use plus signs (+) and minus signs (-) to show whether numbers are positive or negative.

To indicate a number data type, choose Number when prompted by the ADD TYPE menu. When you specify a number data type, INFORMIX-SQL displays the ADD NUMBER menu and asks you for the exact data type.

A summary of the number data types follows. The SERIAL data type is described separately in "SERIAL Columns" on page 17.

DECIMAL Columns

DECIMAL data types are numbers with definable precision and scale.

To use a DECIMAL data type, choose Decimal when the ADD NUMBER menu appears. You can then stipulate the precision and scale. The total number of digits you want the decimal to hold is the *precision*, and the number of digits after the decimal point is the *scale*. For example, the phrase `(6,2)` indicates that the column stores six-digit numbers, with four digits before the decimal point and two after the decimal point. Specifying the precision and scale is optional.

DECIMAL has two synonyms, DEC and NUMERIC. While neither DEC nor NUMERIC appear on the ADD NUMBER menu, you can use them in place of DECIMAL in the CREATE TABLE statement.

SMALLINT and INTEGER Columns

SMALLINT and INTEGER columns store whole numbers—numbers that have no fractional portion. SMALLINT columns store whole numbers from -32,767 to +32,767. INTEGER columns store whole numbers from -2,147,483,647 to +2,147,483,647.

To use these data types, choose Smallint or Integer when the ADD NUMBER menu appears.

INT is a synonym for INTEGER. INT does not appear on the ADD NUMBER menu, but you can use it in place of INTEGER in the CREATE TABLE statement.

SMALLFLOAT and FLOAT Columns

SMALLFLOAT and FLOAT columns store binary floating-point numbers. SMALLFLOAT columns contain single-precision, floating-point numbers with approximately 8 significant digits. FLOAT columns contain double-precision, floating-point numbers with approximately 14 significant digits. FLOAT columns do not store larger numbers; they store numbers with greater precision.

To use these data types, choose Smallfloat or Float when the ADD NUMBER menu appears.

REAL is a synonym for SMALLFLOAT, and DOUBLE PRECISION is a synonym for FLOAT. While neither synonym appears in the ADD NUMBER menu, you can use REAL in place of SMALLFLOAT, or DOUBLE PRECISION in place of FLOAT in the CREATE TABLE statement.

SERIAL Columns

Serial numbers ensure that a sequential number is assigned to each row of the table. You do not need to enter data in a SERIAL column. Each time you add a new row to a table, INFORMIX-SQL automatically assigns the next number, in sequence, to the SERIAL column. Normally, the starting number is one, but you can select any number greater than zero as your starting number. The highest serial number INFORMIX-SQL can assign is 2,147,483,647.

After INFORMIX-SQL assigns a SERIAL number, you cannot change it. (If you were allowed to change the number, INFORMIX-SQL could not guarantee uniqueness.) Each table in a database can contain only one SERIAL column.

To create a SERIAL column, choose Serial from the ADD TYPE menu.

When you specify a SERIAL data type, INFORMIX-SQL displays the ADD STARTING NUMBER screen and asks you for the starting number. You can select any starting number greater than zero. Do not enter a comma in the number you select. For example, you would enter `62000` to start the numbering at 62,000.

DATE Columns

Use the DATE data type for columns that store calendar dates. A DATE column holds a date in the form *mm/dd/yyyy* where *mm* is the month (1-12), *dd* is the day of the month (1-31), and *yyyy* is the year (0001-9999).

DATE value	Meaning
05/02/1985	May 2, 1985
09/08/1883	September 8, 1883
12/25/1900	December 25, 1900

To create a DATE column, choose Date from the ADD TYPE menu.

MONEY Columns

MONEY columns store currency amounts.

To create a MONEY column, choose Money from the ADD TYPE menu. INFORMIX-SQL displays the ADD LENGTH screen and prompts you for the length (precision) of the MONEY column. Type the length (total number of digits that the MONEY column will accommodate), then press the RETURN key. You then see the ADD SCALE screen. Type the number of digits that should appear to the right of the decimal point and press the RETURN key.

DATETIME Columns

A DATETIME column holds a value that represents a moment in time. It can indicate any time value from a year through a fraction of a second. You establish the precision of the DATETIME column when you create the column.

To create a DATETIME column, choose date-Time from the ADD TYPE menu. INFORMIX-SQL displays the ADD DATETIME QUALIFIER first-TO-last screen and prompts you for the precision of the DATETIME column. Select *first*, then select *last*.

INTERVAL Columns

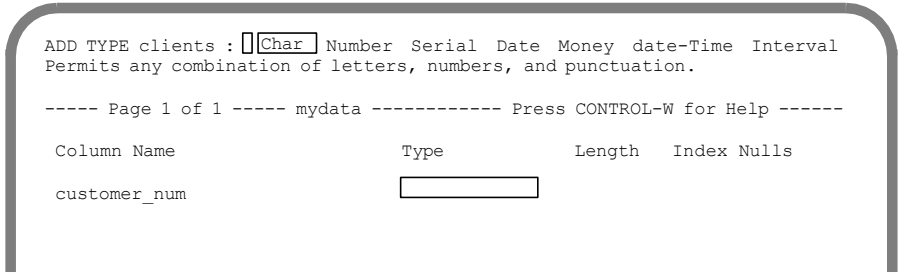
Use the INTERVAL data type for columns in which you plan to store values that represent a span or period of time. An INTERVAL value can represent a span of years and months, or it can represent a span of days through a fraction of a second. You establish the precision of the INTERVAL column when you create the column.

To create an INTERVAL column, choose Interval from the ADD TYPE menu. INFORMIX-SQL displays the ADD INTERVAL QUALIFIER first(n)-TO-last screen and prompts you for the precision of the INTERVAL column. When you select *first*, INFORMIX-SQL prompts you again for the precision of *first*.

Assigning a Data Type to the Column

You are now ready to assign a type to the **customer_num** column.

The ADD TYPE menu should still appear on the screen.



The **customer_num** column contains the customer number for each store. Data type SERIAL is appropriate for this column because you want to ensure that a unique number is assigned to each customer.

Type `s` for serial, and INFORMIX-SQL displays the ADD STARTING NUMBER screen.

```
ADD STARTING NUMBER >>
Enter the starting number.  RETURNadds it.

----- Page 1 of 1 ----- mydata ----------- Press CONTROL-W for Help -------

 Column Name                    Type            Length   Index   Nulls

 customer_num                   Serial               1  Unique  No
```

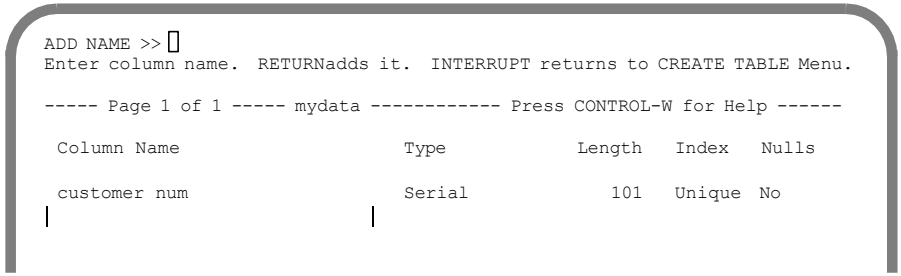
INFORMIX-SQL displays the SERIAL data type under the Type heading of the schema and, because you are describing a SERIAL column, asks you for the number INFORMIX-SQL should begin with when assigning numbers. INFORMIX-SQL automatically starts assigning SERIAL numbers at the number 1. If you want a different starting number, indicate it at this time; otherwise, press RETURN to select the default.

Enter `101` to begin the numbering sequence at 101.

INFORMIX-SQL automatically inserts `Unique` under the Index heading and `No` under the Nulls heading. INFORMIX-SQL assumes you want to index a SERIAL column and, because it is type SERIAL, duplicate values should not exist and NULL values should not be allowed. (Indexes and NULL values are discussed in the next two sections.)

Assign the SERIAL data type to complete the definition of the **customer_num** column.

The highlight moves down a line and to the left, the ADD NAME menu appears, and you can begin defining the next column.



Creating Indexes

INFORMIX-SQL finds information more quickly if you create indexes for the columns in a table. However, just as you can find information in a book without consulting the index, INFORMIX-SQL can find information in a database even if you do not index any columns.

In fact, if only one table is in your database, you probably will not need any indexes until you have entered several hundred rows of data into the table. Placing too many indexes on a table has the undesired effect of slowing down the system. As your database increases in size and your database needs change, you can always add and delete indexes.

A detailed discussion of indexes and indexing strategies is in the *Informix Guide to SQL: Syntax*.

When to Index Columns

It usually makes sense to index only columns that are included in queries or sorts. For example, if you generally sort personnel information alphabetically by last name, it makes sense to index the column that contains last names.

Indexing Guidelines

Indexing choices depend on how you use your database. In general, you do not need to create an index on a column until one or more of the following conditions exists:

- A table contains at least 200 rows.
- A column is used in a join operation.
- A column is regularly used in searches or sorts.

What You Can Index

You can create an index that applies to a single column or to several columns as a group.

Columns You Should Not Index

SE

IDS

For INFORMIX-SE, you cannot create an index for a column or group of columns whose total length exceeds 120 bytes. You should not create an index on a column that contains a large number of duplicate values. For information about column lengths for data types specific to Informix Dynamic Server, see the *Administrator's Guide for Informix Dynamic Server*. ♦

Preventing Duplicate Entries

INFORMIX-SQL normally allows you to enter the same value in different rows of an indexed column. In other words, even if you index a **last_name** column, you can still enter information about many people whose last name is Smith.

Sometimes you might want to prevent users from entering duplicate information in an indexed column. If you create a column to store social security numbers, you probably want to prevent duplicate entries. To do that, choose `no` on the ADD DUPLICATES menu.

You also can prevent duplicate entries by placing a unique constraint on a column when you issue a CREATE TABLE statement. For details, see the description of the CREATE TABLE statement in the *Informix Guide to SQL: Syntax*.

Ascending and Descending Indexes

Unless you instruct otherwise, INFORMIX-SQL creates indexes in ascending order. You also can create descending indexes. Create descending indexes only for fields you intend to regularly sort in descending order.

Clustered Indexes

Because the operating system extracts information from the disk in blocks, the more rows that are physically on the same block and are already in the same order as an index, the faster an indexed retrieval proceeds. You can cause the physical order in the table to be the same as the order in an index through *clustering.* For more information, see the CREATE INDEX statement in the *Informix Guide to SQL: Syntax*.

NULL Values

INFORMIX-SQL lets you identify columns in which NULL values are allowed. The purpose of NULL values in a database is to indicate when no value is assigned to a particular column in a particular row of a table. The reasons for not having assigned a value could include not knowing the correct value or that no value yet exists. The NULL also might indicate that no value is appropriate for a given column because of the values that were entered in other columns.

For example, consider entering data for a bank customer who is requesting a loan. If the customer is not employed, the **employer** column in the **customer** table will not contain an entry for this person. This CHAR column has the value NULL.A **hire_date** column in the same table is meaningless in this case. There is no appropriate date to enter, thus the value is NULL.

Adding the Remaining Columns to the Table

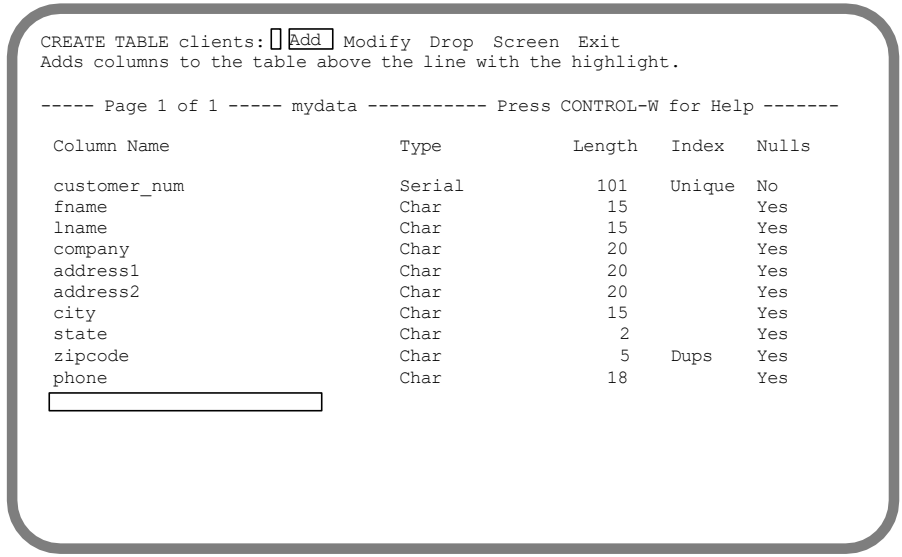
To add the remaining columns to the **clients** table, repeat the steps used to enter the **customer_num** column into the table.

The following table lists the kinds of information that INFORMIX-SQL prompts you to enter for each of the database columns and an appropriate response for each.

Column Name	Type	Length	Index	Nulls
fname	char	15	no	yes
lname	char	15	no	yes
company	char	20	no	yes
address1	char	20	no	yes
address2	char	20	no	yes
city	char	15	no	yes
state	char	2	no	yes
zipcode	char	5	duplicates	yes
phone	char	18	no	yes

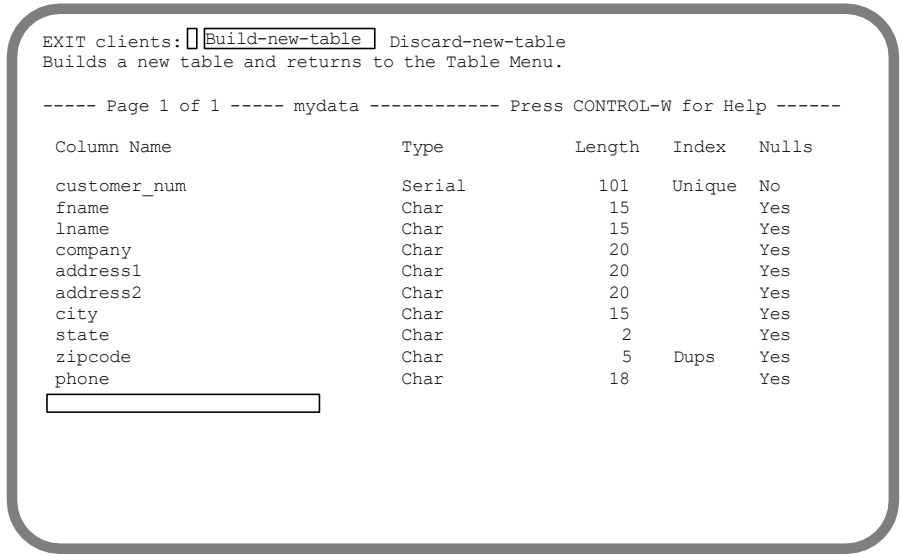
Exiting the Schema Editor

After you enter the final piece of information that defines the last column in the **clients** table (the **phone** column), INFORMIX-SQL moves the highlight under the Column Name heading and displays the ADD NAME screen. Since this is the last column, press RETURN, and INFORMIX-SQL exits the ADD NAME screen and displays the CREATE TABLE menu.



```
CREATE TABLE clients:  Add  Modify  Drop  Screen  Exit
Adds columns to the table above the line with the highlight.

----- Page 1 of 1 ----- mydata ----------- Press CONTROL-W for Help -------

 Column Name                    Type            Length   Index   Nulls

 customer_num                   Serial             101   Unique  No
 fname                          Char                15           Yes
 lname                          Char                15           Yes
 company                        Char                20           Yes
 address1                       Char                20           Yes
 address2                       Char                20           Yes
 city                           Char                15           Yes
 state                          Char                 2           Yes
 zipcode                        Char                 5   Dups    Yes
 phone                          Char                18           Yes
```

The schema for the **clients** table is now complete. Select the Exit option and INFORMIX-SQL displays the EXIT menu.



The EXIT menu displays the following two options:

Build-new-table Builds the table

Discard-new-table Discards the table instructions

Select the Build-new-table option.

System Files

For each table you create, INFORMIX-SQL creates two system files. INFORMIX-SQL stores these files in the same directory as the system catalog tables and names them with the first several characters of the name of the table, a unique number starting at 100, and the following extensions:

.dat This file contains table data.

.idx This file contains table indexes.

When you use the operating-system command to list the contents of the **mydata.dbs** directory, you see the names of these files and the names of the system catalog tables. Do not do anything with these files; INFORMIX-SQL uses them to keep track of the information in the table. If you delete them, you also delete your table.

Changing the Table Specifications

After you set up the schema for a table (even after you have put data into the table), INFORMIX-SQL allows you to change the way you store your data. Occasionally, making a change of this kind also might result in lost data. INFORMIX-SQL always advises you of this situation and gives you the option of not making the change. An example of a change that could cause a loss of data is changing the length of a CHAR column from 20 to 10. INFORMIX-SQL would drop (truncate) the last 10 characters of this column in each row in the table.

The TABLE menu includes an Alter option that you can use to change a schema. Select the Alter option, and INFORMIX-SQL prompts you for the name of the table you want to modify. After you enter the table name, INFORMIX-SQL displays the ALTER TABLE menu and accesses the schema editor.

The ALTER TABLE menu displays the following five options:

Add	Adds a column to the table
Modify	Modifies a column definition by changing one or more of the five schema headings (Column Name, Type, Length, Index, and Nulls)
Drop	Removes an entire column by erasing one line of the schema
Screen	Scrolls down the screen
Exit	Returns to the TABLE menu

The Alter option on the TABLE menu is described in detail in "Changing the Structure of a Table" on page 9-6.

Chapter

3

Entering Data

In This Chapter 3-3

What Is a Screen Form? 3-3
- Screen Forms in the Demonstration Database 3-4

What Is PERFORM? 3-5
- Including Data from Multiple Tables 3-5
- Each Table Can Contribute to Several Forms 3-6

Using a Screen Form 3-6

The PERFORM Screen 3-8
- Information Lines 3-9
- Screen Form 3-10
- Status Lines 3-10

The PERFORM Menu Options 3-11

Adding Data to a Database 3-12
- Entering Data in the Fields 3-13
  - Number Fields 3-14
  - Character Fields 3-14
  - Date Fields 3-14
  - Money Fields 3-16
  - Datetime Fields 3-16
  - Interval Fields 3-16
- Confirming Your Entry 3-17
- Data Checking in PERFORM 3-17
- Special Functions 3-18
- Positioning the Cursor 3-18
- Field Editing 3-19
- Using the Multiline Editor 3-20
- Storing the Row 3-21

Changing Existing Data 3-21
Removing a Row 3-22
Exiting PERFORM 3-22

In This Chapter

This chapter explains how to enter and store data in an INFORMIX-SQL database. The topics discussed include:

- What is a screen form?
- What are fields?
- What are field delimiters?
- How to add a new row to a table
- How to change an existing row in a table
- How to delete a row from a table

This chapter covers important features that you should be familiar with before you go to Chapter 4, "Querying a Database."

For more information about how to enter data on a screen form, refer to the *INFORMIX-SQL Reference Manual*.

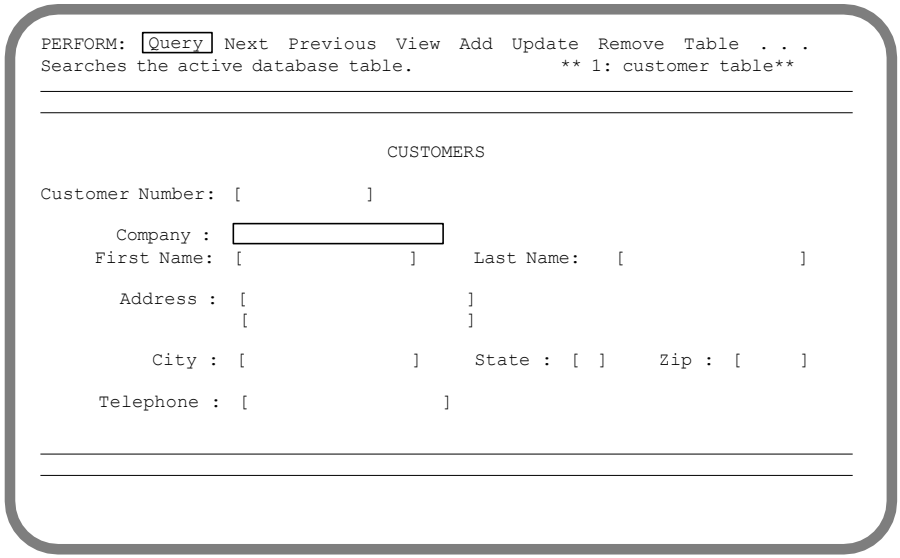
What Is a Screen Form?

After you create a table, you can store data in it. The easiest way to enter the data you want to store is to use a *screen form*. Much like a printed form, a screen form contains blank spaces that you can fill in. The blank spaces on a screen form are called *fields*.

You can create your own screen forms or use forms that someone else creates. Each screen form is assigned a name when the form is created. This chapter explains how to use an existing form. Chapter 6, "Creating Your Own Forms," explains how to create screen forms.

Screen Forms in the Demonstration Database

The demonstration database includes a screen form named **customer** that you can use to enter data into the **customer** table. You will soon learn how to select this form. Here is what the **customer** form looks like when it appears on the screen.



You add data to a table by entering it in the fields on a form. Each field normally corresponds to a column in one of the database tables. The data you enter in a field is stored in the corresponding column. The correspondences between fields and columns are specified when the form is created.

For example, the Company field on the **customer** form corresponds to the **company** column in the **customer** table. Whatever you enter in the Company field is stored in the **company** column in the **customer** table.

Fields presented on the screen are surrounded by brackets [ ] called *delimiters*. You can specify different delimiters when you create a form if you do not want to use the brackets. You normally include a descriptive label alongside each field to indicate what data is in the field. For example, the **customer** form contains labels like Customer Number and Name.

You can include horizontal and vertical lines in your form. You also can use special graphics characters to draw boxes around fields on a form. The **customer** form demonstrates the use of horizontal lines.

Fields can be displayed in reverse video and in color. The Company field in the **customer** form appears in reverse video.

A form might be one or several pages long. If a form includes more data than can fit comfortably on a single screen, you can split the form over several screen *pages*. If your screen is longer than the standard 24 lines, you can size your form to the actual screen size.

What Is PERFORM?

When you use a screen form, you are using a feature of INFORMIX-SQL called PERFORM. You can access PERFORM from the INFORMIX-SQL Main menu. Because it is a separate program, you also can access it directly from the operating system. For information on accessing programs from the command line, refer to the *INFORMIX-SQL Reference Manual*.

Including Data from Multiple Tables

A single form can include data from several tables. The open-file limitation on your system limits the number of tables included in a form.

The Generate option lets you include data from up to eight different tables on a single form. To use more than one table in a screen form, choose the Select-more-tables option. INFORMIX-SQL displays the list of available tables and lets you select another table. The tables must all be part of the same database.

The examples include two forms that contain data from several tables. The **orderform** form combines data from the **customer**, **orders**, **items**, and **manufact** tables. The **sample** form combines data from the **customer**, **orders**, **items**, **stock**, and **manufact** tables.

When you use a form that contains data from multiple tables, one table is always the *active table*. The active table is the table with which you are currently working. PERFORM stores the data you enter in the active table. If you delete data, PERFORM deletes it from the active table. You can easily identify the fields on the screen that belong to the active table because they are surrounded by brackets (or the alternative delimiters you selected).

If a form includes data from only one table, that table is always the active table. For more information, see Chapter 5, "Using Multiple-Table Forms."

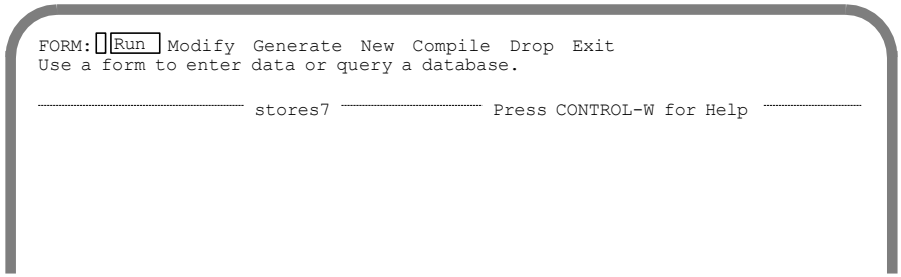
Each Table Can Contribute to Several Forms

There is no practical limit to the number of forms you can create for a single database. The various forms might contain different information or might display the same information in different ways. By creating multiple forms, you can combine information in whatever formats are most useful to you.

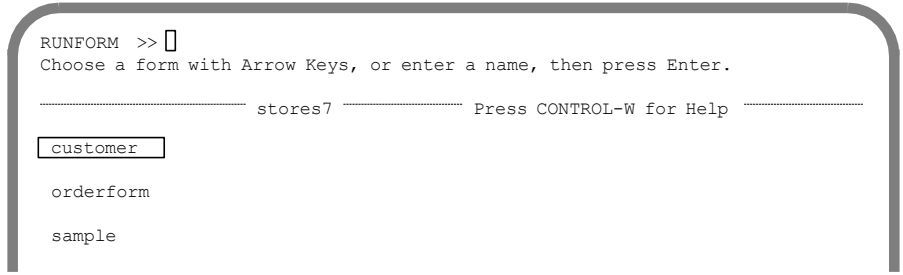
For example, the demonstration database includes one form for customer information (the **customer** form); a second form that combines customer, order, item, and manufacturer information (the **orderform** form); and a third form (the **sample** form) that uses information from all of the tables in the **stores7** demonstration database. The **customer** table contributes information to each form.

Using a Screen Form

To access PERFORM, select the Form option on the INFORMIX-SQL Main menu. The FORM menu appears.



To use a screen form, select the Run option on the FORM menu. PERFORM displays the RUN FORM screen and a list of available forms.

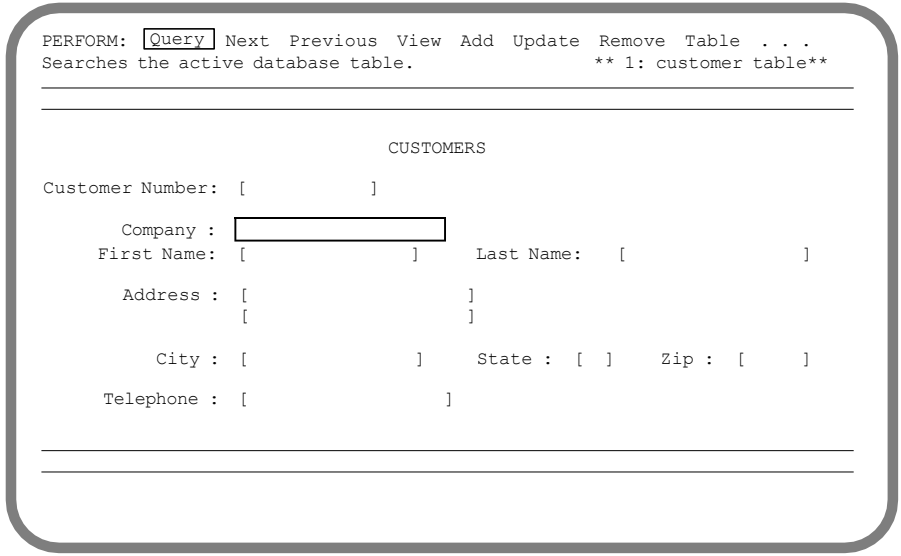


Choose the form you want by highlighting its name with the Arrow keys and then pressing RETURN. If you prefer, you can type the name of the form and then press RETURN.

The examples in this chapter use the **customer** form and the **stores7** demonstration database. To select the **customer** form, position the highlight on `customer` and press RETURN.

The PERFORM Screen

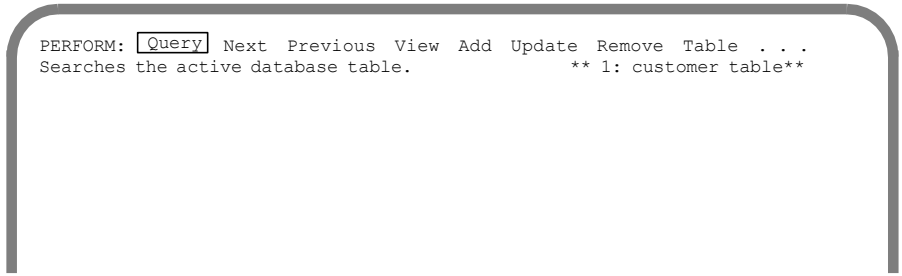
After you select a form, INFORMIX-SQL locates the form and loads it into PERFORM. The PERFORM screen with the selected form then appears. Here is the **customer** form.



The PERFORM screen is divided into three sections: information lines, the screen form, and status lines.

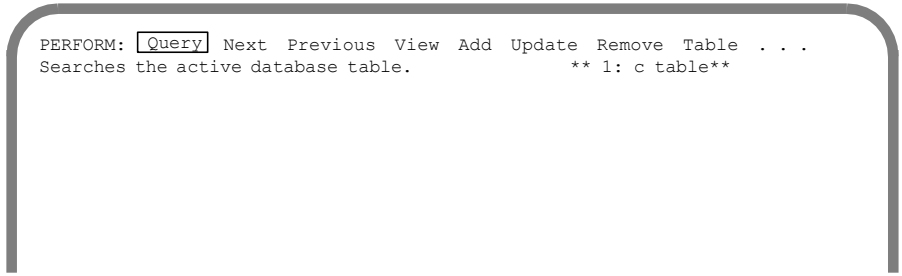
Information Lines

The PERFORM menu appears on the top two lines of the screen (also called *information lines*). The first line displays a list of menu options; the second line describes the current option and indicates the number and name of the active database table. The PERFORM menu initially appears, as follows.



PERFORM assigns a number to each table contributing data to the screen form; if there are three tables, PERFORM numbers them 1, 2, and 3. In the sample figure, the **customer** table is both table 1 and the active table.

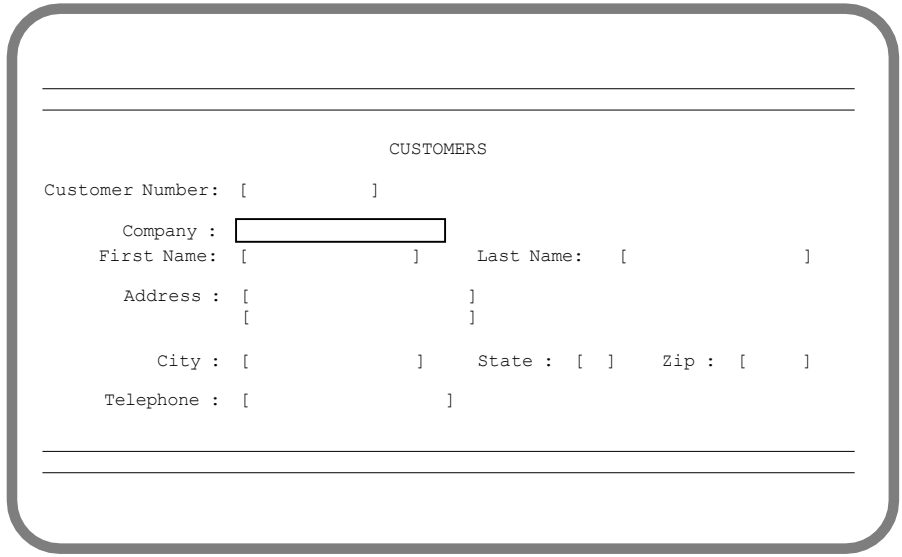
You can create an alias for the table name in your form. When you do so, the second information line shows the alias you select rather than the original table name. Here the **customer** table is aliased as **c**.



For more information on using aliases, see the *INFORMIX-SQL Reference Manual*.

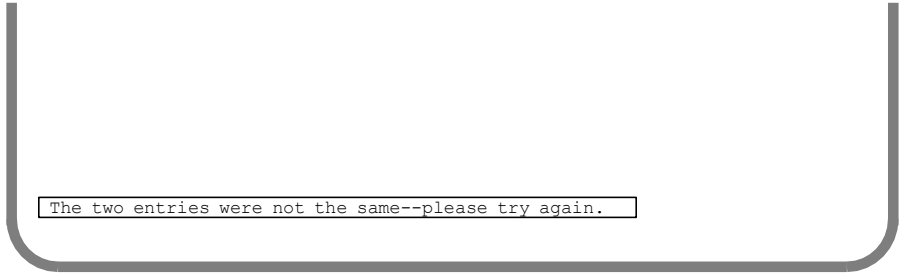
Screen Form

The middle section of the screen displays the form you select, in this case the **customer** form.



Status Lines

PERFORM uses the last two lines of the screen to display error messages, as well as any messages generated by the form itself.



The PERFORM Menu Options

The PERFORM menu is two menu pages wide. The following figure illustrates the first menu page.

```
PERFORM: [Query] Next  Previous  View  Add  Update  Remove  Table  . . .
Searches the active database table.              ** 1: customer table**
```

Tip: *The number of options that appear on the first menu page depends on the character capacity of your screen. The two-page screens displayed here demonstrate a monitor with an 80-character screen. Terminals with a larger character capacity show more options on the first menu page.*

The ellipsis (. . .) at the end of the first line of the screen indicates that additional menu items are available on the second page of the menu. Move the highlight past the Table option to display the second menu page. The following figure illustrates the second menu page.

```
PERFORM: . . .  [Screen] Current  Master  Detail  Output  Exit
Shows the next page of the form.                ** 1: customer table**
```

The ellipsis on the second menu page indicates that additional items are available on the previous menu page. Use the SPACEBAR or the Arrow keys to move the highlight to the menu options. When you move the highlight past the first or last menu option on a page, the alternate menu page appears.

The [←] moves the highlight to the left and, after the first item, moves the highlight to the last item on the alternate page. The [↑] and [↓] move the highlight to the first item on the alternate page. The [→] moves the highlight to the right and, after the last item, moves the highlight to the first item on the alternate page.

You select an option in one of two ways:

- Position the highlight on a menu option and press RETURN.
- Type the first letter of the option name, regardless of whether the option appears on the current menu page.

The Add, Update, Remove, and Exit options are discussed in detail in this chapter. The Query, Next, and Previous options are briefly mentioned in this chapter and are discussed in further detail in Chapter 4, "Querying a Database."

The current, master, and detail options are explained in Chapter 5, "Using Multiple-Table Forms." The Screen and Output options are discussed in Chapter 4. The View option displays the contents of a field of data type TEXT or BYTE and applies to Informix Dynamic Server only. ♦

IDS

Adding Data to a Database

Use the Add option on the PERFORM menu to create a new row in the active table. For example, you would use Add to create an entry in the **customer** table for a newly acquired customer.

To use the Add option

1. Type `a` to select the Add option.

   PERFORM displays the ADD menu and the cursor moves into the first field on the form.
2. Enter the data you want to add into the first field on the form (the Company field).
3. Press TAB or RETURN to shift the cursor to the next field.
4. Enter the data you want to add into the next field on the form (the First Name field).

5. Press TAB or RETURN to shift the cursor to the next field and enter the appropriate data.
6. Repeat this process until you have entered the desired information in all fields in the form.

Tip: *In the demonstration database, the State field accepts only the values* `ca`, `or`, `nv`, *and* `wa`.

7. Press ESCAPE to add the new row data to the table. Press Interrupt to cancel the Add option without storing the row.

The following screen displays the information lines that appear when you select the Add option on the PERFORM menu.

```
ADD: ESCAPE adds new data.  INTERRUPT discards it.  ARROW keys move cursor.
Adds new data to the active database table.     ** 1: customer table**
```

Entering Data in the Fields

SE

What you can enter in a field depends on the data type of the column to which the field corresponds. The six major types of fields available with INFORMIX-SE are:

- Number
- Character
- Date
- Money
- Datetime
- Interval ♦

IDS

Informix Dynamic Server supports TEXT, BYTE, and VARCHAR data types in forms. For more information, refer to the *INFORMIX-SQL Reference Manual*. ♦

Tip: *Complete information about Informix data types appears in the "Informix Guide to SQL: Reference."*

Number Fields

Number fields correspond to SMALLINT, INTEGER, SMALLFLOAT, FLOAT, DECIMAL, or SERIAL columns. You can enter only numbers in number fields. You must enter the correct type of number in each number field—integers in INTEGER fields, floating-point numbers in FLOAT fields, and so on.

PERFORM automatically assigns values to SERIAL fields when you add data to a database. You cannot enter numbers in SERIAL fields and, once INFORMIX-SQL assigns a serial number, you cannot change it.

As you enter data on the **customer** form, the cursor does not move into the Customer Number field because the corresponding column in the **customer** table is a SERIAL field. PERFORM displays a serial number in the Customer Number field when you confirm the data you have entered by pressing ESCAPE.

Character Fields

Character fields in PERFORM correspond to CHAR columns in a database table. You can enter letters, numbers, and symbols in character fields. You can enter as many characters as you can fit in the field.

A character field is normally the same length as the CHAR column to which it corresponds. However, if the field is longer than the column, you might occasionally enter more characters than the column can hold. If you do, PERFORM stores as many characters as it can and displays a message indicating that some of the characters you entered have not been stored.

Date Fields

Enter calendar dates in date fields in the form [*mm*]*m*[*d*]*d*[*yy*]*yy*, with any non-number characters as optional dividers between the month, day, and year. For example, you can type August 4, 1994 in any of these ways:

```
Aug 4 1994
08/04/94
8.4.94
08 04 1994
```

If you enter a two-digit year, PERFORM assumes it is a year in the 1900s unless you set the **DBCENTURY** environment variable. For more information, see "Date Fields and Abbreviated Year Values" on page 3-15.

After you press RETURN, PERFORM displays the date in the format specified in the instructions used to create the form.

Date Fields and Abbreviated Year Values

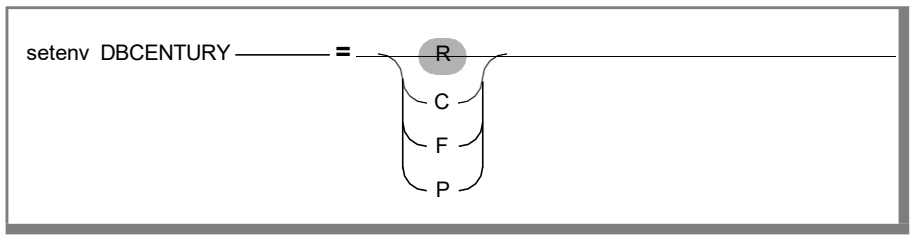
In most releases of INFORMIX-SQL earlier than 7.20, if the user enters only the two trailing digits of a year for literal DATE or DATETIME values, these are automatically prefixed with the digits "19." For example, `12/31/02` is always expanded to `12/31/1902`, regardless of when the program is executed. This legacy behavior is sometimes called *the Y2K problem*.

You can correct this behavior in two ways:

- Set the **DBCENTURY** environment variable.
- Specify the CENTURY field attribute.

The **DBCENTURY** environment variable specifies how to expand abbreviated one- and two-digit *year* specifications within DATE and DATETIME values. Expansion is based on this setting (and on the system clock at runtime). The algorithms **DBCENTURY** supports expand abbreviated years into four-digit year values that end with the same digits (or digit) that the user entered.)

The syntax for setting **DBCENTURY** is as follows:



C	Use the past, future, or current year closest to the current date.
F	Use the nearest year in the future to expand the entered value.
P	Use the nearest year in the past to expand the entered value.
R	Prefix the entered value with the first two digits of the current year.

Here *past*, *current*, and *future* are all relative to the system clock

For more detailed information on **DBCENTURY**, see the *Informix Guide to SQL: Reference*.

The CENTURY field attribute supports the same settings as the **DBCENTURY** environment variable, but with a scope that is restricted to a single field. For more detailed information on the CENTURY field attribute, see the *INFORMIX-SQL Reference Manual*.

Money Fields

Enter dollars-and-cents amounts in money fields. PERFORM displays money amounts with a dollar sign and decimal point. When you enter data in a money field, do not enter a dollar sign or comma. For example, you can type four thousand dollars in a money field in either of the following ways:

```
4000
4000.00
```

PERFORM displays `$4000.00`.

Datetime Fields

Enter date and/or time values in datetime fields in the following form: [*yyy*]*y*-[*m*]*m*-[*d*]*d* [*h*]*h*:[*m*]*m*:[*s*]*s*.[*ffff*]*f*. For example, you would enter 6:35:42 a.m., December 7, 1991, as follows:

```
1991-12-7  06:35:42
```

For additional information about Y2K compliance with DATETIME values, see "Date Fields and Abbreviated Year Values" on page 3-15.

Interval Fields

Enter values that represent a span of time in interval fields. Intervals can be made up of years and months, or of days, hours, minutes, seconds, and fractions of a second. You cannot, however, combine months and days in the same interval value. You would enter an interval of 12 hours and 30 minutes as follows:

```
12:30
```

Confirming Your Entry

When you have entered the data you want for a particular field, press RETURN or TAB. This action tells PERFORM to accept the data you entered and move the cursor to the next field.

Data Checking in PERFORM

PERFORM does not let you enter the wrong type of data into a field. For example, PERFORM does not accept alphabetic characters in a number field. If you enter invalid data, PERFORM displays this message on the status line at the bottom of the screen:

```
Error in field
```

You cannot move the cursor out of a field that contains an invalid value. You must enter an acceptable value or press the Interrupt key to cancel the Add option.

When you create a form, you can specify acceptable values for a particular field. For example, you can specify that a number field accept only numbers from 1 through 50, or that a character field accept only one of five particular words. If you enter data the form cannot accept, PERFORM displays this message on the status line:

```
This value is not among the valid possibilities
```

You also can create a special format—called a *picture*—for a field. For example, social security numbers always have the same format, and you might specify that format in a picture for a social security number field. When a picture format is used, you cannot enter data that does not conform to the picture for a field. (For details, refer to the *INFORMIX-SQL Reference Manual*.)

Special Functions

As you enter data or a query, you can access three special functions by using specific keys.

Function	Key Used
Help	The CONTROL-W key displays a HELP screen that contains a short summary of special keys, control keys, and editing keys, as well as other information about PERFORM.
Execute	The ESCAPE key runs the option you select. For example, to add a new row, type `a` to select the Add option, enter the information for the row, and press ESCAPE to add the row to the database.
Interrupt	On most systems, the CONTROL-C key interrupts or cancels the option you are using. For example, if you select Add when you really want Query, press CONTROL-C, then select the Query option.

Positioning the Cursor

You use specific keys to position the cursor on the screen.

Movement	Key Used
Next Field	The RETURN and [↓] keys move the cursor to the next field.
Backspace	The BACKSPACE and [←] keys move the cursor backward one character at a time without erasing any text. Pressing either key at the beginning of a field moves the cursor to the previous field.
Forward	The [→] key moves the cursor forward one character at a time without erasing any text. Pressing the [→] key at the end of a field moves the cursor to the next field.

(1 of 2)

Movement	Key Used
Fast Forward	The CONTROL-F key moves the cursor down the screen rapidly, stopping in the first field on each line. Use CONTROL-F when you want to move quickly to the bottom of a form that contains many fields.
Fast Backspace	The CONTROL-B key moves the cursor up the screen rapidly, stopping at the last field on each line. Use CONTROL-B when you want to move quickly to the top of a long form.

(2 of 2)

Field Editing

If you make a mistake while entering data in a field, you can correct it by backspacing and retyping. However, it is often faster to use the PERFORM field-editing feature. You can use two editing modes to enter data in a field:

- In *typeover mode*, the characters you type replace existing data. You could use typeover mode to change "Sports' R Us" to "Abe's Sporting Goods."
- In *insert mode*, the characters you type push existing data to the right. You could use insert mode to add an *i* to *Rchard*.

When you access PERFORM, you are in typeover mode; you must press CONTROL-A to change to insert mode. Press CONTROL-A a second time to return to typeover mode.

Edit data that appears in a field by using the following keys.

Function	Key Used
Backspace	The BACKSPACE and [←] keys move the cursor back one character at a time without erasing any text. If you press either key at the beginning of a field, the cursor moves back to the previous field.
Forward	The [→] key moves the cursor forward one character at a time without erasing any text. If you press [→] at the end of a field, the cursor moves forward to the next field.
Delete a Character	CONTROL-X deletes the character beneath the cursor. The cursor remains in place, and text shifts over to fill the space that was occupied by the deleted character.
Change Mode	CONTROL-A shifts between insert and typeover mode. When you access PERFORM, you are in typeover mode.
Delete Forward	CONTROL-D deletes everything from the current cursor position to the end of the field you are editing.
Repeat Data	CONTROL-P enters the most recently displayed value in a field. When you use the Add option to enter several rows in which one or more fields contain the same data, you can avoid retyping the data by pressing CONTROL-P. When you use the Update option, CONTROL-P restores the value that appeared in a field before you modified it.
Clear Screen	CONTROL-C clears any search criteria you have entered with the Query option.

Using the Multiline Editor

You can use a multiline editor to edit long CHAR columns. You enable this editor by specifying the WORDWRAP attribute. However, some keys might function differently or have additional features. For more information about entering data using the multiline editor, refer to the *INFORMIX-SQL Reference Manual*.

Storing the Row

When you have entered information in all the appropriate fields, tell PERFORM to store this information in the active table by pressing ESCAPE. To cancel the Add option without storing the new row data, press the Interrupt key.

When you press ESCAPE, PERFORM adds the new row information to the active table and displays the following message:

```
Row added
```

Changing Existing Data

Use the Update option to change the contents of an existing row. For example, you would use Update to enter a new phone number for a customer in the **customer** table.

You can update only the current row, that is, the row currently appearing on the screen. Use the Query, Next, and Previous options to display the row you want to update. You will learn how to use these options in the next chapter.

When the row you want to update appears on the screen, you update it in the following way:

1. Type `u` to select the Update option.

   PERFORM displays the UPDATE menu, and the cursor moves into the first field on the form.
2. Use the cursor-positioning keys to move the cursor from field to field and the editing keys to change the data in as many fields as you like.
3. Press ESCAPE to store the changes in the active table.

   If you do not want to save the changes, press the Interrupt key to cancel the Update option. The row remains unchanged.

When you press ESCAPE, PERFORM modifies the row to reflect the changes you made and displays the following message:

```
This row has been changed
```

Removing a Row

Use the Remove option to delete a row from the active table. For example, you would use Remove to delete information about a customer who quits the sporting goods business.

Like Update, the Remove option affects only the row that currently appears on the screen. When the row you want to delete appears on the screen, you can delete it in the following way:

1. Type `r` to select the Remove option.

   PERFORM displays the REMOVE menu and, in the upper left corner of the screen, asks:

   ```
   REMOVE: Yes No
   Removes this row from the active table.
   ```

2. To remove the row, enter `y`. PERFORM removes the row, clears the information from the screen, and displays the following message:

   ```
   Row deleted
   ```

3. If you do not want to remove the row, enter `n`.

Exiting PERFORM

When you have finished working with a form, use the Exit option to leave PERFORM and return to the FORM menu. You can then choose to work with another form or return to the INFORMIX-SQL Main menu.

If you accessed PERFORM from the operating-system command line, the Exit option returns you to the operating system.

Chapter

4

Querying a Database

In This Chapter 4-3

What Is a Database Query? 4-3
- Searching for an Exact Match 4-4
- Searching for a Group of Rows 4-7
  - The Current List 4-8
  - The Next and Previous Options 4-9
- Finding All the Rows in a Table 4-9

How Database Queries Work 4-10
- Searching with Relational Operators 4-11
  - Number Fields 4-12
  - DATE and DATETIME Fields 4-12
  - INTERVAL Fields 4-12
  - MONEY Fields 4-12
  - CHAR Fields 4-13
- Using Wildcard Characters 4-13
- Searching for a Range of Values 4-14
- Searching for the Highest and Lowest Values 4-15
- Searching for Alternative Values 4-16
- Query Operators and Short Fields 4-16
- Searching in SMALLFLOAT and FLOAT Fields 4-17
- Sample Database Queries 4-18

Moving from Screen to Screen 4-18

Saving the Results of a Query 4-19

In This Chapter

This chapter explains how to use a screen form to retrieve data stored in an INFORMIX-SQL database. The following topics are discussed:

- Conducting database queries
- Using search criteria in a database query
- Searching for a single row
- Searching for a group of rows
- Retrieving all rows in a table
- Scanning through a group of rows
- Moving from page to page on a multiple-page form
- Saving the results of a query in a file

Read this chapter to learn how to use PERFORM to query a database.

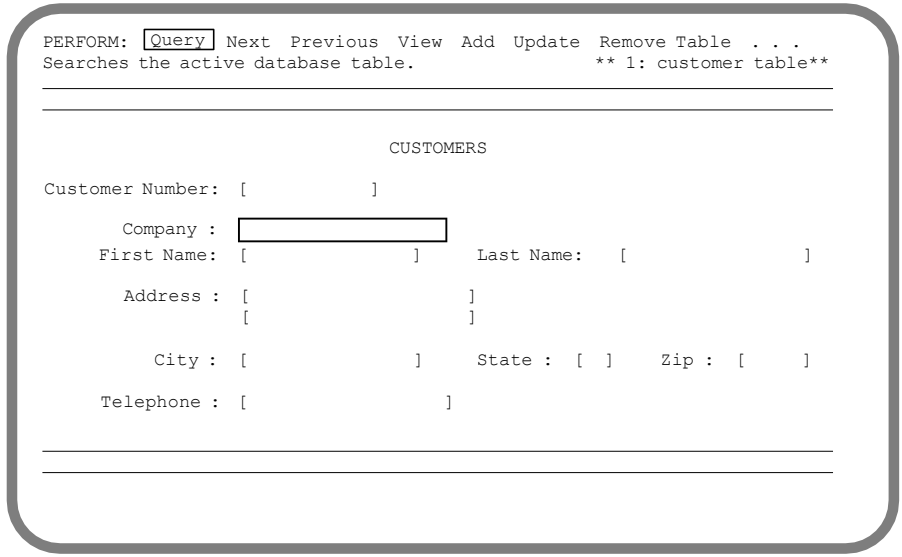
For additional information about querying a database with PERFORM, refer to the *INFORMIX-SQL Reference Manual*.

What Is a Database Query?

After you store data in a database, you can use a screen form to inquire about the data. You can find out how many rows of data exist in a table, and you can select and display the particular row or rows you want to review.

This process is called *querying* a database. PERFORM can retrieve and organize your data in many different ways, depending on the screen form you use and how you structure your query.

The examples in this section are based on the **customer** form included with the demonstration database. The form on the screen is displayed as follows.

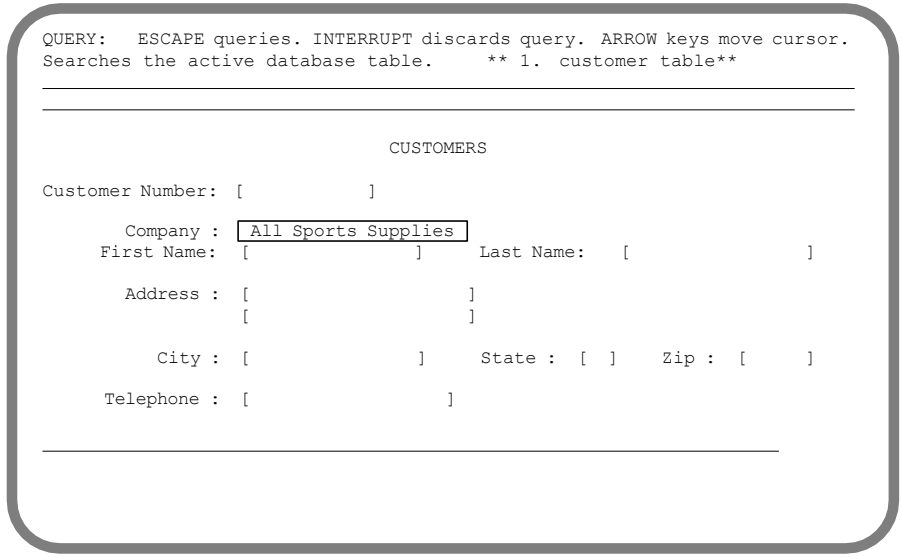


Searching for an Exact Match

The simplest way to query a database is to ask PERFORM to locate a single row. To search for a row, select the Query option and then enter data on the screen form the way it appears in the row you want PERFORM to find.

The data you enter is called *search criteria* because PERFORM uses this information to select rows as it searches the database. Some database management systems refer to this approach as *query by example (QBE)*.

You can use the **customer** form to search for an individual customer's row. First, type `q` to select the Query option, and then type the company's name in the Company field. For example, to search for the row containing information about the "All Sports Supplies" company, you would complete the form as follows.



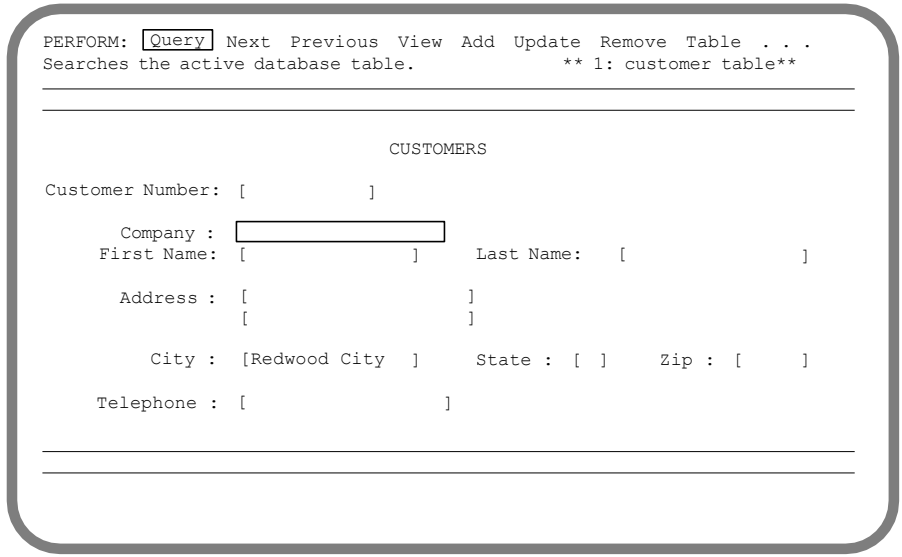
The completed QUERY Screen tells PERFORM to search for rows in which the company column contains "All Sports Supplies." Leaving the other fields blank tells PERFORM not to consider them as part of the query.

To initiate the search, press ESCAPE. When PERFORM locates the row, it appears on the screen with a message on the Status line that indicates the results of the search.

```
PERFORM: Query  Next  Previous  View  Add  Update  Remove Table  . . .
Searches the active database table.              ** 1: customer table**
_____________________________________________________________________
_____________________________________________________________________

                                CUSTOMERS

Customer Number:  [101         ]

       Company :  All Sports Supplies
     First Name:  [Ludwig          ]     Last Name:   [Pauli           ]

        Address :  [213 Erstwild Court  ]
                   [                    ]

           City :  [Sunnyvale       ]     State :  [CA]     Zip :  [94086]

      Telephone :  [408-789-8075       ]

_____________________________________________________________________
_____________________________________________________________________

 1 row(s) found
```

Searching for a Group of Rows

PERFORM can also search for a *group* of rows. For example, to find the rows for all customers with stores in Redwood City, complete the **customer** form as follows.



This screen instructs PERFORM to search for rows in which the **city** column contains "Redwood City" and to disregard the values contained in all other fields. When you press ESCAPE, PERFORM responds as follows.

```
PERFORM: Query  Next  Previous  View  Add  Update  Remove  Table  . . .
Searches the active database table.              ** 1: customer table**
_____________________________________________________________________
_____________________________________________________________________

                                CUSTOMERS

Customer Number: [104         ]

       Company :  Play Ball!
    First Name: [Anthony        ]     Last Name:   [Higgins         ]

       Address : [East Shopping Cntr. ]
                 [422 Bay Road        ]

          City : [Redwood City   ]     State :  [CA]    Zip :  [94026]

     Telephone : [415-368-1100     ]

_____________________________________________________________________
_____________________________________________________________________

 5 row(s) found
```

The message on the Status line indicates that PERFORM found rows for five customers with locations in Redwood City. However, only the first row PERFORM found appears on the screen.

The Current List

In many instances, several rows satisfy the search criteria you enter. When this happens, PERFORM places all the matching rows into a temporary storage area called the *current list*, displaying only the first row it finds.

The current list is a place where PERFORM keeps track of the results of database queries. All rows that satisfy your search criteria are stored in the current list. Each time you select the Query option, PERFORM clears the current list to make room for the results of the new query.

The Next and Previous Options

The Next and Previous options let you scan through the rows in the current list. Type `n` to display the next row in the list, and type `p` to go back to the previous row. When you reach either end of the list, PERFORM displays the following message:

```
There are no more rows in the direction you are going.
```

If you want to skip more than one row at a time, type a number before you enter the option. For example, to skip forward three rows, type `3n`.

Finding All the Rows in a Table

You can locate *all* the rows in the **customer** table (and determine how many there are) by using the Query option (without filling in any fields on the screen form) by typing `q` and then pressing ESCAPE.

The following screen displays the results of the query.

```
PERFORM: [Query] Next  Previous  View  Add  Update  Remove  Table  . . .
Searches the active database table.              ** 1: customer table**
_____________________________________________________________________
_____________________________________________________________________

                                CUSTOMERS

Customer Number:  [101         ]

       Company :  [ All Sports Supplies ]
     First Name:  [Ludwig          ]     Last Name:   [Pauli           ]

        Address :  [213 Erstwild Court  ]
                   [                    ]

           City :  [Sunnyvale      ]     State :  [CA]    Zip :  [94086]

      Telephone :  [408-789-8075     ]

_____________________________________________________________________
_____________________________________________________________________

 18 row(s) found
```

How Database Queries Work

The Query option lets you find information in a database. To use it, you specify search criteria by filling in one or more fields on the screen form with the data that appears in the row or rows you want to find.

PERFORM compares each row in the database with the criteria you enter and selects only those rows that satisfy the criteria. PERFORM places all the rows it finds that satisfy your search criteria into the current list, displaying the first one on the screen.

If you enter search criteria in more than one field, PERFORM selects only rows that satisfy the criteria for all search fields. If you make an error while entering search criteria, you can use CONTROL-C to clear the criteria you entered.

When you want only rows in which a field contains a particular value, enter that value in the field. You also can use these other methods:

- *Relational* and *range* operators to search for values that lie within a particular range
- *Wildcard* characters to search for a single character or for character strings that match a certain pattern
- Highest-to-lowest value operators to search for the row in which a field has the highest or lowest value

These operators are explained in "Searching with Relational Operators" on page 4-11. Figure 4-1 lists the operators, their names, and the data types with which you can use them.

Figure 4-1
Query Operator Syntax

Operator	Name	Data Types	Pattern
=	equal to	all	=x
>	greater than	all	>x
<	less than	all	<x

(1 of 2)

Operator	Name	Data Types	Pattern
>=	greater than or equal to	all	>=x
<=	less than or equal to	all	<=x
<>	not equal to	all	<>x
:	range	all	x:y
..	range	DATETIME and INTERVAL	x..y
*	wildcard	CHAR	*x, x*, *x*
?	single-character wildcard	CHAR	?x, x?, ?x?, x??
|	or	all	x|y

(2 of 2)

Searching with Relational Operators

The first six operators in the table are *relational operators*; you can use them to describe the acceptable values for a field. They are called relational operators because they describe the relationship between the value you enter and the value or values for which you are looking.

Normally, you do not have to enter the equal to (=) operator; if you do not enter an operator, PERFORM assumes you mean "equal to." For example, when you enter `Redwood City` in the City field, PERFORM assumes you mean `=Redwood City`.

However, you need to use an equal sign in two situations involving CHAR fields:

- If you enter an equal sign by itself in a CHAR field, PERFORM searches for rows in which that field contains NULL values.
- If you enter an equal sign followed by an asterisk (*) in a CHAR field, PERFORM searches for rows in which that field contains an asterisk. (You have to use the equal sign with the asterisk because the asterisk is also a query operator.)

Normally, you enter an operator followed by a value, with no spaces between them. The following guidelines explain how to use the relational operators in different types of fields. The examples are based on the **customer** form in the demonstration database.

Number Fields

Enter an operator and a number. For example, to search for all rows in which the customer number is greater than 110, enter `>110` in the Customer Number field.

DATE and DATETIME Fields

Enter an operator and a value. *Less than* means before, and *greater than* means after the date or date and time that is entered. For example, to find orders placed on or before June 5, 1991, you can enter `<=06051991` in the **order_date** field.

INTERVAL Fields

Enter an operator and a value. *Less than* means a shorter span of time, while *greater than* means a longer span of time than you entered. For example, to find orders that are older than 1 year and 6 months, enter `>1-6`.

MONEY Fields

Operators are also useful when querying MONEY fields. For example, to find rows with a value less than $12,000, enter `<12000` in a MONEY field. Do not enter a dollar sign or comma in your query. PERFORM adds the dollar sign automatically. A MONEY field is also present in the form that is used in the next chapter.

CHAR Fields

When you use a relational operator in a CHAR field, the following conditions exist:

- *Greater than* means later in the alphabet, and *less than* means earlier in the alphabet.
- Lowercase letters are greater than uppercase letters.
- Both uppercase and lowercase letters are greater than numbers.

PERFORM compares characters based on ASCII values. (An ASCII collating-order chart appears in the *Informix Guide to SQL: Syntax*.)

For example, to find rows for customers whose last names are alphabetized after Baxter, enter `>Baxter` in the Last Name field. Because lowercase letters have a higher ASCII value than uppercase letters, this search criterion also finds all names that begin with lowercase letters.

GLS

The results of character comparisons are dependent on GLS locale settings and can differ from strict ASCII ordering. For more information, refer to the *Informix Guide to GLS Functionality*. ♦

Using Wildcard Characters

You can use the asterisk (*) and the question mark (?) *wildcard* characters in queries with CHAR data types. You can use the asterisk to match any group of zero or more characters. For example, to find customers whose last names start with a capital B, enter `B*` in the Last Name field.

You can use multiple asterisks to search for fields that contain a certain pattern of characters. For example, you might want to search for customers whose retail locations include the words Avenue or Ave. If you enter `*Ave*` in the Address field, PERFORM finds "41 Jordan Avenue" and "776 Gary Avenue." The search pattern also would find a truncated address like "41 Jordan Ave."

You can use the ? to match a single character. The following screen shows a query using the ? wildcard character. PERFORM retrieves rows in the **customer** table where the value in the **fname** column contains a four-letter "word" beginning with any character and ending with the character string "ick" ("Dick," "Rick," "Nick," and so on).

```
QUERY:   ESCAPE queries. INTERRUPT discards query.  ARROW keys move cursor.
Searches the active database table.             ** 1: customer table**
_____________________________________________________________________________
_____________________________________________________________________________

                                CUSTOMERS

Customer Number:  [            ]

       Company :
     First Name:  [?ick           ]     Last Name:   [                ]

        Address :  [                    ]
                   [                    ]

           City :  [               ]     State :  [ ]     Zip :  [    ]

      Telephone :  [                  ]

_____________________________________________________________________________
_____________________________________________________________________________
Please enter first name if available
```

Searching for a Range of Values

The colon (:) is a *range operator*. It lets you search for all values that lie between one value and another. The range is inclusive.

For example, to search for all zip codes from 94060 through 94080, enter `94060:94080` in the Zip field. When you fill in a field with two values separated by the range operator, PERFORM finds rows in which the value of that field lies within the specified range.

For example, to identify stores with customer numbers between 110 and 115, enter `110:115` in the Customer Number field on the **customer** form. Since the range is inclusive, this example finds customers with the numbers 110 and 115, as well as any numbers in between.

You can use the range operator with CHAR fields. Because PERFORM compares characters based on ASCII values, lowercase letters are processed after uppercase letters. For example, to find customer names that are alphabetized after (and including) Baxter—but omitting names that begin with a lowercase letter— you must enter `Baxter:a` in the Last Name field.

GLS

The evaluation of characters relative to character ranges is dependent on GLS locale settings and can differ from strict ASCII ordering. For more information, refer to the *Informix Guide to GLS Functionality*. ♦

When an odd number of colons (greater than two) appear in a query expression, the middle one is assumed to be a query operator. If the query expression contains an even number of colons and is not a valid single DATETIME or INTERVAL value, INFORMIX-SQL returns an error.

Because DATETIME and INTERVAL constants might include colons, do not use the colon (:) to search for a range of values. Instead, use the `..` search operator with DATETIME and INTERVAL data types to avoid ambiguity. The `..` range operator functions the same way as the colon does for other data types.

Searching for the Highest and Lowest Values

You can use two greater-than (>>) symbols and two less-than (<<) symbols in a query to locate the maximum and minimum column values, respectively.

Enter either operator in a field to search the database for the row that contains the lowest or highest value for that field. The >> and << operators work with all data types.

For example, if you enter `>>` in the Customer Number field on the **customer** form in the demonstration database, PERFORM searches for the row with the highest customer number.

Searching for Alternative Values

You can use the pipe sign (|) in queries to signify "or". The following screen shows a query using the | operator.

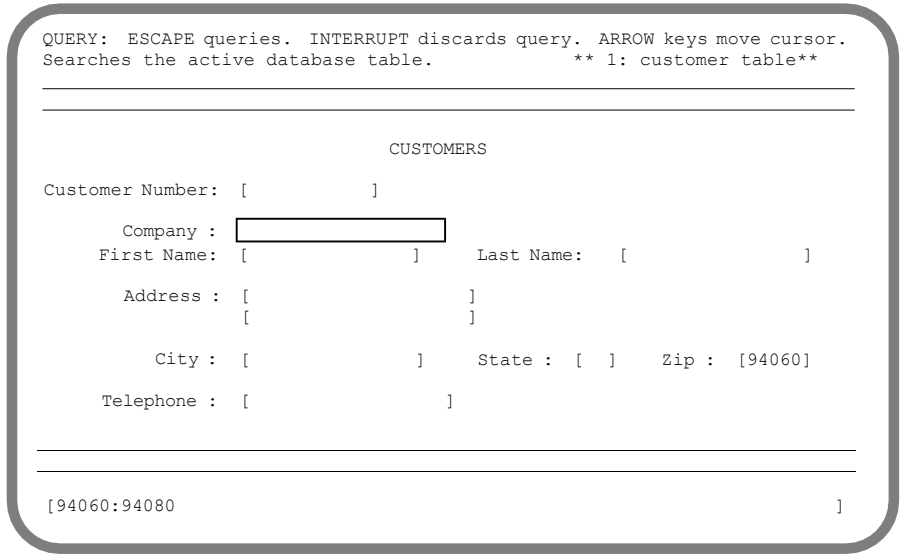
```
QUERY:   ESCAPE queries. INTERRUPT discards query.  ARROW keys move cursor.
Searches the active database table.             ** 1: customer table**
_____________________________________________________________________________
_____________________________________________________________________________

                                CUSTOMERS

Customer Number:  [110|118]

       Company :
    First Name:  [                ]     Last Name:    [                 ]

      Address :  [                     ]
                 [                     ]

         City :  [                ]     State :  [  ]    Zip :  [     ]

    Telephone :  [                    ]

_____________________________________________________________________________
_____________________________________________________________________________
```

In this query, PERFORM searches for rows where the value in the **customer_num** column equals either 110 *or* 118. The row(s) returned from this query are placed in the current list.

Query Operators and Short Fields

Occasionally a display field might be too short to hold the search criterion you enter. When this occurs, PERFORM creates a work space at the bottom of the screen; when you press RETURN, the work space disappears. The field contains the criterion you entered in the work space, even though you can see only the portion that fits in the field.

For example, when you enter `94060:94080` in the Zip field, you see the following screen.



Searching in SMALLFLOAT and FLOAT Fields

The number PERFORM displays in a SMALLFLOAT or FLOAT field might differ slightly from the number you enter because of the way in which INFORMIX-SQL stores FLOAT and SMALLFLOAT numbers. This difference can make searching for an exact match in a floating-point field difficult. For example, if you enter `1.1` in a floating-point field, PERFORM might actually search for `1.10000001`. You can solve this by searching for a range of values, such as `1:1.2`.

Sample Database Queries

In the following queries, you can verify the results displayed on the screen by referring to the **customer** table in the *Informix Guide to SQL: Reference*.

- To find rows for customers located in Menlo Park, type `q` to select the Query option. Enter `Menlo Park` in the City field, and then press ESCAPE to begin the search.
- To search for information on customers located in Menlo Park with a customer number greater than 110, enter `Menlo Park` in the City field and `>110` in the Customer Number field, and then press ESCAPE.
- To find customers located in Redwood City with a customer number between 101 and 110 and a zip code of 94062, enter the following values in the **customer** form:
  - `101:110` in the Customer Number field
  - `Redwood City` in the City field
  - `94062` in the Zip field

Moving from Screen to Screen

Just as printed forms sometimes fill several pages, screen forms might take up more than one screen *page*. You might want to create a multiple-page form to include a substantial amount of data, or simply because it makes better visual or organizational sense to do so.

When a form contains more than one page, you use the Screen option to move back and forth between pages. The Screen option moves you to the next page of a form. When you reach the last page, the Screen option takes you back to page 1.

If you type a number before the Screen option to select the particular page of the form that you want, PERFORM advances directly to that page. For example, to move to the fourth page of a multiple-page form, type `4s`.

The **sample** form included in the demonstration database is a three-page form. You can try the Screen option with this form.

Saving the Results of a Query

Even though the Query option provides rapid access to the information in a database, you might occasionally want to keep a row of query results permanently in a file or use those results in a formatted report. To do this, you use the Output option (on menu page 2 of the PERFORM menu). The Output option writes the results of a query to a file. You can create a new file or you can append the data to an existing file.

You can produce an output file in which rows, data, display-field labels, boxes, lines, and so on, appear as they do on your screen. Alternatively, you can produce an output file in which rows appear just as they do when you run an UNLOAD statement. Rows retrieved using this alternative method appear in an ASCII file, where they can be used either with the READ statement of ACE to produce a report, or as input to an application that accepts ASCII files.

The Output option produces the current list only. Before you can select the Output option, you must first use the Query, Next, or Previous option to locate the row or rows you want to output.

For example, you can use the **customer** form to create a list of customers located in Palo Alto. Select the Query option, enter `Palo Alto` in the City field, and then press ESCAPE. The results of the query look like those on the next screen.

```
PERFORM: Query Next Previous View Add Update Remove Table . . .
Searches the active database table.                ** 1: customer table**
_____________________________________________________________________
_____________________________________________________________________

                                CUSTOMERS

Customer Number: [103         ]

       Company : [ Phil's Sports      ]
    First Name: [Philip          ]     Last Name:   [Currie          ]

      Address : [654 Poplar           ]
                [P. O. Box 3498       ]

         City : [Palo Alto      ]     State : [CA]    Zip : [94303]

    Telephone : [415-328-4543      ]

_____________________________________________________________________
_____________________________________________________________________

2 row(s) found
```

Rows for the two Palo Alto businesses are now in the current list.

Always use the Query option before you select the Output option. Then type o to select the Output option. In the upper left corner of the screen, PERFORM displays the following instruction.

```
Enter output file (default is perform.out):
Enter an output file name.
```

The default file is the file you used most recently with the Output option. If you have not used the Output option since accessing PERFORM, the default is **perform.out**. You can type the name of the file you want PERFORM to use and then press RETURN or simply press RETURN to use the default file name. By default, the file appears in your current working directory. You can place the file in another directory by providing a full pathname.

When you press RETURN, PERFORM displays the name of the file you selected and asks if you want to create a new file or append the output to an existing file. If you choose the default filename **perform.out**, you see the following menu.

```
FORM OUTPUT FILE: Append Create
Adds new data to an existing output file.
```

If you select a file that already contains data, and you do not want to delete that data, type `a` to append the new data to the end of the existing file. To create a new file, type `c`. If you select an existing file and select the Create option, PERFORM deletes the old version of the file when you run the Output option, and you lose any data stored in that file.

When you press RETURN, PERFORM displays this menu.

```
FORM OUTPUT FILE LIST: Current-list One-page
Writes the Current List to the file.
```

You can ask PERFORM to output data for all rows in the current list, or you can request a single row at a time. Type `c` or press RETURN if you want to output every row in the current list. Type `o` to output only the row that currently appears on your screen. To output some but not all of the rows in the current list, you can alternate between the Output and Next/Previous menu options.

When you type c or o, PERFORM prompts you for the format of the output file.

```
OUTPUT FORMAT: Unload-format Screen-format
Writes the selected output in ASCII format
```

Press RETURN or type u to output the retrieved row or rows as an ASCII file in *unload* format. Select this option if you plan to use this file in an ACE report or as input for another application.

Type s if you want to output the retrieved row or rows in a file formatted to look the same as your screen display. Like the screen display, the file includes the data and any additional field labels, boxes, lines, or other screen items.

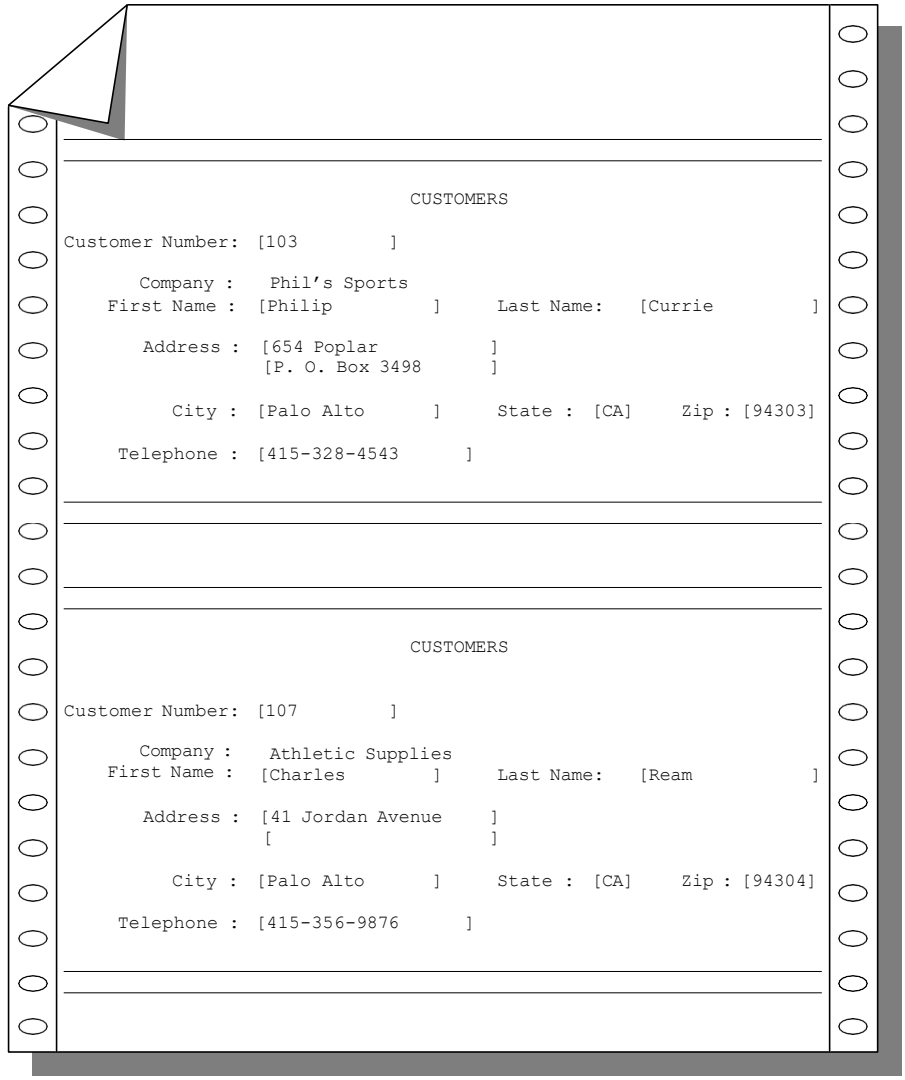
The Screen-format option just produces a copy of the displayed page of the screen form. This output is fine if you use a single-page form. However, if you want to output data from several pages of a multiple-page form, you must alternate between the Output and Screen options.

When you type s, PERFORM writes the rows to the file. A counter at the bottom of the screen increments as each output row is written to the file:

```
Output record number 1
```

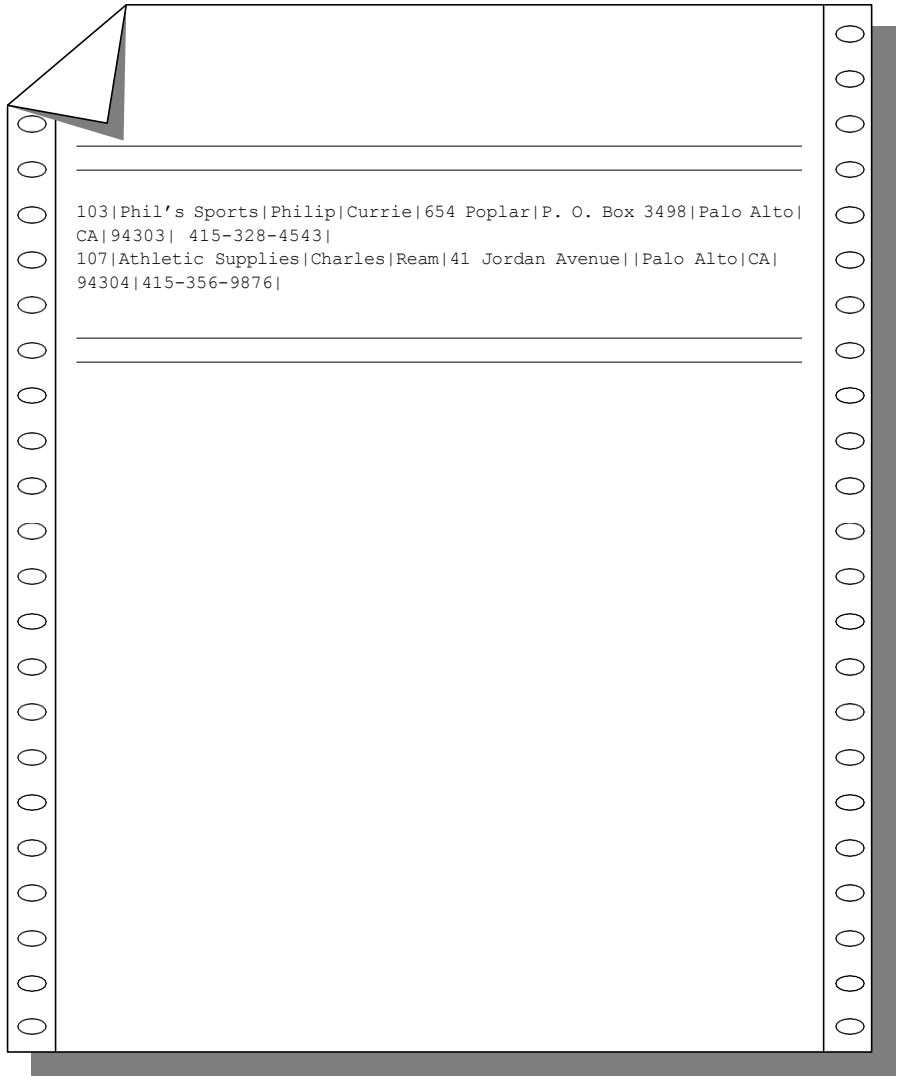
PERFORM displays each row that it writes and updates the counter as it does so.

The following figure illustrates the **perform.out** file, which is based on your search for Palo Alto businesses, when printed in the screen format.



```
                                CUSTOMERS

Customer Number:  [103         ]

       Company :    Phil's Sports
    First Name :  [Philip          ]     Last Name:    [Currie          ]

       Address :  [654 Poplar           ]
                  [P. O. Box 3498       ]

          City :  [Palo Alto       ]     State :  [CA]     Zip : [94303]

     Telephone :  [415-328-4543      ]


                                CUSTOMERS


Customer Number:  [107         ]

       Company :    Athletic Supplies
    First Name :  [Charles         ]     Last Name:    [Ream            ]

       Address :  [41 Jordan Avenue     ]
                  [                     ]

          City :  [Palo Alto       ]     State :  [CA]     Zip : [94304]

     Telephone :  [415-356-9876      ]
```

The following figure illustrates the **perform.out** file when printed in the unload format.



```
103|Phil's Sports|Philip|Currie|654 Poplar|P. O. Box 3498|Palo Alto|CA|94303|
415-328-4543|
107|Athletic Supplies|Charles|Ream|41 Jordan Avenue||Palo Alto|CA|94304|415-
356-9876|
```

Chapter

5

Using Multiple-Table Forms

In This Chapter 5-3

What Is a Multiple-Table Form? 5-3

The Active Table 5-4
- Changing the Active Table 5-4

Join Fields 5-8
- Joins in the orderform Form 5-9
- Verify Joins 5-11
- Lookup Joins 5-12

The Master-Detail Relationship 5-12
- Selecting the Detail Table 5-13
- Returning to the Master Table 5-16

Multiple Tables and Current Lists 5-16
- Using the Current Option 5-16

In This Chapter

This chapter describes the special features of PERFORM that are available when you work with forms that include data from more than one table. The discussion includes the following topics:

- Multiple-table forms
- The active table
- Join fields
- Verify joins
- Lookup fields
- Master-detail relationships
- Multiple-screen forms

Read this chapter if you will be using or creating screen forms that include data taken from more than one table.

What Is a Multiple-Table Form?

A form that includes data from more than one table is called a *multiple-table form*. The maximum number of tables that can be included in the form depends on the open-file limitation on your system.

When you generate a form, you can include data from up to eight different tables on a single form. To use more than one table in a screen form, choose the Select-more-tables option. INFORMIX-SQL displays the list of available tables and lets you select another table. The tables must all be part of the same database.

Two PERFORM features are available only with multiple-table forms:

- *Join fields* let you use a single field to represent data from columns in different tables.
- After you join fields, you can create *master-detail relationships* that make it possible to conduct cross-table queries with a single keystroke.

This chapter presents important background information on the PERFORM program and discusses the various techniques in detail.

The Active Table

PERFORM assigns a sequential number to each database table represented on a screen form. The order in which PERFORM assigns these numbers is determined when you create the form.

When you access PERFORM, it selects the first table as the *active table*. The table name and number appear on the second information line and its fields are enclosed in brackets (or the alternative field delimiters you have selected).

Changing the Active Table

You can use the Table option on the PERFORM menu to select the table with the next sequential number as the new active table. When you do, PERFORM updates the name and number on the information line and shifts the brackets on the screen to the fields in the next table. When the last table is the active table, selecting the Table option selects the first table again.

The Table option automatically selects and displays whichever page of the screen form contains the greatest number of fields from the new active table. To use the Table option, type `t`. If you know the sequential number that PERFORM has assigned to the table you want (the number that appears on the information lines), type it before you select the Table option. For example, you select the third table as the active table by typing `3t`.

The **orderform** form includes data from four of the tables in the demonstration database: **customer**, **orders**, **items**, and **manufact**. When you use the **orderform** form with PERFORM, you see the following screen.

```
PERFORM: Query  Next  Previous  View  Add  Update  Remove  Table  . . .
Searches for the active database table.          ** 1: customer table**
-------------------------------------------------------------------------------

CUSTOMER INFORMATION:
Customer Number: [            ]             Telephone: [                    ]
        Company: [                     ]
     First Name: [           ]               Last Name: [                ]
        Address: [                     ]
                 [                     ]
   City: [               ]  State: [   ]     Zip: [     ]

.......................................................................

ORDER INFORMATION:
     Order Number:                          Order Date:
     Stock Number:                        Manufacturer:

                                              Quantity:
                                         Total Price: $
SHIPPING INFORMATION:
     Customer P.O.:

          Ship Date:                          Date Paid:
```

If you use the Table option to select the next table, you see the following screen.

```
PERFORM: Query Next Previous  View  Add  Update  Remove [Table] . . .
Selects the current table.                    ** 2: orders table**
-------------------------------------------------------------------------------
CUSTOMER INFORMATION:
Customer Number: [           ]             Telephone:
         Company:
      First Name:                          Last Name:
         Address:

   City:                    State:          Zip:

-------------------------------------------------------------------------------
ORDER INFORMATION:
     Order Number: [             ]        Order Date: [           ]
     Stock Number:                       Manufacturer:

                                             Quantity:
                                         Total Price: $
SHIPPING INFORMATION:
     Customer P.O.: [          ]

         Ship Date: [          ]          Date Paid: [          ]
```

The information line shows that **orders** is now the active table. Fields from the **orders** table (on the lower half of the screen) are enclosed in brackets.

Tip: *The Customer Number field remains in brackets when either the* ***customer*** *table or the* ***orders*** *table is the active table. The significance is discussed in* *"Join Fields" on page 5-8*.

If you use the Table option again to select the next table, you see the following screen.

```
PERFORM: Query Next Previous  View  Add  Update  Remove  Table  . . .
Selects the current table.                          ** 3: items table**
-------------------------------------------------------------------------------

CUSTOMER INFORMATION:
Customer Number:                          Telephone:
        Company:
     First Name:                          Last Name:
        Address:

   City:                   State:          Zip:

-------------------------------------------------------------------------------

ORDER INFORMATION:
     Order Number: [            ]        Order Date:
     Stock Number: [           ]       Manufacturer: [  ]

                                            Quantity: [           ]
                                         Total Price: [$                  ]
SHIPPING INFORMATION:
     Customer P.O.:

         Ship Date:                        Date Paid:
```

The second information line shows that **items** is now the active table, and fields from the **items** table (on the lower half of the screen) are in brackets. The Order Number field remains in brackets when either the **orders** table or the **items** table is the active table.

Tip: *Although the* ***manufact*** *table contributes information to the screen form, it is never available as the active table. Thus, you cannot use it for querying, updating, or adding new information.*

Join Fields

Use a multiple-table form whenever you want to have access to the data in more than one table at the same time. Normally, the tables have at least one column that contains the same kind of information. When you create a multiple-table form, you join columns from different tables that contain the same information.

In PERFORM, *joining* means using a single field to represent data taken from more than one table. The display field is called a *join field*, and the columns involved are called *join columns*.

A join field represents data from more than one table and appears in brackets whenever any of the tables containing the data is active. Think of the join field as a bridge that links the related information in two or more different tables.

For example, the **customer_num** column in the **customer** table and the **customer_num** column in the **orders** table both contain customer numbers. Similarly, the **order_num** column in the **orders** table and the **order_num** column in the **items** table both contain order numbers.

As illustrated in the previous section, the Customer Number field is in brackets when either the **customer** table or the **orders** table is the active table. Similarly, the Order Number field remains in brackets when either the **orders** table or the **items** table is the active table.

Joins in the orderform Form

The **orderform** form lets you switch back and forth between customer, order, and item information. The following screen shows how the **orderform** form looks after you use the Query option to locate all customer rows.

```
PERFORM: Query Next  Previous  View  Add  Update  Remove  Table  . . .
Searches the active database table.              ** 1: customer table**
------------------------------------------------------------------------

CUSTOMER INFORMATION:
Customer Number: [101         ]             Telephone: [408-789-8075       ]
        Company: [All Sports Supplies ]
     First Name: [Ludwig    ]              Last Name: [Pauli          ]
        Address: [213 Erstwild Court  ]
                 [                    ]
   City: [Sunnyvale      ]  State: [CA]      Zip: [94086]

------------------------------------------------------------------------

ORDER INFORMATION:
     Order Number:  1002                 Order Date: 06/01/1986
     Stock Number:  4                   Manufacturer: HSK
                                                Husky
                                           Quantity:  1
                                        Total Price:  $960.00
SHIPPING INFORMATION:
     Customer P.O.:  9270
         Ship Date:  06/06/1986           Date Paid:  07/03/1986

28 row(s) found
```

The Customer Number field is a join field. It links the **customer_num** field from the **customer** table and the **customer_num** field from the **orders** table. You can recognize it as a join field because it is displayed in brackets when either **customer** or **orders** is the active table.

The bottom portion of the **orderform** form contains order information for customer 101. The join field lets you simultaneously review information about this customer and information about an order placed by this customer.

To verify the results shown on the screen, refer to the description of the **stores7** demonstration database in the *Informix Guide to SQL: Reference*.

To review the next customer row, use the Next option to advance to the next row in the current list.

```
PERFORM:  Query  Next  Previous  View  Add  Update  Remove  Table  . . .
Shows the next row in the Current List.          ** 1: customer table**
------------------------------------------------------------------------

CUSTOMER INFORMATION:
Customer Number: [102         ]            Telephone: [415-822-1289       ]
        Company: [Sports Spot          ]
     First Name: [Carole    ]              Last Name: [Sadler         ]
        Address: [785 Geary St         ]
                 [                     ]
   City: [San Francisco  ]  State: [CA]      Zip: [94117]

------------------------------------------------------------------------

ORDER INFORMATION:
     Order Number:                        Order Date:
     Stock Number:                       Manufacturer:

                                             Quantity:
                                        Total Price: $
SHIPPING INFORMATION:
     Customer P.O.:

         Ship Date:                         Date Paid:
```

The Sports Spot company is customer 102. The join field lets you see, at the bottom of the screen, information about orders placed by this customer. In this instance, the Sports Spot has yet to place an order, so fields in the bottom half of the screen are empty.

Press the `n` key two more times. The information about the customer and that customer's order(s) changes each time you press `n`.

When you reach customer number 104, your screen looks like this.

```
PERFORM: Query [Next] Previous View Add Update Remove Table . . .
Shows the next row in the Current List.          ** 1: customer table**
-------------------------------------------------------------------------------
CUSTOMER INFORMATION:
Customer Number: [104         ]              Telephone: [415-368-1100       ]
        Company: [Play Ball!           ]
     First Name: [Anthony    ]               Last Name: [Higgins         ]
        Address: [East Shopping Cntr. ]
                 [422 Bay Road         ]
   City: [Redwood City    ]  State: [CA]      Zip: [94026]

-------------------------------------------------------------------------------
ORDER INFORMATION:
     Order Number:  1001                  Order Date: 01/20/1986
     Stock Number:  1                    Manufacturer: HRO
                                                 Hero
                                            Quantity:  1
                                         Total Price:  $250.00
SHIPPING INFORMATION:
     Customer P.O.:  B77836

         Ship Date:  02/01/1986           Date Paid:  03/22/1986
```

Verify Joins

It often is useful to verify that data exists in one table before PERFORM accepts it as input to another. For example, you do not want to enter a nonexistent customer number on the **orderform** form.

A special type of join, called a *verify join*, ensures that you can enter only data that already exists in a database table. You set up verify joins when you create a screen form.

If you enter invalid data in a verify join field, PERFORM displays an error message. For example, if you enter an invalid customer number while **orders** is the active table, PERFORM displays the following message:

```
This is an invalid value -- it does not exist in ''customer'' table
```

You can read about how to create verify joins in Chapter 6, "Creating Your Own Forms."

Lookup Joins

Occasionally, you might want PERFORM to display data that is somehow related to the data you enter on the form. For example, for reference purposes, it might be useful for PERFORM to display a manufacturer's full name when you enter the manufacturer code.

You can use the *lookup* feature to find and display data. The fields in which this data appears are called *lookup fields*; you can never enter data (or search criteria) in these fields.

For example, the second line of the ORDER INFORMATION section of the **orderform** form shows the stock number and manufacturer code for each item ordered. The manufacturer code is part of the **items** table, but the manufacturer name is stored only in the **manufact** table. PERFORM looks it up when you enter the manufacturer code in the field labeled Manufacturer. You can read about how to create lookup joins in "Joining Fields" on page 6-29.

Using the lookup feature to display information from a table does not make that table available for entering data or conducting queries. The **orderform** form shows information from the **manufact** table, but the information appears in a lookup field. You cannot make **manufact** the active table. You cannot query it, and you cannot enter, delete, or update the data stored in it.

The Master-Detail Relationship

When a form includes at least one join field, you can define a master-detail relationship. *Master-detail* refers to the relationship between two tables represented on a screen form. The master-detail relationship simplifies queries that involve data from several tables.

In a master-detail relationship, one table is the master table, and the other is the detail table. You use the Master and Detail options to switch between tables and to conduct automatic database queries.

You can define several master-detail relationships for the same form. A table can have several detail tables but only one master table.

You create a master-detail relationship by including specific instructions in the form specification. This process is described in Chapter 6, "Creating Your Own Forms."

Selecting the Detail Table

When you select the Detail option, PERFORM makes the detail table of the currently active table the new active table. PERFORM automatically queries the detail table, using the value displayed in the join field as the search criterion.

Type `d` to select the detail table and initiate the search. PERFORM displays the following message:

```
Searching  ...
```

while conducting the search and then indicates the results of the query:

```
4  row(s)  found
```

Just as it does with the Query option, PERFORM places the rows it finds into the current list, which you can scan by using the Next and Previous options. (If there is already a current list, the Detail query replaces it with the new results.)

For example, the **orderform** screen form, with the **customer** table as the active table and customer number 104 as the current row, appears in the following screen.

```
PERFORM: Query Next Previous View Add Update Remove Table . . .
Searches the active database table.            ** 1: customer table**
------------------------------------------------------------------------

CUSTOMER INFORMATION:
Customer Number: [104       ]             Telephone: [415-368-1100     ]
        Company: [Play Ball!          ]
     First Name: [Anthony    ]              Last Name: [Higgins        ]
        Address: [East Shopping Cntr. ]
                 [422 Bay Road        ]
   City: [Redwood City    ]  State: [CA]     Zip: [94026]

------------------------------------------------------------------------

ORDER INFORMATION:
     Order Number:  1001                 Order Date: 01/20/1986
     Stock Number:  1                   Manufacturer: HRO
                                                Hero
                                           Quantity:  1
                                       Total Price:  $250.00
SHIPPING INFORMATION:
     Customer P.O.:  B77836

         Ship Date:  02/01/1986           Date Paid:  03/22/1986
```

The second information line indicates that **customer** is the active table, and the screen shows data about customer number 104.

The **customer** table is the master table for the **orders** table. You can obtain information about the orders placed by customer 104 by using the Detail option. When you type `d`, PERFORM switches tables and queries the database, using the value in the Customer Number field as its search criterion.

You see the following screen.

```
PERFORM:  . . .  Screen  Current  Master  [Detail]  Output  Exit
Selects a detail table of the current table.        ** 2: orders table**
-------------------------------------------------------------------------

CUSTOMER INFORMATION:
Customer Number: [104        ]              Telephone: 415-368-1100
        Company:  Play Ball!
     First Name:  Anthony                    Last Name: Higgins
        Address:  East Shopping Cntr.
                  422 Bay Road
   City:  Redwood City       State:  CA       Zip:  94026

-------------------------------------------------------------------------

ORDER INFORMATION:
     Order Number: [1001         ]        Order Date: [01/20/1986 ]
     Stock Number:  1                    Manufacturer: HRO
                                                       Hero
                                            Quantity:  1
                                         Total Price:  $250.00
SHIPPING INFORMATION:
     Customer P.O.: [B77836     ]
         Ship Date: [02/01/1986]           Date Paid: [03/22/1986]

 4 row(s) found
```

PERFORM selects the detail table, **orders**, and automatically initiates a query based on the join field (the one labeled Customer Number).

PERFORM found rows for four orders placed by customer 104. Use the Next and Previous options to scan these rows.

If you use the Detail option when no detail table exists for the active table, PERFORM displays this message:

```
No detail table has been specified for this table
```

If no explicit master/detail relationship exists, PERFORM displays an error message when you use the Master option or the Detail option without a table number.

Returning to the Master Table

The Master option returns you to the master table after you use the Detail option to conduct a query. The Master option does not initiate a database query; it simply returns you to the table that was active before you selected the Detail option.

Type `m` to return to the master table. If you use the Master option when there is no master table for the active table, PERFORM displays this message:

```
No master table has been specified for this table
```

Multiple Tables and Current Lists

PERFORM maintains a separate current list for each table included on a screen form. You can query one table, switch tables, query the second table, then return to the first table, and still use the Next and Previous options to review the results of the original query.

Using the Current Option

If your form includes a join field, you might occasionally lose your place in one of the current lists. Should this occur, you can use the Current option to return to the current position in the current list.

The Current option also serves a second purpose. On multiuser computer systems, more than one person can use a database at the same time. Another user could modify a row while it appears on your screen.

When you run the Current option, PERFORM rereads the row from the database and displays the most up-to-date information. If you suspect someone might have changed the currently displayed row, just select the Current option.

Chapter

6

Creating Your Own Forms

In This Chapter 6-3

What Is a Form Specification? 6-3
- Default Form Specifications 6-4
- How to Create a Default Form 6-4
- How to Modify an Existing Form 6-4

How to Create a Form 6-5
- Naming the Form 6-6
- Choosing Tables to Include in the Form 6-6
- Compiling the Default Form Specification 6-7
  - What To Do If There Are Errors 6-8
  - PERFORM Uses Two System Files 6-9
- A Sample Default Form 6-9
- How to Modify an Existing Form 6-10

What the Form Specification Contains 6-11
- An Example Form Specification 6-11
  - The DATABASE Section 6-12
  - The SCREEN Section 6-12
  - The TABLES Section 6-13
  - The ATTRIBUTES Section 6-13
- The customer Form Specification 6-14
- Comparing the custom and customer Forms 6-15
  - The DATABASE and TABLES Sections 6-16
  - The SCREEN Section 6-17
  - The ATTRIBUTES Section 6-17
- Using FORMBUILD Attributes 6-19

How to Enter Attributes 6-19
The Field Tag 6-19
The Equal Sign 6-19
The Column Name 6-20
The Attribute List 6-20
Using Blank Space 6-21
Displaying Field Text in Color 6-21
Making a Field INVISIBLE 6-22
Displaying Comments on the Screen 6-23
Changing Data from Lowercase to Uppercase 6-24
Changing Default Field Values 6-24
Using Today's Date as the Default 6-25
Using the Current Time as the Default 6-26
Requiring User Input 6-26
Specifying Acceptable Values 6-26
The Lowest Value Must Appear First 6-27
Using INCLUDE with CHAR Fields 6-27
Verifying Input 6-28
Joining Fields 6-29
Verify Joins 6-31
Lookup Joins 6-31
The INSTRUCTIONS Section 6-33

Calling C Functions from PERFORM 6-34

In This Chapter

Earlier chapters described how to enter data and query a database using existing screen forms. This chapter explains how to create your own forms. The following topics are discussed:

- Creating a default form specification
- Using a default screen form
- Modifying a default form specification

Read this chapter to learn how to create and customize your own screen forms.

For more information about the features introduced in this chapter, refer to the *INFORMIX-SQL Reference Manual*.

What Is a Form Specification?

Creating a screen form is a two-step process. First, you create a *form specification*, and then you compile it. A form specification is a system file containing instructions that describe how you want the form to look and what database information you want it to access. Compiling converts the instructions in a form specification file into a format that INFORMIX-SQL can use.

You can create a form specification from scratch using a system editor. One way to do this is to use the New option on the FORM menu. Selecting the New option calls the system editor and places you in a new editing buffer. However, it is often easier to first create a *default form specification* using the Generate option on the FORM menu. You can then modify the default form specification to meet your precise requirements.

Default Form Specifications

When you create a default form specification, you indicate which tables contain the data you want to use with the form. INFORMIX-SQL then creates a default form specification.

The default specification contains the basic instructions you need to create the form. You can use the default specification or you can modify it in order to take advantage of the more powerful features of PERFORM.

How to Create a Default Form

Use the Generate option on the FORM menu to create a default form specification. INFORMIX-SQL asks for the name of the table or tables you want to use with the form, creates a default form specification file, and then compiles the specification.

Once the specification file is compiled, you can use the form in PERFORM. You repeat these steps for each form you create.

How to Modify an Existing Form

To change the default form, you must modify the specification file and then recompile it. The FORM menu includes an option that lets you do this.

Until you recompile the specification file, any changes you make to it will not show up on the screen form itself. This chapter describes in detail the process for modifying an existing form.

How to Create a Form

Before you create a screen form, use the Database option on the Main menu to select the database that contains the data you want to use with the form.

After you select a database, choose the Form option on the Main menu.

```
INFORMIX-SQL:  Form  Report Query-Language User-menu Database Table Exit
Run, Modify, Create, or Drop a form.

                          stores7                    Press CONTROL-W for Help
```

INFORMIX-SQL displays the FORM menu. Select the Generate option.

```
FORM: Run Modify   Generate  New  Compile  Drop  Exit
Generate a default form.

                          stores7                    Press CONTROL-W for Help
```

INFORMIX-SQL displays the GENERATE FORM screen.

```
GENERATE FORM >>
Enter the name you want to assign to the form, then press Enter.

                          stores7                    Press CONTROL-W for Help
```

Naming the Form

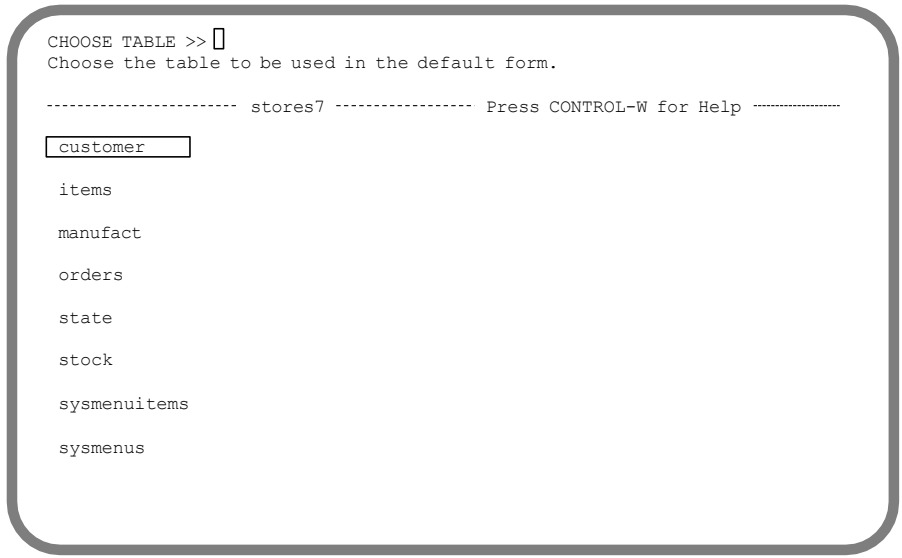
You assign a name to each form when you create it. You can use a name up to 10 characters long. If you specify the name of a form that already exists, INFORMIX-SQL displays the following message, and you must enter a different name:

```
Form with the same name already exists.
```

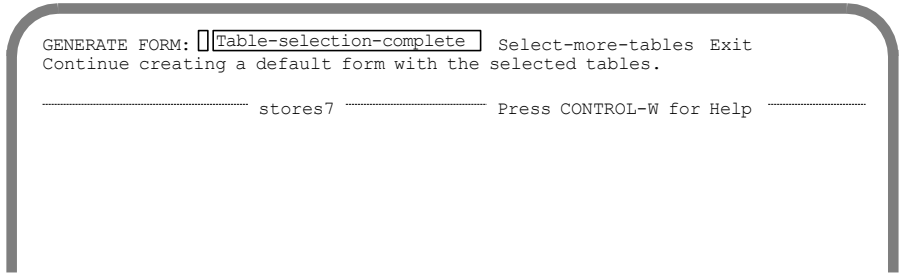
Type the name you want to assign to the form and press RETURN.

Choosing Tables to Include in the Form

INFORMIX-SQL asks for the name of the table in the current database containing the information you want to use in the form.



Choose a table by typing or highlighting its name and pressing RETURN. INFORMIX-SQL displays the GENERATE FORM menu.



The Generate option lets you include data from up to eight different tables on a single form. To use more than one table in a screen form, choose the Select-more-tables option. INFORMIX-SQL displays the list of available tables and lets you select another table.

When you have chosen all the tables you want to include in the form, select the Table-selection-complete option. INFORMIX-SQL then creates the default form specification.

Tip: *To include more than eight tables in a form, create the form from scratch using the New option on the FORM menu. For more information about the FORMBUILD transaction Form generator, see the "INFORMIX-SQL Reference Manual."*

If you decide you do not want to generate a default form, use the Exit option to return to the FORM menu.

Compiling the Default Form Specification

When you select the Table-selection-complete option, INFORMIX-SQL creates a default form specification file and compiles the specifications.

You must compile the form specification before you can use the form. Compiling allows INFORMIX-SQL to convert the specifications into a format that PERFORM can understand. INFORMIX-SQL automatically compiles the specification file when you select the Table-selection-complete option.

During compilation, messages appear on the screen. When the compilation is successfully completed, the following message flashes across the bottom of the screen, and you are returned to the FORM menu:

```
Form was successfully compiled
```

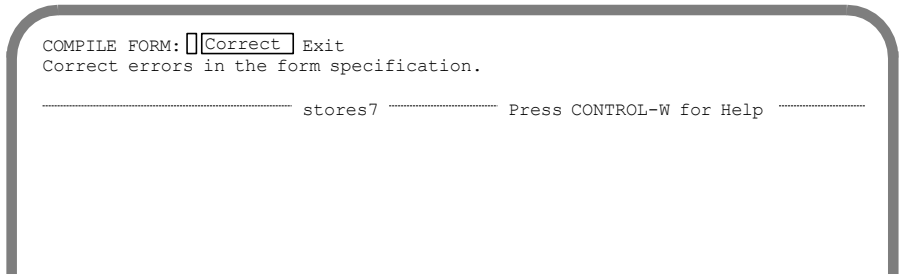
You can now use the form. Later in this chapter, you will learn about the modifications you can make to the default form specification file.

What To Do If There Are Errors

If you use the Modify option to modify a default specification or the New option to create a form specification from scratch, you might introduce errors. If FORMBUILD, the INFORMIX-SQL program that compiles form specifications, cannot compile the specification, the following message appears at the bottom of the screen:

```
832: Error(s) found in Form specifications.
```

INFORMIX-SQL then displays the COMPILE FORM menu.



Choose the Correct option to correct the errors. INFORMIX-SQL prompts you for the name of the system editor you want to use and displays the form specification file within the editor. (See "Using the SQL Editor" on page 7-9 for a discussion of how to use the system editor.) The file now contains error messages describing any errors you made.

Tip: *The text of all error messages and suggestions for corrections are included in the Informix Error Messages documentation in Answers OnLine.*

Correct any errors and save the specification file. You do not need to delete the error messages themselves; FORMBUILD automatically removes them from the file.

When you save the corrected form specification file, INFORMIX-SQL redisplays the MODIFY FORM menu if you are modifying a form or the NEW FORM menu if you are creating a form from scratch. You can then recompile the file and generate a usable screen form.

PERFORM Uses Two System Files

Once you compile and save a form, two system files are created. The name of each file includes the name you assigned to the form and a three-letter extension. INFORMIX-SQL stores these two files in the current directory.

The first file contains the form specifications and has the extension **.per**. This file contains the instructions that control the appearance and behavior of the form. Use a system editor to modify this specification file if you elect to customize the form for your particular requirements.

The second file contains a compiled version of the form specifications and has the extension **.frm**. PERFORM uses this file when you work with a screen form.

Tip: *If you create a default form specification from the operating-system command line, you can specify a full pathname and have INFORMIX-SQL store the two files in that directory. For command-line syntax, see the "INFORMIX-SQL Reference Manual."*

A Sample Default Form

When you create a default form, INFORMIX-SQL includes one field for each column in the table or tables you select. The fields are labeled with the column names. If you select more than one table, each table receives a separate page on the screen form. Use the Screen option in PERFORM to move from page to page.

Follow these steps to create a default form named **custom** that works with the **customer** table in the **stores7** demonstration database:

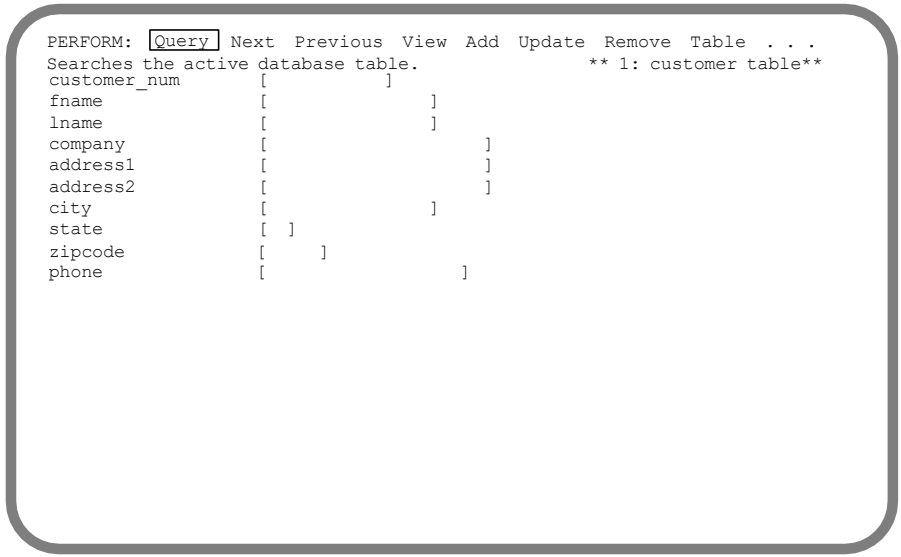
1. Use the Database option on the Main menu to select the **stores7** demonstration database.
2. Select the Form option on the Main menu followed by the Generate option on the FORM menu.
3. Enter `custom` as the name of the form.

4. Select the **customer** table from the list of tables that INFORMIX-SQL displays.
5. Select the Table-selection-complete option.

   INFORMIX-SQL compiles the form specification and displays the FORM menu.

Now you can use the **custom** form with PERFORM. Select Run on the FORM menu and then choose the **custom** form from the list that INFORMIX-SQL displays.

INFORMIX-SQL displays the following default form.



How to Modify an Existing Form

To change a form that already exists, you must modify the form specification and then recompile the form. Use the Modify option on the FORM menu for this purpose.

The next part of this chapter describes the sections that make up a form specification. As you read this discussion, you can modify the **custom** form to include desired changes. For more information on the Modify option and the structure of a form specification, see the *INFORMIX-SQL Reference Manual*.

What the Form Specification Contains

Each form specification must include at least these four sections:

- A DATABASE section
- A SCREEN section
- A TABLES section
- An ATTRIBUTES section

These sections appear in every default form specification. A form specification file that is not a default form specification also can contain an optional fifth section called the INSTRUCTIONS section.

An Example Form Specification

The default form specification for the **custom** form appears in Figure 6-1.

Figure 6-1
The custom Form Specification

```
database stores7
screen
{
customer_num          [f000         ]
fname                 [f001              ]
lname                 [f002              ]
company               [f003                   ]
address1              [f004                   ]
address2              [f005                   ]
city                  [f006              ]
state                 [a0]
zipcode               [f007  ]
phone                 [f008                ]
}
end
tables
customer
```

```
attributes
f000 = customer.customer_num;
f001 = customer.fname;
f002 = customer.lname;
f003 = customer.company;
f004 = customer.address1;
f005 = customer.address2;
f006 = customer.city;
a0  =  customer.state;
f007 = customer.zipcode;
f008 = customer.phone;
end
```

The form specification contains all the information PERFORM needs for the **custom** form. It includes DATABASE, SCREEN, TABLES, and ATTRIBUTES sections; it does not include an INSTRUCTIONS section.

You can use the optional END keyword to mark the end of sections in the form specification file. Some users find it helpful to indicate the close of a section with END. The forms included with the **stores7** demonstration database include the END keyword.

The DATABASE Section

Each form specification begins with a DATABASE section that specifies the database with which the form is designed to work. The DATABASE section consists of the DATABASE keyword followed by the name of the database.

In a default form specification, the DATABASE section specifies the database that was the current database when you created the specification. In the **custom** specification, the following line indicates that the **custom** form works with the **stores7** demonstration database:

```
database  stores7
```

The SCREEN Section

The SCREEN section shows how the form appears when you use it with PERFORM. However, the form appears differently on the screen than in the specification file.

- The SCREEN section of the form specification is enclosed within a set of curly braces { } and begins with the SCREEN keyword. You can use the optional END keyword to indicate the end of the section.

- The SCREEN section of the form specification file includes a *field tag* for each field on the form. These tags appear between the brackets that delimit the fields. Each field must have a field tag. FORMBUILD assigns field tags automatically when you create a default form specification. In the **custom** specification file, the field tags are **f000**, **f001**, **f002**, and so on.

Except for field tags and special graphics characters, PERFORM displays everything between the braces exactly as it appears in the specification. For information on how to use graphics characters in forms, refer to the *INFORMIX-SQL Reference Manual*.

The TABLES Section

The TABLES section lists the name of each table that contains data used by the form. It appears immediately after the SCREEN section.

The TABLES section in a default form specification lists the table or tables that you chose when you created the specification. In the **custom** specification file, the following lines indicate that the **custom** form uses data from only the **customer** table:

```
tables
customer
```

The ATTRIBUTES Section

The ATTRIBUTES section explicitly associates each field on the form with a column in one of the tables listed in the TABLES section. It begins with the ATTRIBUTES keyword and lists each field tag and the name of the column to which the tag corresponds.

For example, the second line of the ATTRIBUTES section in the **custom** specification assigns the **f001** field to the **fname** column in the **customer** table:

```
f001 = customer.fname;
```

This statement means that PERFORM will display **fname** values in the **f001** field when you use the form to query the database. PERFORM will store any values you enter in the **f001** field in the **fname** column.

Tip: *When you create a default form specification, FORMBUILD automatically assigns both table and column names to each field. In most cases, the table name is optional. However, you must specify the table name when you refer to a column name that appears in more than one of the tables used in the form. You might prefer to adopt the "table.column" naming convention when you create all of your screen forms to ensure that all the column references are unambiguous.*

The customer Form Specification

You can use the **custom** form to enter and retrieve customer information. Because it is a default form, however, it does not take advantage of the more powerful features of PERFORM.

The **customer** form in the **stores7** demonstration database contains precisely the same information as the default **custom** form, but the two forms look and act differently. Here is how the **customer** screen form appears when used by PERFORM.

```
PERFORM: Query Next Previous View Add Update Remove Table . . .
Searches the active database table.              ** 1: customer table**
_____________________________________________________________________
_____________________________________________________________________

                                CUSTOMERS

Customer Number:  [           ]

       Company :  [                    ]
     First Name:  [               ]     Last Name:   [                ]

        Address :  [                    ]
                   [                    ]

           City :  [               ]     State :  [  ]    Zip :  [     ]

      Telephone :  [                  ]

_____________________________________________________________________
_____________________________________________________________________
```

The **customer** form was created by modifying and recompiling the **custom** specification file.

Comparing the custom and customer Forms

The form specification for the **customer** form appears in Figure 6-2. For comparison, Figure 6-3 on page 6-16 shows the form specification for the **custom** form.

Figure 6-2
The customer Form Specification

```
database
    stores7

screen
{
_____________________________________________________________________________
_____________________________________________________________________________

                                CUSTOMERS

Customer Number:  [f000       ]

        Company:  [f001                ]
     First Name:  [f002           ]       Last Name:  [f003           ]

        Address:  [f004                ]
                  [f005                ]

           City: [f006            ]       State: [a0]     Zip: [f007 ]

      Telephone: [f008               ]

_____________________________________________________________________________
_____________________________________________________________________________
}
end

tables
     customer

attributes

f000 = customer_num;
f001 = company, reverse;
f002 = fname, comments = "Please enter first name if available";
f003 = lname;
f004 = address1;
f005 = address2;
f006 = city;
a0  = state, upshift, autonext, include = ("CA", "OR", "NV", "WA"),
       comments = "legal states are CA, OR, NV, or WA";
f007 = zipcode, autonext;
f008 = phone, picture = "###-###-####XXXXXX";

end
```

Figure 6-3
The custom Form Specification

```
database stores7
screen
{
customer_num          [f000        ]
fname                 [f001             ]
lname                 [f002             ]
company               [f003                  ]
address1              [f004                  ]
address2              [f005                  ]
city                  [f006             ]
state                 [a0]
zipcode               [f007  ]
phone                 [f008               ]
}
end
tables
customer
attributes
f000 = customer.customer_num;
f001 = customer.fname;
f002 = customer.lname;
f003 = customer.company;
f004 = customer.address1;
f005 = customer.address2;
f006 = customer.city;
a0  =  customer.state;
f007 = customer.zipcode;
f008 = customer.phone;
end
```

The following sections discuss the similarities and differences between the two form specifications.

The DATABASE and TABLES Sections

Because both forms are designed to work with the **customer** table in the **stores7** demonstration database, the DATABASE and TABLES sections look the same in both specification files.

The SCREEN Section

The two forms look quite different on the screen:

- Fields on the **custom** form appear in a single column. The **customer** form places them across the full width of the screen.
- The **custom** form labels fields with the names of the columns to which they correspond (**customer_num**, **fname**, and so on). The **customer** form uses labels like Name, Address, and Telephone.
- The **customer** form also includes horizontal lines to make the screen more attractive.

These differences exist because of differences in the SCREEN sections of the two form specification files.

You can use your system editor to make any changes you want to the appearance of a form. For instance, you can move fields, change the labels that appear alongside the fields, or add horizontal and vertical lines.

You also can use special graphics characters in the SCREEN section to place boxes and rectangles on your screen form. The meaning of these graphics characters is terminal-dependent. For information on how to use graphic characters in forms, refer to the *INFORMIX-SQL Reference Manual*.

Important: *Be careful not to decrease the widths of the fields when you modify the SCREEN section, or they may no longer be wide enough to display the data in your database. The width of a display field is the distance between the brackets that surround it.*

The ATTRIBUTES Section

In the **custom** specification, the ATTRIBUTES section simply establishes relationships between field tags and database columns. The **customer** specification assigns different database columns to the field tags and includes attributes that specify how the fields should behave.

Assigning Field Tags

The order in which field tags appear in the ATTRIBUTES section determines the sequence of fields to which the cursor moves during a Query, Add, or Update action. You can change the order of field tags—or the assignment of database columns to field tags—in the ATTRIBUTES section to make the form more effective.

For example, the assignment of database columns to field tags in the **custom** and **customer** specifications is different because the layout of fields on the screen is different in each form. In the **customer** form, the Company field (corresponding to the **company** column in the **customer** table) appears as the second field in the form and is assigned field tag **f001**. In the **custom** form, the Company field appears as the fourth field in the form and is assigned field tag **f003**.

Using Attribute Keywords

The ATTRIBUTES section of the form specification file assigns each field on the form to a column in one of the tables listed in the TABLES section. In the **custom** and **customer** forms, the fields are linked to columns in the **customer** table. For example, in the **custom** form, the line:

```
f003 = customer.company;
```

assigns the field **f003** to the column **company** in the **customer** table. The **customer** screen form, however, includes specific attributes that define the behavior of individual fields. For example, the **customer** specification assigns the REVERSE attribute to the **f001** field:

```
f001 = customer.company, reverse;
```

The REVERSE attribute causes PERFORM to display the field in reverse video (dark letters on a light background). The next section explains how to use attributes like REVERSE to customize the way your forms look and behave.

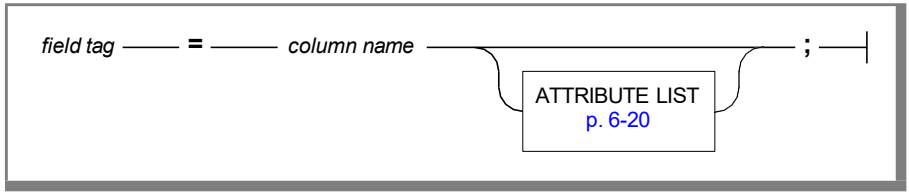
Using FORMBUILD Attributes

You can customize each field on a form by adding one or more keywords, called *attributes*, in the ATTRIBUTES section. Some attributes, like REVERSE, simply affect the way a field appears. Others specify actions that you want PERFORM to take when you enter data in the field. For example, if you use the UPSHIFT attribute in a CHAR field, PERFORM automatically converts any data you enter in the field to uppercase. (The UPSHIFT attribute is used in the **customer** form in the State field.)

This section explains how to use several more frequently used attributes. Once you understand how to use attributes, you can refer to the *INFORMIX-SQL Reference Manual* for additional information.

How to Enter Attributes

The format you use for entries in the ATTRIBUTES section is always the same.



Only the *field tag*, equal sign, *column name*, and semicolon are required; an attribute list is optional.

The Field Tag

The field tag appears between brackets in the SCREEN section of the specification. In the **customer** form, these tags are **f000**, **f001**, **f002**, and so on.

The Equal Sign

The equal sign associates the field indicated by the field tag with the column you specify (the column name).

The Column Name

The column name indicates the column whose data you want to appear in the field indicated by the field tag. If the same column name appears in more than one of the tables in the TABLES section, you must enter both the table name and the column name. For example, to refer to the **customer_num** column in the **customer** table, enter:

```
customer.customer_num
```

To refer to the **customer_num** column in the **orders** table, enter:

```
orders.customer_num
```

Default form specifications include the table name for every field in the ATTRIBUTES section.

The Attribute List

The attribute list can be one attribute or a series of attributes. Use commas to separate each attribute from other attributes and from the column name. The attribute list ends with a semicolon.

The attribute list for a field can occupy more than one line in the form specification. You can break a line at any blank space, except within the text of a comment.

Attributes include the following list:

- AUTONEXT
- COLOR
- COMMENTS
- DEFAULT
- DOWNSHIFT
- FORMAT
- INCLUDE
- INVISIBLE
- LOOKUP
- NOENTRY
- NOPICTURE

- PICTURE
- QUERYCLEAR
- REQUIRED
- REVERSE
- RIGHT
- UPSHIFT
- VERIFY
- W0RDWRAP
- ZEROFILL

The following sections describe how to specify the most common attributes. For more information on the attributes, see the *INFORMIX-SQL Reference Manual*.

Using Blank Space

You can use blank lines, spaces, and tabs freely in the form specification file, although the text of a comment attribute cannot be broken across lines. You do not have to use blank lines in your specification file, but using them makes the instructions easier to read.

Displaying Field Text in Color

Use the COLOR attribute to display field text with one of the following colors or attributes.

Type `COLOR` followed by an equal sign and the desired color or attribute, as follows:

- WHITE
- YELLOW
- MAGENTA
- RED
- CYAN
- GREEN
- BLUE

- BLACK
- REVERSE
- BLINK
- UNDERLINE
- LEFT

For example, if you were to assign color values to the **customer** specification file, the lines:

```
f004=address1, color=yellow;
f005=address2,  color=yellow;
```

would cause PERFORM to display customer addresses in yellow type. You can display field text in dark letters on a colored background by combining a color name and REVERSE. You also can use the optional WHERE keyword to specify a condition under which the attribute takes effect. For example, you might want to combine attributes to call special attention to a particular customer as follows:

```
f009  =  customer.customer_num,
      color  =  red  reverse  where  f009  =  104;
```

For more information about the COLOR attribute, see the *INFORMIX-SQL Reference Manual*.

Making a Field INVISIBLE

Use the INVISIBLE attribute to display nothing when information is entered into a field. The following example defines the **phone** field as INVISIBLE:

```
f007  =  customer.zipcode;
f008  =  customer.phone,  INVISIBLE;
```

If you assign both the INVISIBLE and COLOR attributes to a field, INFORMIX-SQL ignores the COLOR attribute. However, if you specify COLOR=REVERSE, INFORMIX-SQL displays the field in reverse video and makes the contents of that field invisible.

If you assign both the INVISIBLE and PICTURE attributes to a field, INFORMIX-SQL does not display the picture pattern.

You can also see an INVISIBLE field by using the Output option on the PERFORM menu. The Output option captures the information displayed on a screen. For information on displaying INVISIBLE fields using the PERFORM Output option, see Chapter 4, "Querying a Database," and the *INFORMIX-SQL Reference Manual*.

Displaying Comments on the Screen

Use the COMMENTS attribute to display a comment on the comment line whenever the cursor moves into a field. For example, you might use a comment to indicate what type of data to enter in a field.

Type `comments`, followed by an equal sign, followed by the text of the comment enclosed in quotation marks. The entire text must fit on one line. For example, in the **customer** specification file:

```
f002 = fname, comments = "Please enter first name if available";
```

causes PERFORM to display the comment appearing in quotation marks whenever the cursor moves into the Name field, as shown in the following screen.

```
QUERY: ESCAPE queries. INTERRUPT discards query. ARROW keys move cursor.
Searches the active database table.          ** 1: customer table**
_____________________________________________________________________
_____________________________________________________________________

                                CUSTOMERS

Customer Number:  [           ]

       Company :  [                ]
     First Name:  [          ]     Last Name:    [                ]

        Address :  [                    ]
                   [                    ]

           City :  [               ]     State :  [  ]    Zip :  [     ]

      Telephone :  [                  ]


_____________________________________________________________________
_____________________________________________________________________

Please enter first name if available.
```

Changing Data from Lowercase to Uppercase

The DOWNSHIFT and UPSHIFT attributes change the case of the data you enter in a CHAR field. UPSHIFT converts anything you enter to uppercase letters; DOWNSHIFT converts entries to lowercase letters. For example:

```
a0 = customer.state, upshift;
```

causes PERFORM to convert two-letter state abbreviations to uppercase when you enter them into the State field.

Uppercase and lowercase letters have different ASCII values. Storing some kinds of CHAR data in one or the other format can simplify sorting data and querying a database. For example, without the UPSHIFT attribute, you might have to search for **CA**, **ca**, **Ca**, and **cA** to find records for customers with locations in California.

The UPSHIFT and DOWNSHIFT attributes are used only in CHAR fields. If you attempt to use them elsewhere, FORMBUILD ignores them.

Changing Default Field Values

PERFORM automatically establishes a default value of NULL for every field on a screen form. When you add data in PERFORM, the value you enter into a field replaces its default NULL value. If you do not enter data into a field, PERFORM normally stores the NULL default value in the table into which you are adding data.

When you use the Add option, PERFORM displays the screen form with the following characters in the indicated fields.

Type of Field	Default Display
Character	Blanks
Number	Blanks
Date	Blanks

(1 of 2)

Type of Field	Default Display
Money	$
Datetime	Blanks
Interval	Blanks

(2 of 2)

You can avoid NULL values by including the WITHOUT NULL INPUT keywords in the DATABASE section of the form specification. When you add a new row to the database, PERFORM inserts blanks in CHAR and DATE fields without data, and zeros in number and MONEY fields without data.

You can use the DEFAULT attribute to replace the NULL default value of a field. For example, you might change the default for a phone number field to show your local area code.

Type `default`, followed by an equal sign and the value you want to assign, as the default value for the field. When you specify a DATE value, a DATETIME value in character-string format, or a CHAR value that contains punctuation, symbols, or blank spaces, you must enclose the value in quotation marks.

For example, if most customers are located in California, you might want to set a default for the State field as follows:

```
a0=customer.state,  upshift,  default="CA";
```

Using Today's Date as the Default

If you enter `default=today` for a DATE field, PERFORM uses the current date as the default value. In the **orderform** form specification file, the following lines appear in the ATTRIBUTES section:

```
o12 = order_date,
       default = today;
```

The Order Date field on the **orderform** form defaults to the current date.

The TODAY attribute is used only in DATE fields. If you attempt to use it elsewhere, FORMBUILD generates an error message.

Using the Current Time as the Default

You can use CURRENT as the default wherever you want the degree of precision that a DATETIME value provides. For example, say your database contains a field called **order_time**. If you want to timestamp the orders you receive, from year to minute, you can set a default for the **order_time** field like this:

```
fo13 = order_time,
        default = current YEAR TO MINUTE;
```

The CURRENT attribute is used only in DATETIME fields. If you attempt to use it elsewhere, FORMBUILD generates an error message.

Requiring User Input

When a column contains essential data, you might want to require a user to enter data for that column. For example. to require a phone number entry for each customer, you would alter the **customer** screen form specification file as follows:

```
f008   = phone, required;
```

This statement ensures that a telephone number appears in each row of the **customer** table.

When you assign the REQUIRED attribute to a field, PERFORM requires that you enter data in that field before it accepts a new row for a table. If you try to save a new row without entering data in a required field, PERFORM displays the following message:

```
This field requires an entered value
```

Tip: *The REQUIRED attribute works only with the Add option. You can remove data from required fields with the Update option.*

Specifying Acceptable Values

Use the INCLUDE attribute if you want to specify values that are acceptable as input to a field. PERFORM accepts only those values listed with the INCLUDE attribute.

Type include followed by an equal sign and the acceptable values enclosed in parentheses. You can enter one value or a list of values separated by commas, and you can use the TO keyword to specify ranges of acceptable values. For example, in the **customer** specification file, the following example specifies the values CA, OR, NV, and WA as acceptable values for the **a0** (State) field:

```
a0 = state, upshift, autonext, include = ("CA", "OR", "NV", "WA");
        comments = "legal states are CA, OR, NV, or WA";
```

The comment line indicates that PERFORM accepts no other values.

The Lowest Value Must Appear First

When you specify a range of acceptable values, the lowest value must appear first. That is, you must specify a range as being from 1 to 50, not as 50 to 1. For example, in the **sample** form included with the demonstration database, the following INCLUDE attribute specifies the values 1 to 50:

```
i18 = items.quantity, include = (1 to 50),
comments = "Acceptable values are 1 through 50" ;
```

You can also specify multiple ranges. For example, the following attribute specifies the ranges 1 to 50 and 60 to 100:

```
include = (1 to 50, 60 to 100);
```

Using INCLUDE with CHAR Fields

You also can assign the INCLUDE attribute to CHAR fields. For example, you could specify a yes/no field as follows:

```
field-tag = column,
            upshift,
            include = (Y, N);
```

You can specify a CHAR range, too. PERFORM evaluates ranges based on the ASCII values for the low and high limits of the range. The following example prevents PERFORM from accepting values other than A1, A2, A3, A4, and A5:

```
field-tag = column,
            include (A1 to A5);
```

When you include a CHAR value that contains punctuation marks, symbols, or blank spaces, you must enclose the entire value in quotation marks:

```
field-tag = column,
     include = ("sporting goods", "goods, sporting");
```

Tip: *PERFORM displays an error message if you enter a value not specified with the INCLUDE attribute:*

```
This value is not among the valid possibilities.
```

The message does not specify the acceptable values. You might consider using the COMMENTS attribute to display a message reminding users of the acceptable values.

Verifying Input

If a field contains particularly critical data, you can use the VERIFY attribute to require users to enter the same value twice before PERFORM accepts it. That way, your database is less likely to contain incorrect data because of typographical errors.

For example, if you were entering personnel information into a form, you might want to force the user to enter the employee's salary twice before PERFORM accepts it. The specification file would include the following lines:

```
f006 = personnel.salary,
       verify;
```

When you enter a number in the Salary field and press RETURN, PERFORM clears the field and displays this message:

```
Please type again for verification
```

You must enter *exactly* the same data, character for character, a second time. (Remember that `49500` and `49500.00` are not exactly the same.)

You can use many other attributes in addition to the ones discussed in this section. For a complete list of PERFORM attributes, see the *INFORMIX-SQL Reference Manual*.

Joining Fields

If you include data from more than one table on a form, you probably will want to include a join field. To create a join field, equate two database columns with the same field tag in the ATTRIBUTES section. For example, the following line specifies that **field-tag** is a join field that joins the **col1** and **col2** columns:

```
field-tag = col1 = col2;
```

In the **orderform** form specification file,

```
c1 = *customer.customer_num = orders.customer_num;
```

specifies that **c1** is a join field that joins the following columns: **customer.customer_num** and **orders.customer_num.** (The use of the asterisk in a join is described in "Verify Joins" on page 6-30.)

With a join field, you can assign attributes that take effect only when one table is active, as well as attributes that are always in effect. If you want an attribute to apply at all times, enter it after the last column name. For example,

```
field-tag = col1 = col2, include = (1 to 100);
```

joins **col1** to **col2** in the tag field and prevents PERFORM from accepting values less than 1 or greater than 100 for either **col1** or **col2**.

You also can specify an attribute that applies only when one file or the other is active. To do so, you must enter a separate attribute list for each column name. For example,

```
field-tag = col1, required, upshift;
          = col2, upshift;
```

again joins **col1** and **col2**. When the **col1** table is active, PERFORM requires that you enter a value and converts that value to uppercase letters. When the **col2** table is active, you do not have to enter a value, although PERFORM will still convert any CHAR value you enter to uppercase letters. You can find additional information about the placement of attributes in the *INFORMIX-SQL Reference Manual*.

The **orderform** form contains data from four of the tables in the **stores7** demonstration database. Figure 6-4 shows the form specification for the **orderform** form.

Figure 6-4
The orderform Form Specification

```
database stores7

screen
{
================================================================================
CUSTOMER INFORMATION:
Customer Number: [c1         ]             Telephone: [c10               ]
        Company: [c4                  ]
     First Name: [c2        ]              Last Name: [c3            ]
        Address: [c5                  ]
                 [c6                  ]
   City: [c7             ]  State: [c8]     Zip: [c9    ]
================================================================================
ORDER INFORMATION:
     Order Number: [o11         ]        Order Date: [o12       ]
     Stock Number: [i13        ]       Manufacturer: [i16]
                                                     [manu_name      ]
                                            Quantity: [i18       ]
                                         Total Price: [i19               ]
SHIPPING INFORMATION:
     Customer P.O.: [o20       ]
         Ship Date: [o21       ]          Date Paid: [o22       ]

}
end
tables
customer orders
items manufact

attributes

c1 = *customer.customer_num = orders.customer_num;
c2 = fname,
       comments = "Please enter initial if available ";
c3 = lname;
c4 = company;
c5 = address1;
c6 = address2;
c7 = city;
c8 = state, upshift, autonext,
       include = ("CA","OR","NV","WA");
c9 = zipcode;
c10 = phone, picture = "###-###-####x#####";
o11 = *orders.order_num = items.order_num;
o12 = order_date,
       default = today;
i13 = items.stock_num;
i16 = items.manu_code, lookup manu_name = manufact.manu_name,
        joining *manufact.manu_code, upshift;
i18 = quantity, include = (1 to 100);
i19 = total_price;
o20 = po_num;
o21 = ship_date;
o22 = paid_date;
```

```
instructions
customer master of orders;
orders master of items;

end
```

Verify Joins

Like the INCLUDE attribute, *verify joins* let you compare the value you enter in PERFORM with a list of acceptable values. With the INCLUDE attribute, you enter the acceptable values in the form specification. With a verify join, the acceptable values are stored in one of the tables in your database.

To use a verify join, specify which column contains the acceptable values. PERFORM checks to see that the value you enter in a verify join field already exists in the column you specify.

To create a verify join, type an asterisk to the left of the name of the column in which the value should already exist. For example:

```
field-tag = col1 = *col2;
```

means that when the **col1** table is active, PERFORM rejects any input to the field that does not already exist in the **col2** column.

The **orderform** screen form in the **stores7** demonstration database uses a verify join to prevent users from assigning orders to nonexistent customers:

```
c1 = *customer.customer_num = orders.customer_num;
```

When **orders** is the active table, PERFORM rejects any customer number you attempt to enter that does not already exist as a **customer_num** value in the **customer** table.

Lookup Joins

Lookup joins let you extract data from a table and display it on a form based on the value a user enters in one of the fields on the form. For example, the **orderform** form uses a lookup join to display the name of a manufacturer as soon as the user enters the manufacturer code.

To use a lookup join, you must create one or more fields in the SCREEN section for the data you want to display. These lookup fields differ from other fields in that PERFORM never displays brackets around them, and thus users cannot modify the data that appears in them.

You do not create a separate entry for the lookup fields in the ATTRIBUTES section. Instead, include the instructions for the lookup join in the attribute list for the field on which you want PERFORM to base the lookup.

Format for Lookup Joins

The general format for a lookup join is as follows:

```
field-tag = col-name,
    LOOKUP lookup-tag = lookup-col JOINING join-col
```

- Here, field-tag is the field tag that appears in the SCREEN section for the field that PERFORM uses as the basis for the lookup. For example, in the **orderform** form, the manufacturer code is field **i16**.
- The *col-name* is the name of the column in which that data should be stored; again, this is the **manu_code** column in the **items** table.
- LOOKUP is a required keyword.
- The *lookup-tag* is the field tag you used for the lookup field in the SCREEN section. In the example, **manu_name** is the lookup tag.
- The equal sign is required.
- The *lookup-col* is the name of the column that contains the data you want to look up. Normally, *lookup-col* is in a different table than *col-name*. In the example, the **manu_name** column in the **manufact** table is the lookup column.
- JOINING is a required keyword.
- The *join-col* is the column that PERFORM joins to *col-name* to find the data you want to look up. The *join-col* is in the same table as the *lookup-col*. In the example, the **manu_code** column in the **manufact** table is the join column.

A Sample Lookup Join

The SCREEN section of the **orderform** form contains a lookup field with the field tag **manu_name**. The ATTRIBUTES section includes instructions for a lookup join in the attribute list for the **i16** field:

```
i16 = items.manu_code, lookup manu_name = manufact.manu_name,
      joining *manufact.manu_code, upshift;
```

When you enter a manufacturer code in the **i16** field, PERFORM joins the **manu_code** column in the **items** table to the **manu_code** column in the **manufact** table. Based on the join, PERFORM extracts the manufacturer name from the **manu_name** column in the row of the **manufact** table that matches the manufacturer code you entered.

The asterisk placed before *join-col* means that this is a verify join. PERFORM prevents you from entering a **manu_code** into the **items** table that does not already exist in the **manufact** table.

The INSTRUCTIONS Section

The INSTRUCTIONS section of the form specification lets you take advantage of more advanced PERFORM and FORMBUILD features. Here are some of the things you can do in the INSTRUCTIONS section:

- Create master-detail relationships that simplify multiple-table queries.
- Perform arithmetic calculations based on the values entered in a field and display the results on a form or store them in a table.
- Use different characters to delimit the fields on a form.
- Use composite joins to join tables based on the values contained in more than one column.
- Use instruction control blocks to indicate specific processing after a user enters or changes a value in a field.

These features are discussed in detail in the *INFORMIX-SQL Reference Manual*.

Calling C Functions from PERFORM

PERFORM usually can handle all your database screen form needs without modification. However, you might occasionally find it necessary to add a feature that is not available. For example, a C function call from PERFORM might check on the validity of data, record the data and time and the name of the person updating the records, or update the database. For more information on calling C functions from PERFORM, see the *INFORMIX-SQL Reference Manual*.

After you have written the file that contains your C functions, you must compile the file and link the necessary library functions to create a custom version of PERFORM. For information on how to create and compile a custom form, see the *INFORMIX-SQL Reference Manual*.

Chapter

7

Using SQL

In This Chapter 7-3

Examples Used in This Chapter 7-3

What Is SQL? 7-4
- Using SQL Statements 7-4
- Accessing SQL 7-5
- SQL Options 7-5
- The SQL SYNTAX Menu 7-6
- Learning to Use SQL 7-6

Differences Between SQL and PERFORM 7-7

How to Create a Table with SQL 7-8
- Entering an SQL Statement 7-8
- Using the SQL Editor 7-9
- Using the Use-Editor Option 7-11

Running an SQL Statement 7-12
- If There Are Errors 7-12
- Saving the Current Statements 7-14
  - Guidelines for Assigning Command Filenames 7-14

Using a Command File 7-15

Formatting SQL Statements 7-16

Sending Query Results to a Printer or a File 7-17

Displaying Table Information 7-18

In This Chapter

This chapter explains how to use SQL with the data in a database. It covers the following topics:

- Creating a table with SQL
- Using SQL as an interactive query language
- Granting database privileges to other users
- Using SQL to query a database and to store and sort query results
- Modifying the data in a table with SQL

This chapter also explains how to create tables and indexes with SQL and how to grant database privileges. In addition, it describes how to use SQL to work with the data in a database to retrieve, manipulate, add, remove, and modify rows and columns. Chapter 9, "Database Structure and Integrity," explains how to use the SQL menu system to modify the structure of databases and tables.

Examples Used in This Chapter

Many of the SQL statement examples used in this chapter are included with your INFORMIX-SQL software when you build the **stores7** demonstration database. When you list the contents of your working directory, the statements appear as **ex1.sql**, **ex2.sql**, **ex3.sql** files, and so on.

Example statements that are included with the database appear in the text with an example number.

What Is SQL?

SQL is an English-like interactive query language that you can use when working with databases. The SQL provided with INFORMIX-SQL is an enhanced version of the industry-standard query language developed by HCL. With SQL you can do the following:

- Create and drop tables and indexes
- Enter and delete data
- Query a database
- Select a different current database
- Display information about one or more tables
- Rename tables and columns
- Check and repair tables
- Grant and revoke database and table privileges

Using SQL Statements

To use SQL, select the Query-language option on the INFORMIX-SQL Main menu. The SQL menu appears, and you can enter one or more SQL statements. A statement is simply an instruction that tells SQL what you want to do. For example, to create a table, use the CREATE TABLE statement. To create an index, use the CREATE INDEX statement.

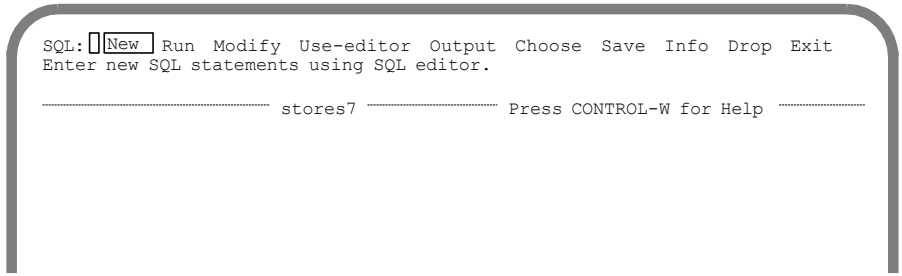
You can edit the statements as you enter them by using the SQL editor or a system editor. The SQL editor is discussed in detail in this chapter. The use of a system editor is noted briefly and discussed in greater detail in Chapter 6, "Creating Your Own Forms," and Chapter 8, "Creating and Printing Reports."

Just as there is a current database in INFORMIX-SQL, the SQL statements you are currently working with are called *current statements.*

INFORMIX-SQL keeps a copy of current statements. You can modify and run them as many times as you like without retyping them. You can also save them in a file so that you can use them again.

Accessing SQL

Select the Query-language option on the INFORMIX-SQL Main menu to access SQL. If you have not already selected a database to work with, INFORMIX-SQL first displays the CHOOSE DATABASE screen. After you select a database, INFORMIX-SQL displays the SQL menu.



SQL Options

The SQL menu displays the following 10 options:

New Enter new SQL statements. The statements you enter become the current statements, replacing any previous current statements.

Run Run the current SQL statements.

Modify Modify the current SQL statements.

Use-editor Use a system editor to enter or modify the current SQL statements.

Output Send the results of the current SQL statements to the screen or a file. You normally use this option only when you use SQL to query a database.

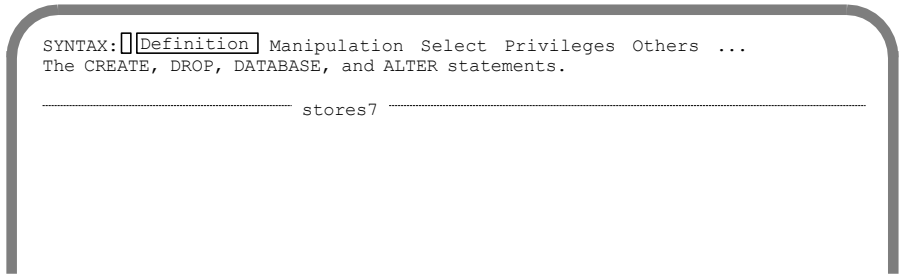
Choose Choose a file that contains SQL statements and make those statements the current statements.

Save Save the current SQL statements in a file so you can use them again later.

Info	Display information about the current database.
Drop	Drop an INFORMIX-SQL command file.
Exit	Return to the INFORMIX-SQL Main menu.

The SQL SYNTAX Menu

In addition to the usual HELP screens that are available with all INFORMIX-SQL menus and screens, the SQL New, Modify, and Use-editor options access a HELP menu that includes information about SQL statement syntax and usage. Select one of those options and then press CONTROL-W. The SYNTAX menu appears.



Learning to Use SQL

The *Informix Guide to SQL: Tutorial* provides information on how to use SQL. It covers the fundamental terms and ideas behind SQL. It tells you how to get data out of a database, how to modify a database by adding and changing data, how to write programs that make queries that modify data, and how to design a new database.

You can use the tutorial to learn how to write simple and advanced SELECT statements, optimize your queries, and build, implement, and tune a data model. The tutorial also covers security and views, and networks and distribution. The tutorial is packaged with companion manuals entitled *Informix Guide to SQL: Reference* and *Informix Guide to SQL: Syntax* that give you statement syntax and related reference information about SQL.

Differences Between SQL and PERFORM

Like PERFORM, SQL lets you enter and modify data and query the database. With SQL, however, you do not use a screen form. Instead, you enter statements. Here is a sample SQL statement:

```
select order_num,            {ex1}
    customer_num,
    order_date,
    paid_date
    from orders
    where order_date > "6/5/91"
    order by paid_date
```

The statement tells INFORMIX-SQL to select the data stored in the **order_num**, **customer_num**, **order_date**, and **paid_date** columns for every row in the **orders** table where the value in the **order_date** column is greater than 6/5/91. INFORMIX-SQL sorts the rows by the value in the **paid_date** column and displays this output on the screen. (The `{ex1}` characters are a comment, indicating that the statement is an example that is included in the demonstration database. INFORMIX-SQL ignores comments.)

After you enter the statement, you can run it and look at the results. The query results look like this:

```
order_num  customer_num  order_date  paid_date

     1006           112  09/19/1990
     1015           110  07/10/1990  08/31/1990
     1013           104  09/01/1990  10/10/1990
     1003           104  10/12/1990  11/04/1990
     1008           110  11/17/1990  12/21/1990
     1005           116  12/04/1990  12/30/1990

       6  row(s)  retrieved
```

INFORMIX-SQL displays only the information you request. It does not display a complete row, for example, unless you ask for a complete row. If you want only the last names of customers with orders totaling more than $500, that is exactly what you get—no first names, no customer numbers, no address information.

With this flexibility, you can specify which tables you want to query, join columns from different tables, perform mathematical and logical operations on data, and sort query results—all at the time that you enter the SQL query.

How to Create a Table with SQL

"Creating Tables with the Schema Editor" on page 1-20 describes how to create a table using the INFORMIX-SQL table schema editor. SQL gives you an alternative method for creating tables and databases.

You can use the CREATE TABLE statement to create a table. A CREATE TABLE statement contains instructions that define the characteristics of a table. You must enter a CREATE TABLE statement (or use the table schema editor) for each table in a database.

Each table in the **stores7** demonstration database was created with a different CREATE TABLE statement. Here is the statement that was used to create the **orders** table:

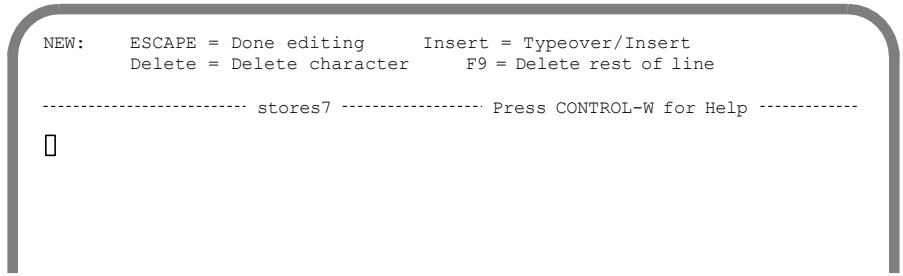
```
create table orders
(
    order_num        serial(1001),
    order_date       date,
    customer_num     integer,
    ship_instruct  char(40),
    backlog          char(1),
    po_num           char(10),
    ship_date        date,
    ship_weight      decimal(8,2),
    ship_charge      money(6),
    paid_date        date
);
```

This statement tells INFORMIX-SQL to create a table named **orders** and specifies names and data types for 10 columns. Column names appear on the left, while data types are indicated on the right.

Entering an SQL Statement

SQL statements always act on the current database. The CREATE TABLE statement creates a table in the current database. Before you create a table, be sure that the current database is the one in which you want the new table to appear.

When you are ready to enter the CREATE TABLE statement, select the New option on the SQL menu. The NEW Screen indicates you are using the SQL editor.



Using the SQL Editor

When the NEW Screen appears, you can start entering the CREATE TABLE statement using the SQL editor. The SQL editor lets you enter statements and edit them before you run them.

If you make a mistake, you can move the cursor with the Arrow keys and re-enter parts of the statement. You can also use the following editing keys listed at the top of the screen to perform special functions:

Escape — When you have finished editing, press ESCAPE to exit the SQL editor.

Insert Mode — CONTROL-A switches you back and forth between insert and typeover mode. While in insert mode, the text beneath the cursor shifts to the right as you enter new characters. While in typeover mode, characters you enter replace the text beneath the cursor. You are automatically placed in typeover mode when you call up the SQL editor.

Redraw the Screen CONTROL-R redisplays the screen. You can use CONTROL-R if, while entering SQL statements, you received electronic mail or some other message that makes it difficult to read what you have entered.

Delete a Character DELETE deletes the character that appears beneath the cursor.

Delete Rest of Line The F9 key deletes characters from the current cursor position to the end of the line.

When you use the SQL editor, you can enter as many lines of text as you want. You are limited only by the memory constraints of your system, not by the size of the screen.

The SQL editor does not display more than 80 characters on a line. If you insert characters so the line exceeds 80 characters, INFORMIX-SQL displays the percent sign (%) at the end of the line to indicate that you have entered characters beyond the eighth column. Although INFORMIX-SQL will read and compile these characters, it will be difficult for you to work with text that is not visible. It is a good idea to insert a RETURN somewhere in the first 80 characters of each line so that the full text appears on the screen.

If you insert lines so that the total number of lines is greater than the screen can hold, the SQL editor scrolls down the page with the extra lines and displays the beginning and ending line numbers of the current page on the fourth line down from the top of the screen.

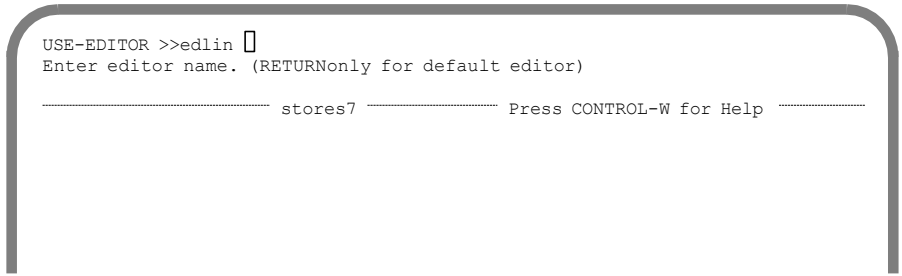
```
NEW:    ESCAPE = Done editing      Insert = Typeover/Insert
        Delete = Delete character      F9 = Delete rest of line

-- 3 to 20 of 20 ----- stores7 ----------- Press CONTROL-W for Help --------

```

If you prefer working with the system editor, you can always press ESCAPE and then select the Use-editor option on the SQL menu.

Using the Use-Editor Option

For a long series of statements, you might want to use a system editor. When you know that the statements you enter are long, you can choose the Use-editor option right away, instead of entering some text with the New option and then changing to the Use-editor option.

If this time is the first time in this session that you use the Use-editor option, INFORMIX-SQL might display the USE-EDITOR screen.

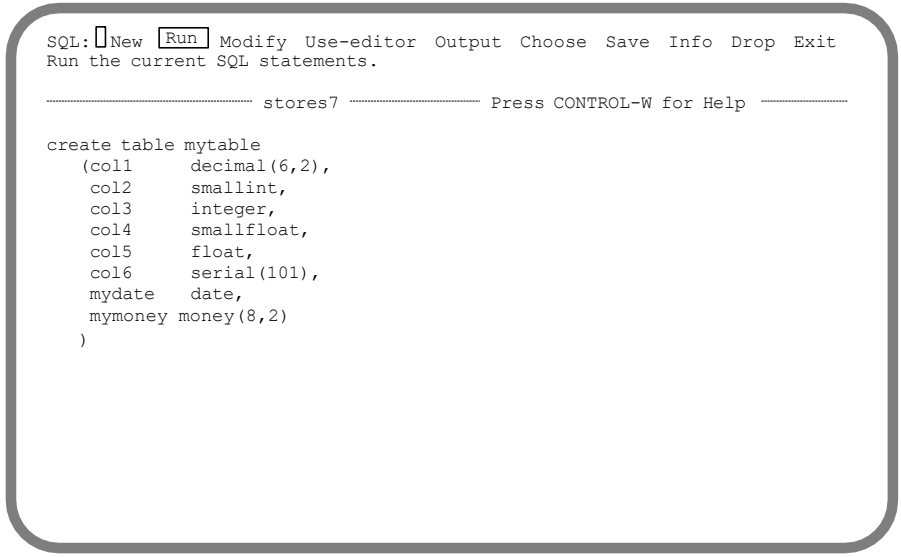


On this screen, **vi** is the default editor. You might have a different default editor on your system. You can specify your choice of editor with the **DBEDIT** environment variable. (For information about environment variables, see the *Informix Guide to SQL: Reference*.) If you have already specified an editor with this screen or if you have specified an editor with **DBEDIT**, INFORMIX-SQL calls up the editor immediately and does not display the USE-EDITOR screen.

When the USE-EDITOR screen appears, type the name of the editor you want to use or press RETURN to select the default editor. INFORMIX-SQL calls the editor you specified. Edit the text and then save the file, following the usual rules for the editor you select. The SQL menu is redisplayed.

Running an SQL Statement

After you enter the CREATE TABLE statement, press ESCAPE to tell INFORMIX-SQL you are finished. When you press ESCAPE, you will see the SQL menu again. The statement you entered will appear in the bottom half of the screen.



To run the statement and create the table, select Run on the SQL menu. The Run option is already highlighted, so just press RETURN.

If There Are Errors

When you select the Run option, INFORMIX-SQL first checks the statements to make sure they conform to the SQL grammar and syntax rules. If your statements contain no mistakes, INFORMIX-SQL processes the statements and displays the results on the screen.

If you make any errors in a statement, INFORMIX-SQL does not process the statement. Instead, it displays the statements you entered and a message that describes the error. For example, if there is a mistake (perhaps a misspelled keyword), INFORMIX-SQL displays an error message on the bottom of the screen and highlights the Modify option on the SQL menu.

```
SQL:  New  Run  Modify  Use-editor  Output  Choose  Save  Info  Drop  Exit
Modify the current SQL statements using the SQL editor.

                        stores7                  Press CONTROL-W for Help

create table mytable
   (col1      decimal(6,2),
    col2      smallint,
    col3      integer,
    col4      smallfloat,
    col5      float,
    col6      serial(101),
    mydate    date,
    mymoney   money(8,2)




201: A syntax error has occurred.
```

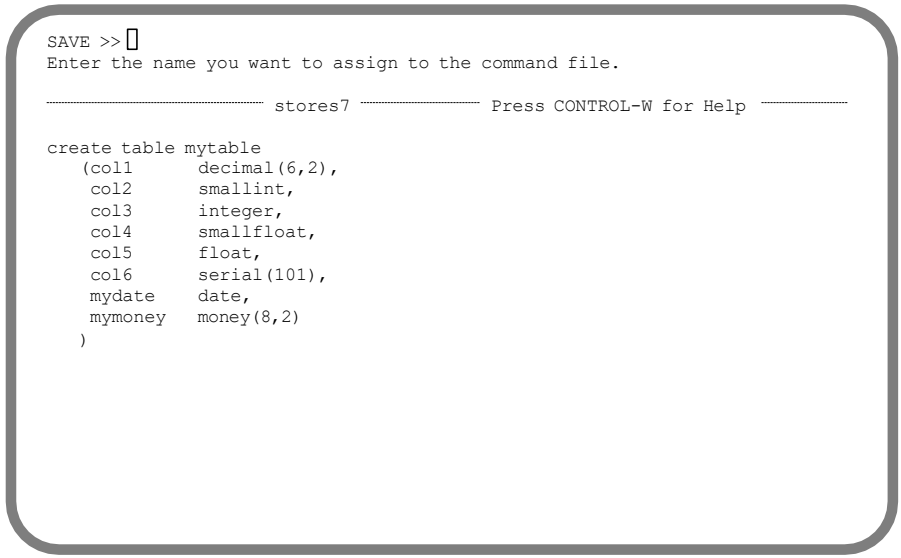
When you press RETURN to select Modify, INFORMIX-SQL calls the SQL editor and positions the cursor on the first error. You can correct the mistake using the SQL editor (press ESCAPE when you have finished editing the statement), or you can select the Use-editor option and edit the statement using your system editor. After you return to the SQL menu, you can run the statement again.

Tip: *The text of all error messages and suggestions for corrections is included in Informix Error Messages in Answers OnLine.*

Saving the Current Statements

When you have entered and run the CREATE TABLE statement successfully, you can save it in a *command file*. A command file is a system file that contains one or more SQL statements.

To save the current statements in a command file, select the Save option on the SQL menu. INFORMIX-SQL displays the SAVE screen and prompts you to enter a name for the command file.



Guidelines for Assigning Command Filenames

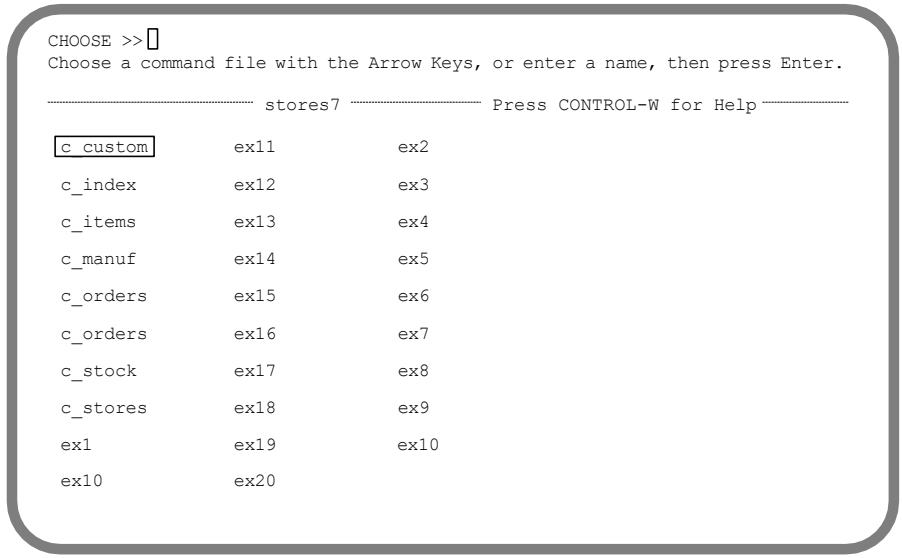
Command filenames can be up to 10 characters long. The first character must be a letter, but you can use letters, numbers, and underscores (_) for the rest of the name. You can use uppercase and lowercase letters. UNIX systems are case sensitive; **ords1** is not the same as **Ords1** or **ORDS1**.

Type a name for the command file that will hold your statements and press RETURN.

INFORMIX-SQL stores the statements in a command file, using the name you gave and the extension **.sql.** For example, a statement you called **ex1** is stored in the command file **ex1.sql**. You can inspect the statements at any time with the Choose option on the SQL menu.

Using a Command File

When you save statements in a command file, you can use them again at any time. To do so, select the Choose option on the SQL menu. The CHOOSE screen then appears and displays a list of the names of command files to which you have access. In addition to the command files shown in the following screen, you might see other command files that have been added to provide further practice once you are familiar with the demonstration database.



To select a command file, use the Arrow keys to move the highlight over its name, then press RETURN. You also can type the name of the file you want and press RETURN.

After you select a command file, the SQL menu appears. The statements contained in the command file are now the current statements, and they appear on the screen. You can modify or run these statements.

Formatting SQL Statements

You can use almost any format you like for SQL statements. You can put every word on a different line or type as much as you can fit on each line. This query:

```
select                    {ex4}
    customer_num,
    lname,
    city,
    phone
    from customer
```

is the same as this query:

```
select customer_num, lname, city, phone
    from customer                              {ex4}
```

The sample statements in this chapter are formatted so you can read them easily. You can include comments in your SQL statements. Just enclose your comments in braces ({ }) like this:

```
select                 {beginning  of  statement}
    customer_num,             {list  of  columns}
    lname,
    city,
    phone
    from customer           {table  in  stores7  db}
```

You can also indicate comments by preceding the comment with a pound sign (#) or double dashes (--). Comments are for your use only; INFORMIX-SQL does not process comments.

You can include several SQL statements at one time, but you must separate them with semicolons, as in the following example:

```
select order_num from orders;       {ex5}
select customer_num, lname,
    city, phone
from customer
```

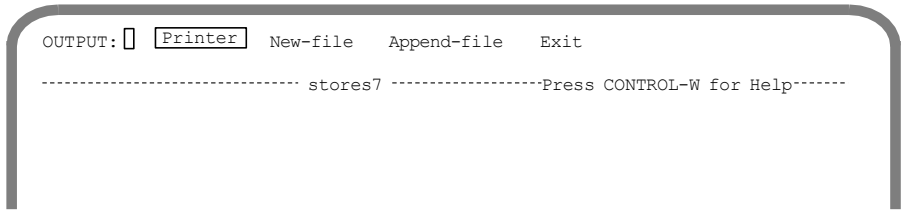
Semicolons are statement separators, not terminators. You do not have to type a semicolon at the end of your last statement. INFORMIX-SQL sequentially processes your statements and displays the output.

Sending Query Results to a Printer or a File

The output from an SQL statement normally appears on the screen. The Output option on the SQL menu lets you send query results to the printer or write them to a file. This section discusses these operations.

When you use the Output option, the query is printed or stored just as it appears on the screen. (Format control and other sophisticated report features available for use with query results are explained in Chapter 8, "Creating and Printing Reports.")

The OUTPUT menu displays as follows.



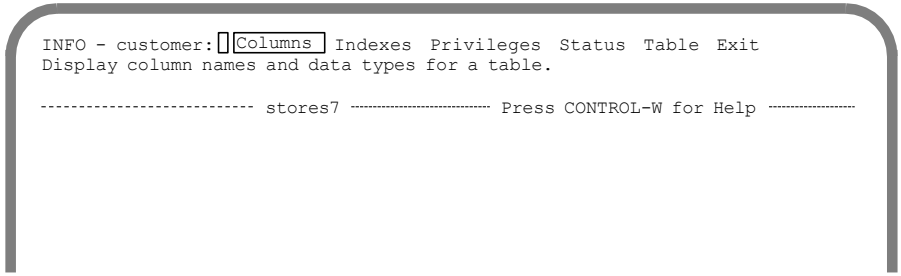
To send your query results directly to the printer, select the **Printer** option on the OUTPUT menu. INFORMIX-SQL sends the query results directly to your printer and displays a message on the bottom of the screen telling how many rows it retrieved. The query results do not appear on the screen.

INFORMIX-SQL lets you write query results to a new file or append the results to an existing file. You can use your operating-system programs to edit, copy, rename, or delete the file.

Displaying Table Information

Use the Info option on the SQL menu to display information about the columns, indexes, privileges, and status of a table.

When you select the Info option, INFORMIX-SQL displays the INFO FOR TABLE screen with a list of tables. Type or highlight the name of the table that you want and press RETURN. The INFO menu then appears.



The **Info** menu has the following six options:

Columns Lists all the columns in the table and the data type of each column.

Indexes Lists the indexes for the table, the owner and type of each index, whether it is a cluster index, and the columns to which each index applies.

Privileges Lists the users who have access privileges for the table and whether they have the table-level privileges Select, Update, Insert, Delete, Index, and Alter. (Unless your login is listed separately, you have the privileges given for public, which is a general category for all users.)

Status Lists the table name, the table owner, the size of the row (in number of characters), the number of rows in the table (as of the last UPDATE STATISTICS statement), the number of columns in a row, the date the table was created, and the name of the audit trail file (if there is one).

Table Lets you select a new table for examination.

Exit Returns to the SQL menu.

The Info option is also available from the TABLE menu. You can use this option to gather information about external tables.

Chapter

8

Creating and Printing Reports

In This Chapter 8-3

What Is a Report Writer? 8-3

Steps in Creating a Report 8-5
- Using the REPORT Menu 8-6
- A Sample Specification and Report 8-7
- Sections of a Report Specification 8-9
  - The DATABASE Section 8-9
  - The SELECT Section 8-9
  - The FORMAT Section 8-10

Creating a Report Specification 8-12
- Generating a Default Report Specification 8-12
- Naming the Report 8-13
- Compiling the Default Report Specification 8-14

Running the Report 8-15

Modifying a Report 8-18
- Saving the Report Specification 8-20

Compiling a Report Specification 8-21
- If the Compilation Is Unsuccessful 8-21

What a Report Specification Contains 8-23
- Listing the Instructions 8-24
- Including Comments 8-24

Selecting the Data for a Report 8-25
- Formatting the Report 8-26
- Selecting All Rows and Columns in a Table 8-26
- Selecting Data from More than One Table 8-27

Reading the Data for a Report....8-27

Basic Report Formatting....8-28
The OUTPUT Section....8-29
The FORMAT Section....8-29
FORMAT for a Default Report....8-30
FORMAT for a Custom Report....8-30
Creating Page Headers and Trailers....8-31
Where Headers and Trailers Appear....8-31
Printing a Report Heading....8-32
Printing Column Headings....8-33
Numbering Pages Automatically....8-34
Printing Rows of Data....8-35
Organizing the Data....8-37
Grouping Data....8-37
The Group Control Blocks....8-39

Using Calculations in a Report....8-40
Arithmetic Operators....8-41
Example of Arithmetic Operations....8-41
Using Parentheses....8-42
Controlling the Appearance of Numbers....8-42
Using Fill Characters....8-43
The Fill Characters....8-44
Adding a Dollar Sign....8-44
Making the Dollar Sign Float....8-45
Math Functions....8-45
Totaling a Column....8-46
Counting Rows in a Report....8-47
Combining Arithmetic Operators and Math Functions....8-49
Group Functions....8-49

Displaying a Report....8-52
Printing the Report on a Printer....8-52
Writing the Report to a File....8-53
Piping the Report to Another Program....8-53

Calling C Functions from ACE....8-53

In This Chapter

This chapter explains how to create, display, and print reports with INFORMIX-SQL. The following topics are discussed:

- Using a report writer
- Creating and compiling a report specification
- Including selected database information in a report
- Including data from an input file in a report
- Grouping and formatting the information
- Adding titles, headings, and other text to customize the report
- Displaying the report on your terminal or sending it to the printer

Read this chapter to learn how to create your own INFORMIX-SQL reports. If you intend to use only reports that someone else creates, you do not need to read this entire chapter. Instead, go to "Running the Report" on page 8-15.

For more information about all the features available for creating and running reports, see the *INFORMIX-SQL Reference Manual*.

What Is a Report Writer?

A report consists of database information arranged and formatted according to your instructions and displayed on the screen, printed on paper, or stored in a file for future use. You can select all or some of the information stored in a database and display it in a report. You also can retrieve data from an ASCII input file and display that information in a report.

INFORMIX-SQL includes a *report writer* that you use to retrieve and format the information you want to include in your reports. The report writer is called ACE.

The ACE report writer retrieves data from a database or from a file, organizes and formats it according to your instructions, adds the headings or column titles you specify, performs calculations on number columns, and then displays the report.

You can combine information from several tables in a database to create whatever kind of report you need. Some of the kinds of reports ACE can produce are:

- Mailing labels
- Form letters
- Payroll summaries
- Medical histories
- Accounting records
- Inventory lists
- Telephone directories
- Project schedules
- Sales and marketing reports
- Payroll checks

Example report specifications are included in "A Sample Specification and Report" on page 8-7. They should serve as useful models for creating similar reports for your databases.

Steps in Creating a Report

To create and display a report in INFORMIX-SQL, you need to complete the following three tasks:

1. Create a *report specification*.

   A report specification is a list of instructions that tells INFORMIX-SQL what information to include in the report, how to organize and group it, what headings or titles to add, what calculations to perform, and what margins and other formatting elements to use.

2. Compile the report specification.

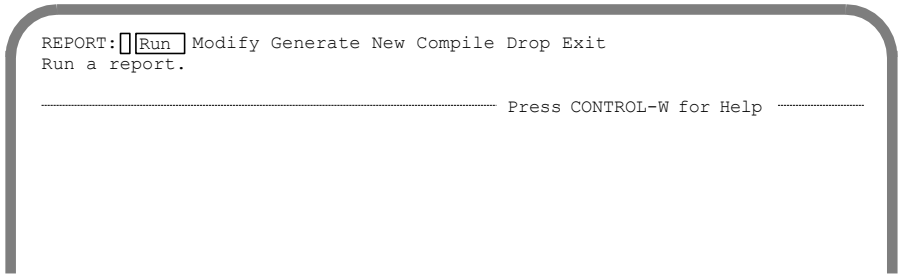
   During compilation, INFORMIX-SQL checks if the specification is correct and complete. If there are any errors, you must correct them and recompile the specification.

3. Run the compiled report.

   When you run the report, INFORMIX-SQL reads the database or input file, formats the information, performs the requested operations, and either displays the report on your screen, prints it on the printer, or sends the report to a file.

Using the REPORT Menu

The Report option on the Main menu provides menu options you can use to create, modify, compile, and run reports.



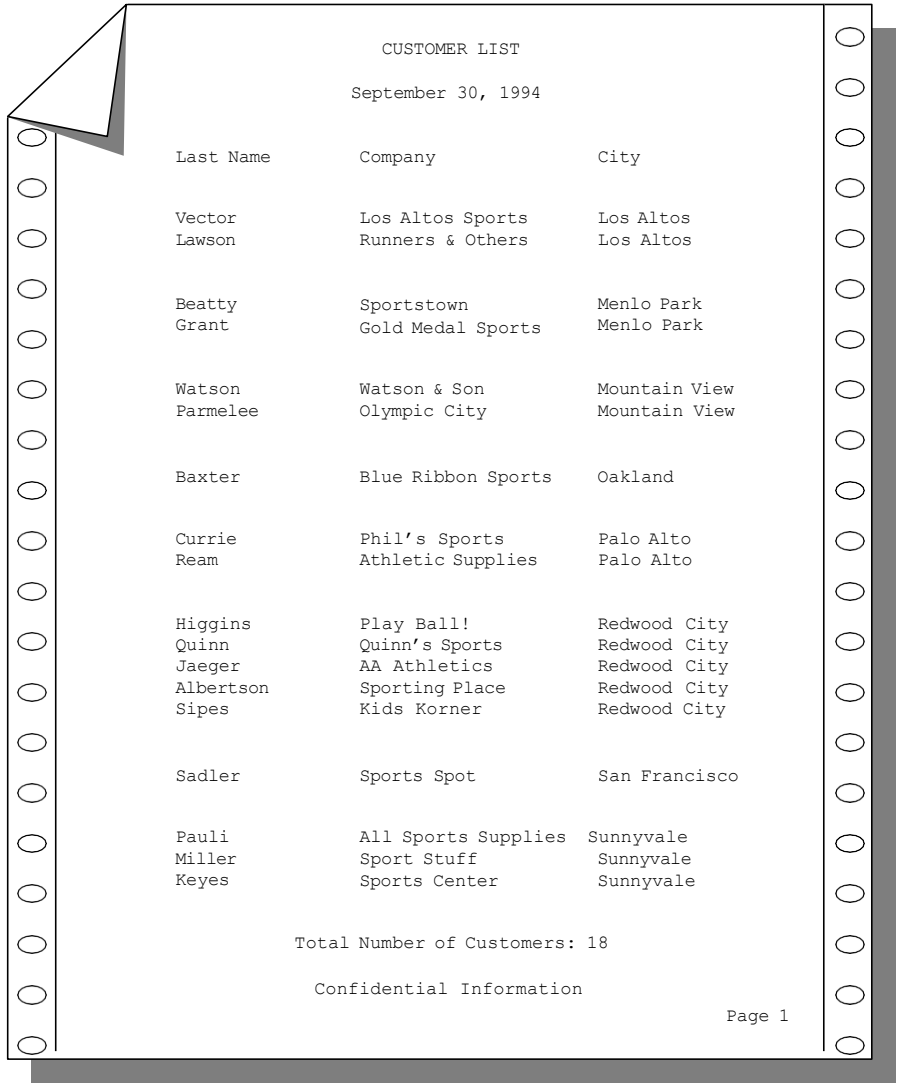
The REPORT menu displays the following seven options:

Run	Displays a report.
Modify	Changes an existing report specification.
Generate	Generates a default report specification based on a database table.
New	Creates a new report specification.
Compile	Compiles a report specification.
Drop	Drops a report.
Exit	Returns to the INFORMIX-SQL Main menu.

A Sample Specification and Report

Figure 8-1 shows a sample report you can produce from the **stores7** demonstration database.

Figure 8-1
A Sample Report



CUSTOMER LIST

September 30, 1994

Last Name	Company	City
Vector	Los Altos Sports	Los Altos
Lawson	Runners & Others	Los Altos
Beatty	Sportstown	Menlo Park
Grant	Gold Medal Sports	Menlo Park
Watson	Watson & Son	Mountain View
Parmelee	Olympic City	Mountain View
Baxter	Blue Ribbon Sports	Oakland
Currie	Phil's Sports	Palo Alto
Ream	Athletic Supplies	Palo Alto
Higgins	Play Ball!	Redwood City
Quinn	Quinn's Sports	Redwood City
Jaeger	AA Athletics	Redwood City
Albertson	Sporting Place	Redwood City
Sipes	Kids Korner	Redwood City
Sadler	Sports Spot	San Francisco
Pauli	All Sports Supplies	Sunnyvale
Miller	Sport Stuff	Sunnyvale
Keyes	Sports Center	Sunnyvale

Total Number of Customers: 18

Confidential Information

Page 1

This report lists the last name, company, and city for all customers. It groups customers by city and calculates the total number of customers. A notation that this is confidential information and the current page number are printed at the bottom of each page.

Figure 8-2 shows the report specification used to generate the sample report.

Figure 8-2
A Sample Report Specification

```
database
    stores7
end

select
    lname, company, city
    from customer
    order by city
end

format

page  header
    print column 20, "CUSTOMER LIST"
    skip 1 line
    print  column  18,  "September  30,  1994"
    skip  2  lines
    print "Last Name",
    column  18,  "Company",
    column  40,  "City"
    skip  2  lines

on every row
    print lname,
    column  18,  company,
    column  40,  city

after group of city
    skip 2 lines

on last row
    skip 2 lines
    print column 12, "Total Number of Customers:",
    column 40, count using "##"

page  trailer
    print "Confidential  Information",
    column  55,  pageno  using  "Page  ##"

end
```

Sections of a Report Specification

The sample report specification in Figure 8-2 contains three sections: DATABASE, SELECT, and FORMAT. All report specifications must contain at least three sections: a DATABASE section, a SELECT or READ section, and a FORMAT section. See "Reading the Data for a Report" on page 8-27 for information on the READ section.

The following sections explain what each section in the sample report specification contains.

The DATABASE Section

All report specifications must begin with a DATABASE section. The DATABASE section tells INFORMIX-SQL which database contains the information you want to use in the report.

```
database
   stores7
end
```

In this sample specification, the information for the report is contained in the **stores7** demonstration database.

The SELECT Section

The SELECT section of a report specification contains one or more SELECT statements. The SELECT statements work exactly the same way in report specifications as they do in SQL. (A report specification must always contain either a SELECT section or a READ section.)

Use a SELECT statement to specify which rows and columns contain the information that you want to include in the report.

```
select
     lname, company, city
     from customer
     order by city
end
```

In this example, the SELECT statement contains three clauses:

SELECT	Selects the information in the last name, company, and city columns.
FROM	Indicates that the columns are contained in the **customer** table.
ORDER BY	Orders the rows (in ascending order) by city.

In earlier releases, you were limited to using the ORDER BY clause in only the last SELECT statement. You can include the ORDER BY clause in any SELECT statement.

For a detailed explanation of the SELECT statement, refer to the *Informix Guide to SQL: Syntax*.

The FORMAT Section

The FORMAT section controls the appearance of a report. You can add report titles and column headings, group information in a specific way, perform calculations, and add a footer at the bottom of the report.

```
format

page header
   print column 20, "CUSTOMER LIST"
   skip 1 line
   print column 18, "September 30, 1994"
   skip 2 lines
   print "Last Name",
   column 18, "Company",
   column 40, "City"
   skip 2 lines

on every row
   print lname,
   column 18, company,
   column 40, city

after group of city
   skip 2 lines

on last row
   skip 2 lines
   print column 12, "Total Number of Customers:",
   column 40, count using "##"
```

```
page trailer
   print "Confidential Information",
   column 55, pageno using "Page ##"

end
```

In this example, the FORMAT section contains a list of instructions in the following five control blocks:

PAGE HEADER Prints a report title (CUSTOMER LIST), skips one line, and prints a date (September 30, 1994); skips two lines and prints column headings (Last Name, Company, and City) in the designated column positions; skips two lines.

ON EVERY ROW Prints the contents of the last name, company, and city columns of every row in the designated column positions.

AFTER GROUP OF Skips two lines after printing each group of rows clustered by city.

ON LAST ROW After selecting the last row, skips two lines and prints a phrase (Total Number of Customers:) in the designated column position; calculates and displays the total number of customers.

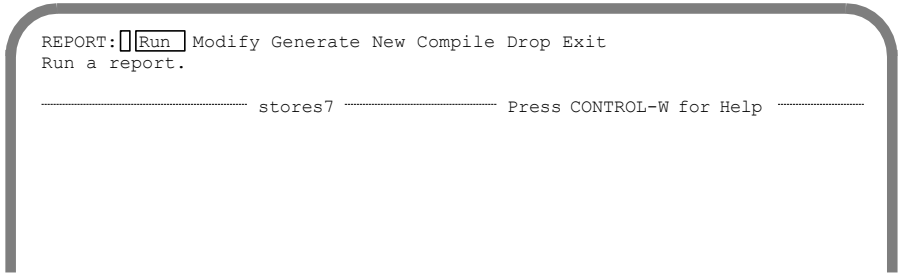
PAGE TRAILER At the bottom of each page, prints the words Confidential Information and the current page number.

Later in this chapter, you will learn how to format control blocks.

Creating a Report Specification

Before you can create your report, you have to tell INFORMIX-SQL which database you will be working with and provide the name of the report you want to create.

To begin creating a report specification, select the Report option from the Main menu. INFORMIX-SQL displays the REPORT menu.



The REPORT menu provides two ways to create a report specification:

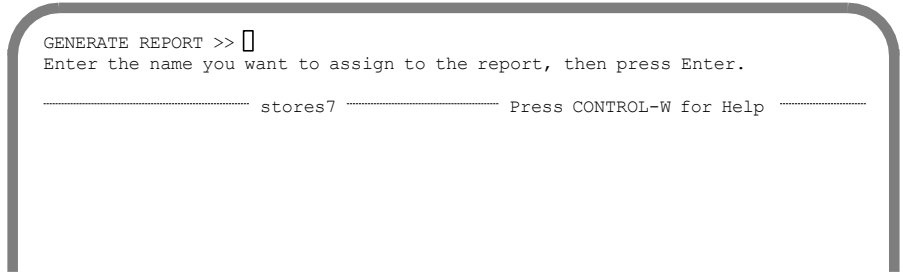
- You can select the Generate option and generate a default report specification.
- You can select the New option and use your system text editor to create a new report specification.

The next few pages describe the process of generating a default report specification. The use of the New option on the Report menu is covered in the *INFORMIX-SQL Reference Manual*.

Generating a Default Report Specification

To create a default report specification, select the Generate option. If you have not yet selected the database with which you want to work, INFORMIX-SQL displays the CHOOSE DATABASE screen. Type or highlight the name of the database you want to work with (select **stores7** so you can work with the demonstration database) and press RETURN.

If you already have selected a database, INFORMIX-SQL displays the GENERATE REPORT screen.



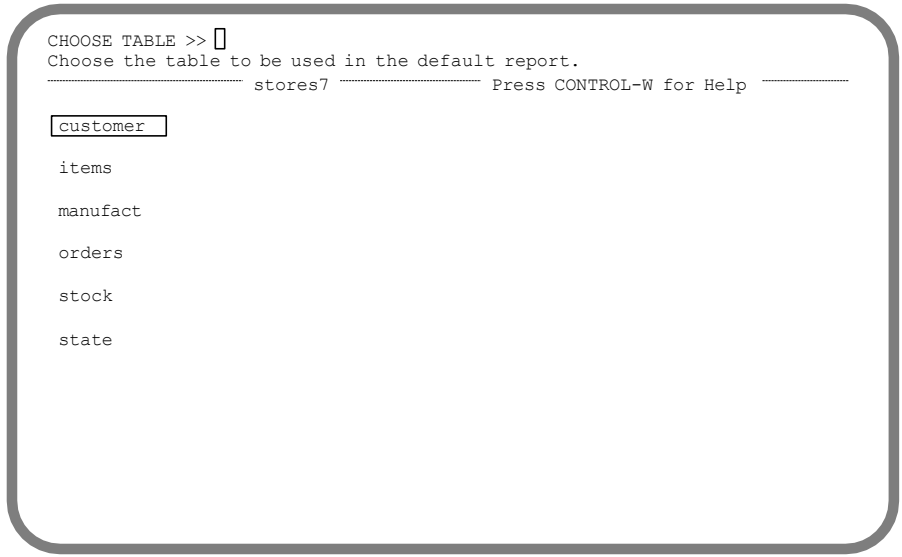
After you select the Generate option (and choose a database), INFORMIX-SQL prompts you to enter the name you want to assign to the new report specification.

Naming the Report

You can assign any name you want to the report specification, so long as it meets the following conditions:

- The name can be up to 10 characters long.
- The report specification name must begin with a letter and can consist of any combination of letters, numbers, and underscores (_). You can use uppercase and lowercase letters.

After you enter the name of the report specification, INFORMIX-SQL displays the CHOOSE TABLE screen. The following example lists some of the available tables.



Choose the table that INFORMIX-SQL will use for the default report. The default report will display information drawn from this table. Select a table from those listed on the screen (select **customer** to work with the **customer** table that is included in the **stores7** demonstration database).

Compiling the Default Report Specification

You must compile a report specification before INFORMIX-SQL can run the report. Compiling lets INFORMIX-SQL convert the specification into a special format that ACE uses.

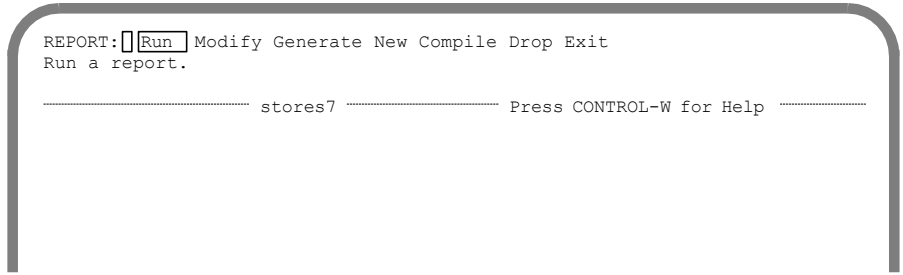
After you select a table for the default report specification, INFORMIX-SQL compiles the specification, saves the specification file, and returns you to the REPORT menu. ACE can now use the compiled report specification.

The default report specification does not take advantage of many of the powerful features available with ACE. For example, the default report includes only one table, the data display might be unattractive, and the report does not perform any data manipulation. You probably will want to make changes to the report specification file before you use it. The remaining sections of this chapter describe ways in which you can tailor a report to meet your specific needs.

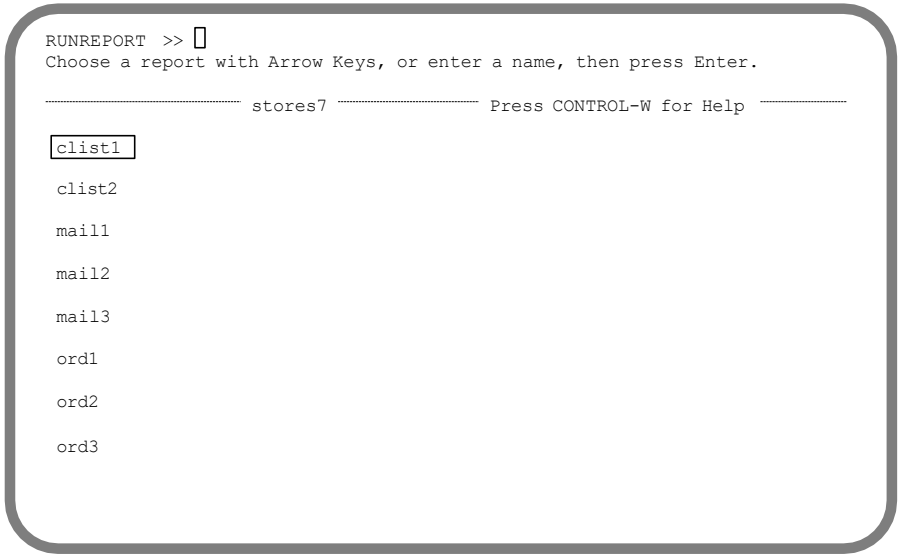
Running the Report

When you run a report, INFORMIX-SQL reads the compiled report specification, locates the specified database or input file, and selects and formats the information according to the instructions in the report specification.

To run a report, select the Run option on the REPORT menu.



INFORMIX-SQL displays the RUN REPORT screen that lists available reports.

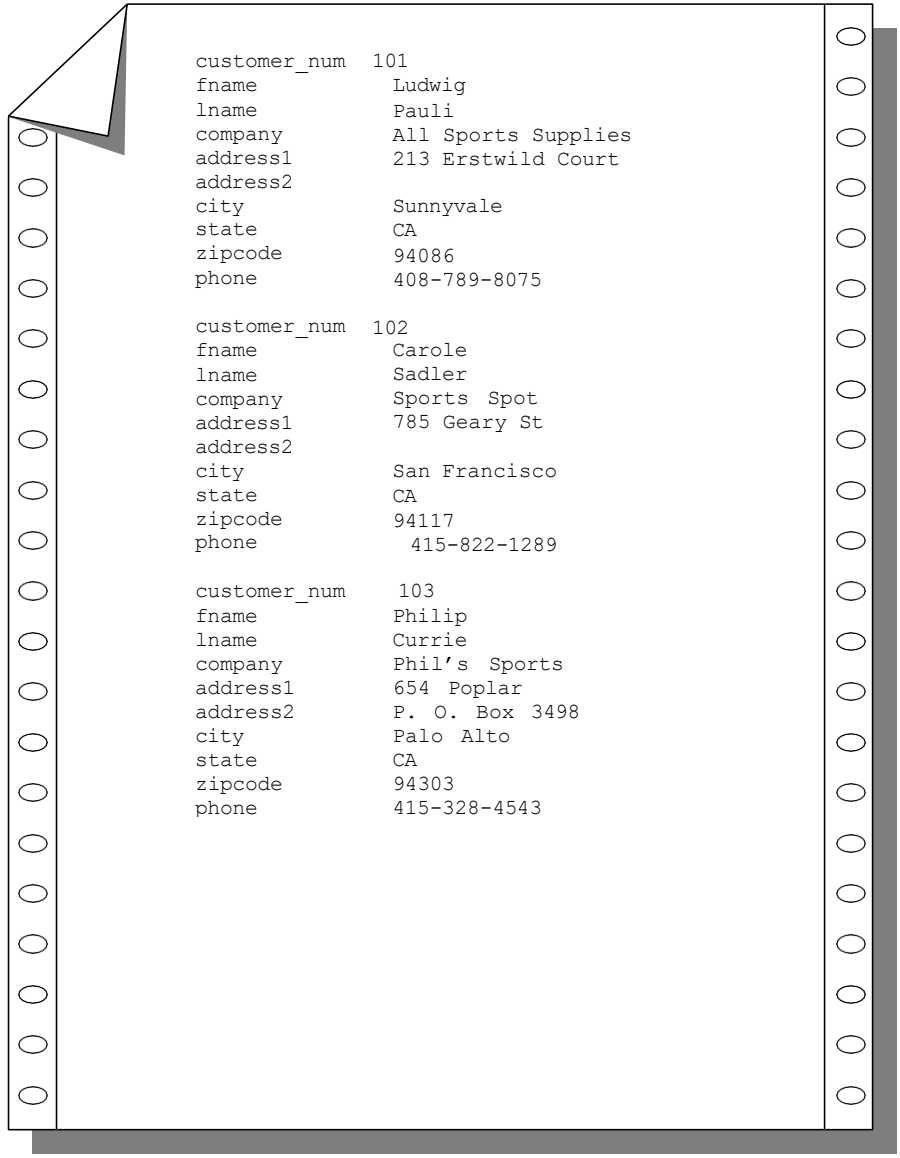


(Reports in addition to the ones listed in this example are provided for further practice.)

Type or highlight the name of the report you want to run and press RETURN. INFORMIX-SQL runs the selected report.

Depending on the instructions that the report specification contains, the finished report is either displayed on your screen, written to a file, or printed on a printer. By default, INFORMIX-SQL displays the results on the screen. Subsequent sections of this chapter describe how to store a report in a file or print it on the system printer.

The following example shows the first few lines of the default report based on the **customer** table.

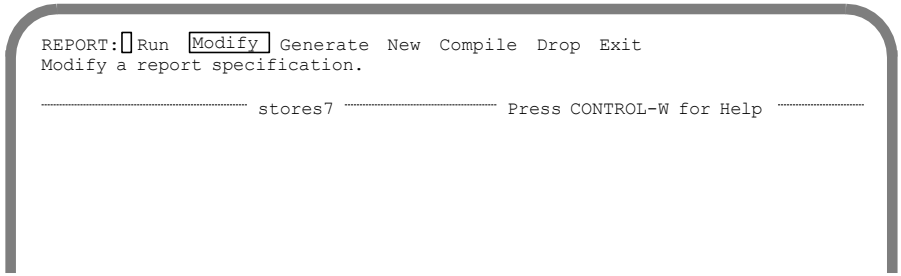


```
customer_num  101
fname          Ludwig
lname          Pauli
company        All Sports Supplies
address1       213 Erstwild Court
address2
city           Sunnyvale
state          CA
zipcode        94086
phone          408-789-8075

customer_num  102
fname          Carole
lname          Sadler
company        Sports  Spot
address1       785 Geary St
address2
city           San Francisco
state          CA
zipcode        94117
phone           415-822-1289

customer_num   103
fname          Philip
lname          Currie
company        Phil's  Sports
address1       654  Poplar
address2       P. O. Box 3498
city           Palo Alto
state          CA
zipcode        94303
phone          415-328-4543
```

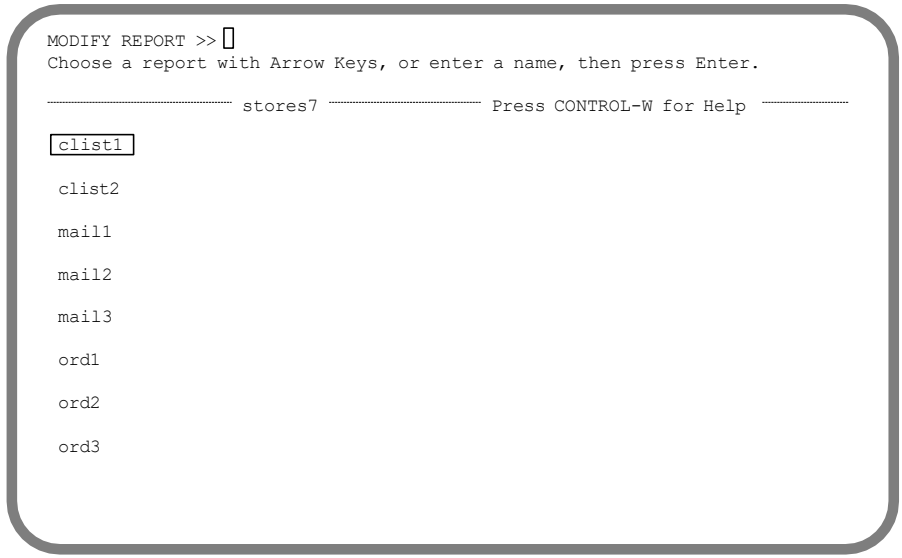
Modifying a Report

To make changes to an existing report, you must modify the report specification and then recompile it.

To change a report specification, select the Modify option on the REPORT menu.



INFORMIX-SQL displays the MODIFY REPORT screen with a list of available report specifications (reports in addition to those listed here are provided for further practice).



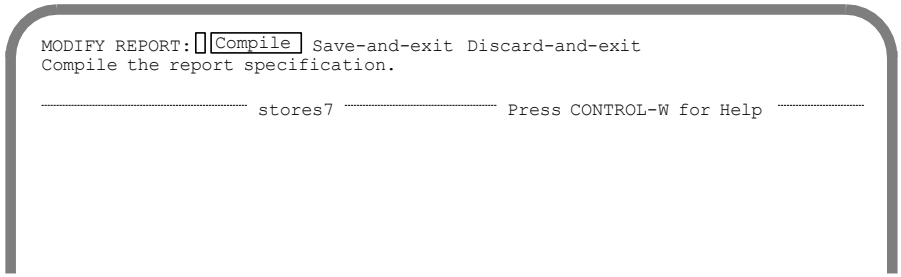
Type or highlight the name of the report specification you want to modify and press RETURN.

After you select the specification, INFORMIX-SQL prompts you for the name of your editor (unless you have specified an editor previously in this session or set the **DBEDIT** environment variable as described in the *Informix Guide to SQL: Reference*). INFORMIX-SQL then activates your system text editor and displays the report specification file. You can make any changes you like to the specification with your system editor.

When you finish entering changes to the specification, save the file according to the procedures you normally follow to save files with your text editor.

Saving the Report Specification

After you save the report specification file, the MODIFY REPORT menu appears.



The MODIFY REPORT menu has the following three options:

Compile	Compiles the report specification and returns to the MODIFY REPORT menu if the compilation is successful. Once the report specification is compiled, you can save it and run it.
Save-and-exit	Saves the report specification without compiling it and returns to the REPORT menu. You can edit and/or compile the report specification file at a later time.
Discard-and-exit	Discards the modifications just made to the report specification file and returns to the REPORT menu.

You probably will want to compile the report specification immediately. To do so, select the Compile option.

Compiling a Report Specification

When you compile a report specification, INFORMIX-SQL reads each line of the specification to see if you entered everything completely and correctly. If the specification is correct, INFORMIX-SQL proceeds with the compilation.

When the compilation is completed, INFORMIX-SQL displays a message to that effect and the MODIFY REPORT menu reappears. You can then select the Save-and-exit option and run the report.

If the Compilation Is Unsuccessful

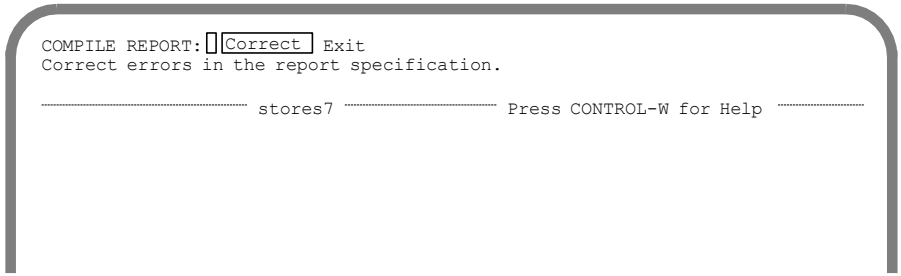
If the report specification contains instructions that INFORMIX-SQL cannot read or process, it stops compiling and displays an error message on the screen. You must correct the errors before INFORMIX-SQL can successfully compile the specification.

The following example shows a portion of a report specification that contains an error:

```
database
     stores7
                    {end  keyword  is  missing}

select
      lname,  company,  city
             .
             .
             .
```

In this example, the DATABASE section is missing the END keyword.

After attempting to compile a report specification that contains an error, INFORMIX-SQL displays a message to that effect and displays the COMPILE REPORT menu.



To correct an error in the specification, select the Correct option on the COMPILE REPORT menu. INFORMIX-SQL calls your system editor and displays the report specification on your screen, with errors flagged by a caret (^), and an error message.

Tip: *The text of all error messages and suggestions for corrections is included in Informix Error Messages in Answers Online.*

The following example shows how the specification file might look after you attempted to compile it:

```
database
     stores7
                        {end keyword is missing}
select
#^
#
#
A grammatical error has been found on line 4, character 1.
# The construct is not understandable in its context.
# See error number -8018.
#
     lname, company, city
```

In this example, you need to correct the report specification by adding the keyword END to the DATABASE section.

Tip: *In the example, ACE flagged the word "select" as the point at which it encountered the error. In fact, it is the missing END keyword that caused the error. This demonstrates a common occurrence with ACE: an error often is located in the control block immediately preceding the flag. When a grammatical error is flagged, look at the block preceding the error flag and check for a missing or misspelled keyword, or a missing or misplaced punctuation mark.*

After you correct the error, save the specification file in the usual way and exit from your text editor. INFORMIX-SQL removes the error message from the specification file when you save it. Choose the Compile option again to recompile the specification.

When the report compiles successfully, INFORMIX-SQL displays the MODIFY REPORT menu. Choose the Save-and-exit option. INFORMIX-SQL saves the compiled report specification and displays the REPORT menu.

What a Report Specification Contains

A report specification is divided into sections. Each section begins with the section name (such as DATABASE, SELECT, or FORMAT), followed by control blocks (such as ON EVERY ROW or ON LAST ROW) and statements (such as PRINT or SKIP), and concludes with the keyword END. The sections must appear in the sequence that follows, and a report must include the sections marked as required:

- DATABASE (required)
- DEFINE
- INPUT
- OUTPUT
- SELECT (SELECT or READ required)
- READ (READ or SELECT required)
- FORMAT (required)

The control blocks contain statements, and most of the statements can contain clauses. For a description of an ACE report specification, refer to the *INFORMIX-SQL Reference Manual*.

Listing the Instructions

Within a section, you can list the control blocks and statements in any style or format you like, as long as you enter them correctly. INFORMIX-SQL ignores extra spaces and ENTER commands in the specification.

Including Comments

You can include comments in your report specification. That way, if you need to modify the report or if someone else reads the specification, comments explain what the specification does.

Include comments within a set of braces ({ }) or use a double dash (--) prefix. INFORMIX-SQL ignores text enclosed in braces or preceded by a double dash. The following example report specification includes comments:

```
{
This report lists customers by company and
city and groups by city
}
database                          {select stores7 database}
   stores7
end

output                              {set left margin to 10}
   left margin 10
end

select                         {select report information}
   lname, company, city
   from customer
   order by city                                {order by city}
end

format

page header                          {set up page headings}
   print column 20, "CUSTOMER LIST"
   skip 1 line
   print column 18, "September 30, 1994"
   skip 2 lines
   print "Last Name",
   column 18, "Company",
   column 40, "City"
   skip 2 lines

on every row                 {format every row of output}
   print lname,
   column 18, company,
   column 40, city
```

```
after group of city
   skip 2 lines        {skip 2 lines after each group}

on last row                        {count number of rows}
   skip 2 lines
   print column 12, "Total Number of Customers:",
   column 40, count using "##"

page trailer                          {at bottom of page}
   print "Confidential Information",
   column 55, pageno using "Page ##"
end
```

Selecting the Data for a Report

The next part of this chapter explains how you can create a custom report based on your own database. Examples using data from the **stores7** demonstration database illustrate the actions of the various format commands.

Use the SELECT statement to specify what information you want to include in your report. The SELECT statement clauses are the same in report specifications as they are in SQL:

- SELECT clause (required)
- FROM clause (required)
- WHERE clause
- GROUP BY clause
- HAVING clause
- ORDER BY clause
- INTO TEMP clause

Refer to the *Informix Guide to SQL: Syntax* for more information about the syntax and usage of the SELECT statement.

Formatting the Report

In the SELECT section, you can list the columns you want INFORMIX-SQL to retrieve from the database. However, you need not display each column in the report. For example, you might include one of the columns because you want to use it in an ORDER BY clause or in a calculation. You might not necessarily want to print it in the report.

To select, but not print, columns for the report, list all the column names in the SELECT section. Then, in the FORMAT section, list only those columns you want to print.

Use the following sample SELECT statement to select customers' last names, store names, and customer numbers from the **customer** table and order them by customer number (**customer_num**):

```
select customer_num,      {selecting data}
    company, lname
    from customer
    order by customer_num
end
```

To print only the last names and store names, specify the FORMAT section as follows:

```
format
    on every row
        print    {printing data}
        lname, company
end
```

Selecting All Rows and Columns in a Table

To select all the rows and columns in a table, you can use an asterisk (*) with the SELECT statement. For example, to select all the rows and columns from the **orders** table, use the following statement:

```
select *{select all rows from orders table}
    from orders
end
```

Selecting Data from More than One Table

Use SELECT statement syntax to select information for a report. For reports drawing on information found in more than one table, include a WHERE clause in the SELECT statement. The WHERE clause indicates which tables contain the columns you want to join.

The following SELECT statement joins the **stock** table and the **manufact** table:

```
select stock_num, stock.manu_code,
     manu_name, description, unit,
     unit_price
     from stock, manufact
     where
          stock.manu_code =
          manufact.manu_code
end
```

By joining the two tables (on the **manu_code** column), you can include in your report information from the columns in both tables. This example selects stock numbers, manufacturer codes, stock descriptions, units, and unit prices from the **stock** table, and manufacturer names from the **manufact** table.

Reading the Data for a Report

As an alternative to SELECT, you can use the READ statement in the READ section to retrieve data from an ASCII file in unload format. The READ statement specifies the filename, any nondefault delimiter, and optional sorting specifications.

The READ statement lets you retrieve data from files that were created with the UNLOAD statement or with the Output option of PERFORM. You also can retrieve data from files that were created or edited using other software products.

Tip: *A report specification must always contain either a READ section or a SELECT section.*

The syntax of the READ statement includes the following elements:

- READ clause (required)
- DELIMITER keyword
- ORDER BY clause

The following READ statement specifies an ASCII file corresponding to the **stock** table in the **stores7** demonstration database:

```
read "stock1" delimiter ":"
    order by unit_price
end
```

The statement contains an ORDER BY clause, which is used to sort the records according to the values in the **unit_price** field**.** The default sort order is *ascending*, that is, from the lowest to the highest price. The specified delimiter is a colon (:); the default delimiter is a vertical bar (|).

The READ statement requires an ASCII statement in the DEFINE section of the report specification. The ASCII statement assigns an identifier and data type to each value that the READ statement references in the file. The following ASCII statement assigns identifiers and data types to the values in the **stock1** ASCII file. Each field in **stock1** has a corresponding field in the **stock** table.

```
ASCII stocknum smallint, manucode char(3),
    unitdesc char(15), price money,
    unit  char(4),  unitname  char(15)
```

Refer to the *INFORMIX-SQL Reference Manual* for more information regarding the READ section and the READ statement.

Basic Report Formatting

You can format the report to suit your reporting needs. You can create columnar reports as well as reports with irregular formats, such as payroll checks or mailing labels. The FORMAT and OUTPUT sections of your report specification control the appearance and contents of a report. These sections let you customize your report as follows:

- Create page headers, page trailers, and column headings
- Print rows of information in the body of the report

- Group information
- Perform calculations, such as totals and averages
- Set up page margins and dimensions
- Specify whether you want the report to go to the screen or a file

The OUTPUT Section

Use the OUTPUT section to set the left and right margins and the page length for your report. You also can use the OUTPUT section to send the report to a printer, write it to a file, or pipe the report output as input to another program.

You do not need to include an OUTPUT section in every report specification. Without an OUTPUT section, INFORMIX-SQL uses default settings, and the report is displayed on the screen.

The default output settings are as follows:

- LEFT MARGIN, 5 spaces
- TOP MARGIN, 3 lines
- BOTTOM MARGIN, 3 lines
- PAGE LENGTH, 66 lines (11 inches)
- Display the report on the screen
- TOP OF PAGE will fill the remaining lines of the current page with line feeds when a new page is set up

If you include an OUTPUT section, you must place it immediately before the SELECT or READ section in your report specification.

The FORMAT Section

The FORMAT section includes instructions for printing a custom report. You can include page headers, column headings, rows of data, and page trailers in this section if you want a custom report, or it can contain instructions for a simple, default report.

FORMAT for a Default Report

If you do not want a custom report, you can use the EVERY ROW statement in the FORMAT section in place of the control blocks. If you use an EVERY ROW statement, you cannot use any control blocks or other statements in the FORMAT section.

INFORMIX-SQL uses a default format for your report. The default format displays every column and row that the SELECT statement selects.

The following sample FORMAT section for an INFORMIX-SQL report specification uses an EVERY ROW statement:

```
format{format for a default report}
    every row
end
```

FORMAT for a Custom Report

The FORMAT section of a custom report is divided into seven control blocks. You can combine the control blocks in a variety of ways to tailor your reports. The order of these control blocks in the FORMAT section of a report specification is not significant. The seven control blocks are as follows:

- FIRST PAGE HEADER
- PAGE HEADER
- PAGE TRAILER
- ON EVERY ROW
- ON LAST ROW
- BEFORE GROUP OF
- AFTER GROUP OF

Creating Page Headers and Trailers

You can print titles and column headings at the top of each page of a report. You also can print footer information at the bottom of each page. Use the following control blocks in your FORMAT section to control the specified report features:

- FIRST PAGE HEADER
- PAGE HEADER
- PAGE TRAILER

For more information on these control blocks, see the *INFORMIX-SQL Reference Manual*.

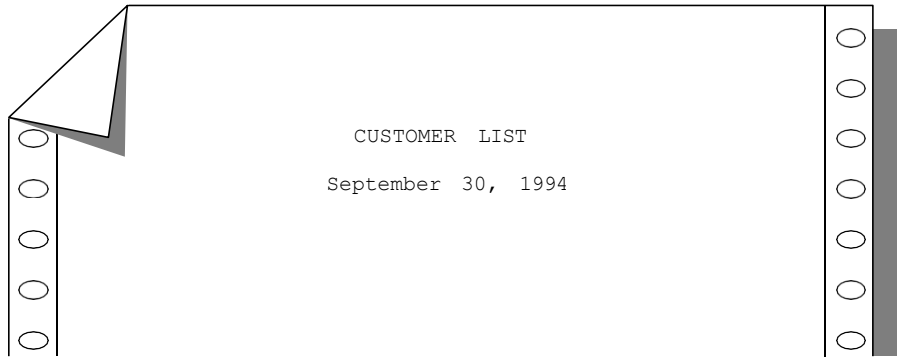
Where Headers and Trailers Appear

Text you specify with the PAGE HEADER control block appears on the line immediately below the top margin on every page of a report. Text you specify with the PAGE TRAILER control block appears at the bottom of each page, on the line immediately above the bottom margin.

If you want different text to appear on the first page of a report, use the FIRST PAGE HEADER control block to specify headings for the first page only.

Printing a Report Heading

Use the PAGE HEADER control block to specify the text you want in the heading on every page. Enter the text exactly as you want it to appear and enclose it in quotation marks. You also can specify where you want the header positioned. For example, you might want to print a report title at the top of the page like this.



The following example shows the PAGE HEADER control block that produces this heading:

```
page header
    print column 20, "CUSTOMER LIST"
    skip 1 line
    print column 18, "September 30, 1994"
```

The following keywords appear in the sample PAGE HEADER control block.

PRINT — Use the PRINT statement to specify the text you want to print and the spacing you want to use. Enclose the text you want to print in quotation marks.

To control spacing, enter the number of spaces you want and the keyword SPACES. For example, to indent text 20 spaces, enter `print 20 spaces`. You can also position text with the COLUMN keyword.

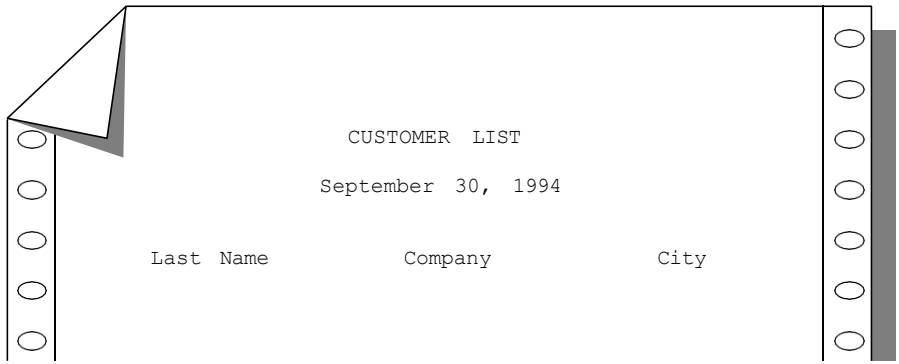
COLUMN — Use the COLUMN keyword to place text in specific column positions. You also can position text with the SPACES keyword.

SKIP — Use the SKIP statement to specify how many blank lines you want to leave. If you want to advance to the top of the next page, use SKIP TO TOP OF PAGE.

Tip: *The COLUMN keyword operates with respect to the left margin. If the left margin is 5 spaces and you specify* `print column 15`*, printing actually will begin in column 20.*

Printing Column Headings

You also can use the PAGE HEADER or FIRST PAGE HEADER control block to specify headings you want to appear above the columns of a report. For example, you might want the following report title and column headings to appear in the report.



To print the title and the column headings, use a PAGE HEADER control block like the one in the following example:

```
page header
    print column 20, "CUSTOMER LIST"
    skip 1 line
    print column 18, "September 30, 1994"
    skip 2 lines
    print "Last Name",     {do not forget}
    column 18, "Company",     {the commas}
    column 40, "City"
    skip 2 lines
```

Tip: *You must use commas to separate column keywords and the data that are to appear on the same line.*

Numbering Pages Automatically

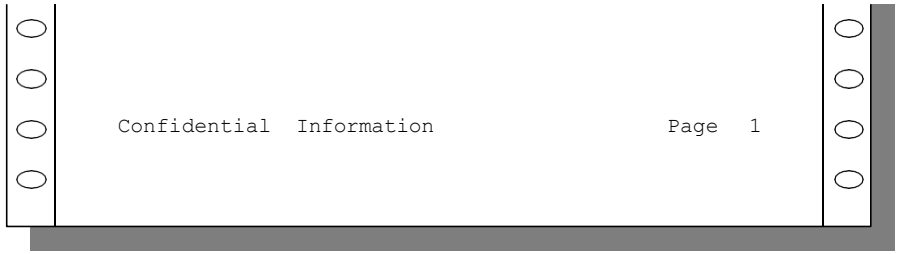
You can include a page number in either the header or trailer by using a PAGENO expression in a PRINT statement. When you print the report, each page is numbered automatically.

The following sample PAGE TRAILER control block produces text and a page number at the bottom of each page:

```
page trailer
    print "Confidential Information",
        column 55, pageno using "Page ##"
```

(Additional information about the keyword USING appears in "Controlling the Appearance of Numbers" on page 8-42.)

This control block produces the following page trailer.



Printing Rows of Data

To print the rows of information in the body of a report, you must use the ON EVERY ROW control block in your FORMAT section. The ON EVERY ROW control block lists the names of the columns that contain the data you want in the report. (Do not confuse the ON EVERY ROW control block with the EVERY ROW statement.)

You can print data only from the columns selected in the SELECT section or from the values retrieved by the READ section. For example, if you do not include the **fname** column in the SELECT section, you cannot list it in the ON EVERY ROW control block of the FORMAT section.

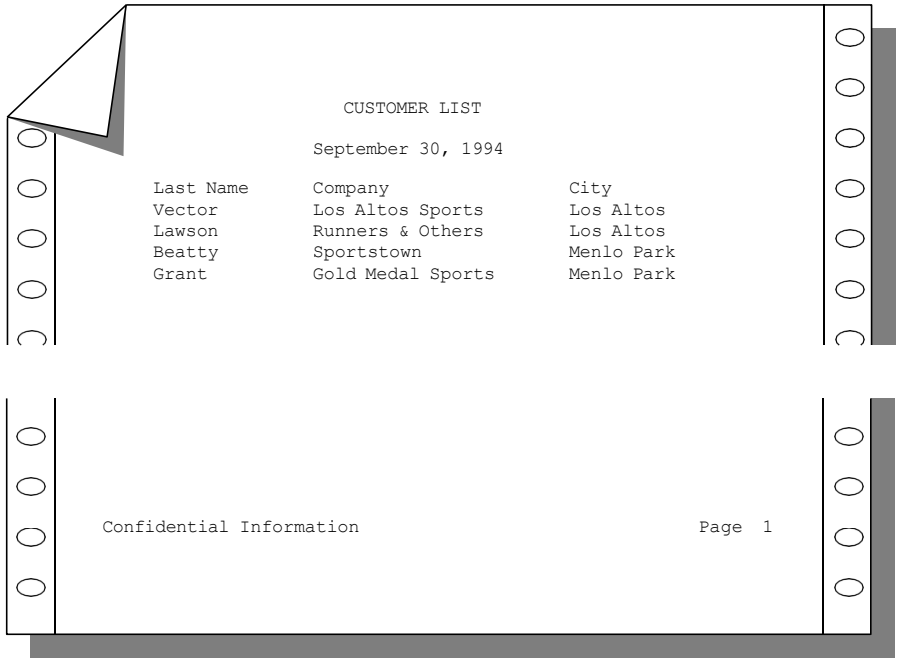
The following example shows a report specification that uses an ON EVERY ROW control block. The ON EVERY ROW control block lists the same column names as the SELECT statement.

```
database
     stores7
end

output
     left  margin  10
end

select
     lname, company, city
     from customer
     order  by  city
end

format

page  header
     print column 20, "CUSTOMER LIST"
     skip 1 line
     print  column  18,  "September  30,  1994"
     skip  2  lines
     print "Last Name",
     column  18,  "Company",
     column  40,  "City"
     skip  2  lines

on  every  row
     print  lname,
     column  18,  company,
     column  40,  city
```

```
    page trailer
        print "Confidential Information",
        column 55, pageno using "Page ##"

    end
```

This specification produces a report that prints the last name, company, and city for every customer.

Here is what a portion of the report looks like.



Organizing the Data

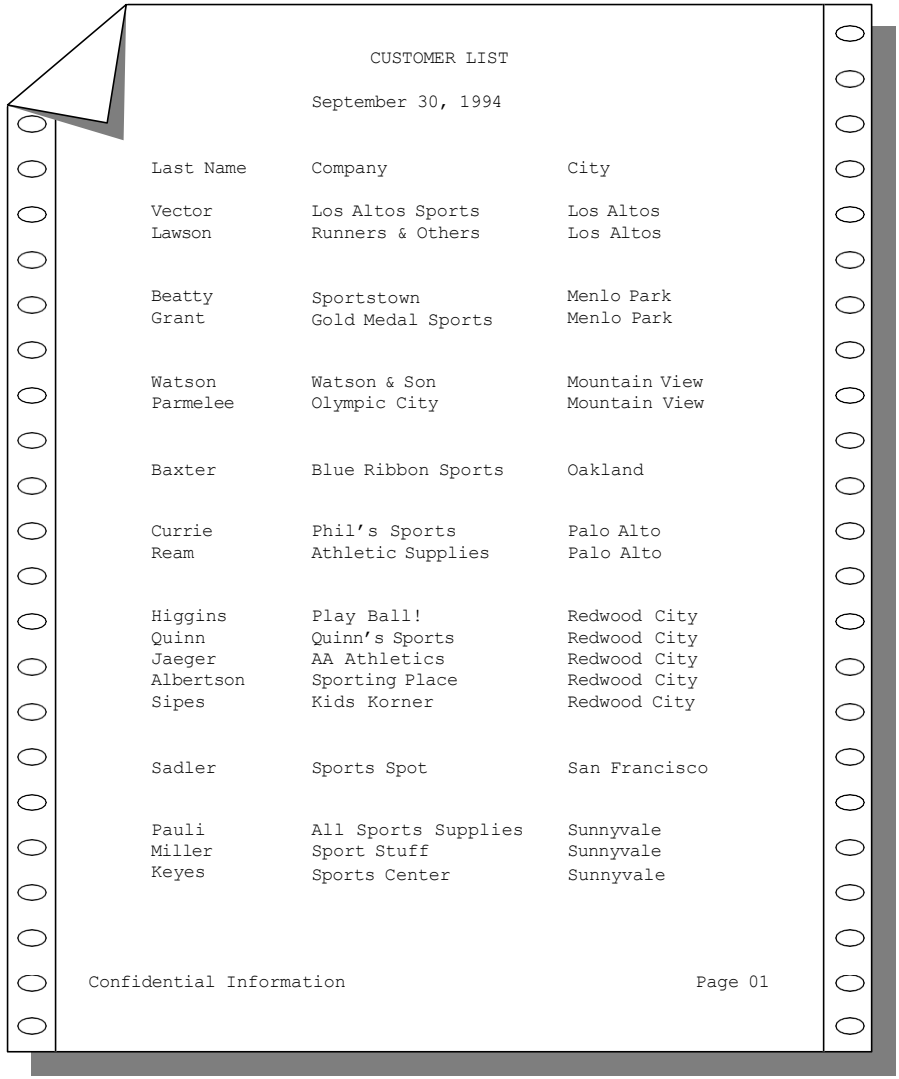
List the column names in the order that you want the information printed in the ON EVERY ROW control block. For example, if you want to alter the preceding report to print the company, followed by last name and city, list the column names in an ON EVERY ROW control block as follows:

```
format
     on every row
     print company,
          column 18, lname,
          column 40, city
end
```

Grouping Data

You can divide the information in a report into groups. For example, you might want to group customers by city, as illustrated in Figure 8-3.

Figure 8-3
A Report with Data Grouped by City



CUSTOMER LIST

September 30, 1994

Last Name	Company	City
Vector	Los Altos Sports	Los Altos
Lawson	Runners & Others	Los Altos
Beatty	Sportstown	Menlo Park
Grant	Gold Medal Sports	Menlo Park
Watson	Watson & Son	Mountain View
Parmelee	Olympic City	Mountain View
Baxter	Blue Ribbon Sports	Oakland
Currie	Phil's Sports	Palo Alto
Ream	Athletic Supplies	Palo Alto
Higgins	Play Ball!	Redwood City
Quinn	Quinn's Sports	Redwood City
Jaeger	AA Athletics	Redwood City
Albertson	Sporting Place	Redwood City
Sipes	Kids Korner	Redwood City
Sadler	Sports Spot	San Francisco
Pauli	All Sports Supplies	Sunnyvale
Miller	Sport Stuff	Sunnyvale
Keyes	Sports Center	Sunnyvale

Confidential Information Page 01

The Group Control Blocks

To group the information in a report, use the following control blocks in your FORMAT section:

- BEFORE GROUP OF
- AFTER GROUP OF

The BEFORE GROUP OF and AFTER GROUP OF control blocks specify what INFORMIX-SQL does when it gets to the beginning or end of a group. To use the group control blocks, enter the name of the column or identifier that contains the information by which you want to group. The same column name or identifier also must appear in the ORDER BY clause in the SELECT section or the READ section.

The instructions you include in a BEFORE GROUP OF control block apply to actions INFORMIX-SQL performs *before* printing the next group of rows. You can use a BEFORE GROUP OF control block to print headings above a group of rows.

The instructions you include in the AFTER GROUP OF control block apply to actions INFORMIX-SQL performs *after* printing a group of rows. You can use an AFTER GROUP OF control block to calculate totals or subtotals at the bottom of a group of rows.

For example, to group the information in a report by city, you first must order by city in the SELECT or READ section. Then, you group by city in a BEFORE GROUP OF or AFTER GROUP OF control block in the FORMAT section.

The following example shows a report specification that groups information by city. As shown in the example, you must use the same column (**city**) to order and to group the information.

```
database
     stores7
end

output
     left  margin  10
end


select
     lname, company, city
     from customer
     order  by  city       {order  by  city}
```

```
end

format

page header
     print column 20, "CUSTOMER LIST"
     skip 1 line
     print column 18, "September 30, 1994"
     skip 2 lines
     print "Last Name",
     column 18, "Company",
     column 40, "City"
     skip 2 lines

on every row
     print lname,
     column 18, company,
     column 40, city

after group of city     {group by city}
     skip 2 lines

page trailer
     print "Confidential Information",
     column 55, pageno using "Page ##"

end
```

The preceding specification produces the CUSTOMER LIST report shown in Figure 8-3. The report groups together all the customers located in the same city.

Using Calculations in a Report

You often might want to perform calculations on the number data you include in a report. For example, you might want to total the values in a column, multiply the values in a column by a constant, or add the values in one column to the values in another.

INFORMIX-SQL provides arithmetic operators, built-in math functions (also called *aggregates*), and group functions that you can include in a report specification to compute values in reports.

Arithmetic Operators

You can perform addition, subtraction, multiplication, division, or exponentiation on any of the number columns in a report. Use any of the following arithmetic operators in the FORMAT section.

Symbol	Function
+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Exponentiation

Example of Arithmetic Operations

For example, if you want to calculate a 15 percent increase in the price for every item in the **stock** table, you could include the following ON EVERY ROW control block in the FORMAT section:

```
on every row
     print stock_num,
     column 15, manu_code,
     column 30, description,
     column 50, unit,
     column 57, unit_price * 1.15
     using "$$$$$.$$"
```

When you run the report, INFORMIX-SQL multiplies the current unit price (**unit_price**) by 1.15 and prints the results in the report.

Here is what a portion of the report would look like.

```
  1        HRO         baseball gloves         case        $287.50
  1        HSK         baseball gloves         case        $920.00
  1        SMT         baseball gloves         case        $517.50
  2        HRO         baseball                case        $144.90
  3        HSK         baseball bat            case        $276.00
  4        HSK         football                case       $1104.00
```

Using Parentheses

You can use parentheses in calculations to control the order in which INFORMIX-SQL computes results. For example, if you want to divide the values in a column by 15 percent of 75, then add 100 to the result, specify the following:

```
print unit_price / (.15 * 75) + 100
    using "$$$$$.$$"
```

Controlling the Appearance of Numbers

To format numbers, include a PRINT USING statement in the FORMAT section. The PRINT USING statement shows the format you want to use for the numbers. Using the following statement,

```
PRINT col1 USING "$,$$$"
```

you can format numbers to include dollar signs and commas, as follows:

```
$5,970
$8,520
$5,550
```

Tip: *If the columns in a table are the MONEY data type, INFORMIX-SQL automatically formats the numbers with dollar signs.*

The symbols between the quotation marks in the following PRINT USING statement are the pattern that INFORMIX-SQL uses to format numbers in the **unit_price** column:

```
print unit_price * 1.15 using "####"
```

Using Fill Characters

The preceding example uses pound symbols (####) as *fill characters*. A collection of fill characters between quotation marks is a *format string*. Use a format string to set up a number format; INFORMIX-SQL substitutes actual numbers for the format string when it prints the report.

In the next example, the format string tells INFORMIX-SQL that every number should include four places. Thus, if the numbers in the report are 45, 456, and 4560, INFORMIX-SQL prints them as follows:

```
  45
 456
4560
```

INFORMIX-SQL prints leading blanks so that every number contains four places. When you use the fill character (#), INFORMIX-SQL adds as many leading blanks as necessary so that every number is right-justified.

If a number is larger than the pattern you set up in the PRINT USING statement, INFORMIX-SQL prints a group of asterisks instead of the number. Thus, if the numbers are 45, 456, 4560, and 45600, they appear in the report as follows:

```
  45
 456
4560
****
```

The Fill Characters

You can use several special fill characters with format strings, as the following table shows.

Character	Function
#	Prints leading blanks if a number does not completely fill the format (as shown in the preceding examples).
&	Prints leading zeros if a number does not completely fill the format (for example, `0045` or `0456`).
-	Prints a leading minus sign if the number is negative. If you use just one minus sign, and the number is negative, INFORMIX-SQL will place a minus sign in the position you indicate in the format string. If you use more than one minus sign, INFORMIX-SQL places it immediately to the left of the most significant digit. (This is often called a "floating" minus sign.)
*	Prints leading asterisks (for example, `**45` or `*456`) if a number does not completely fill the format. (This is useful for printing checks.)
<	Prints numbers left-justified.

You can combine these symbols in a variety of ways to control and print numbers in the format you want. For a detailed explanation of all the number-formatting features available in INFORMIX-SQL, see the discussion of the USING expression in the *INFORMIX-SQL Reference Manual*.

Adding a Dollar Sign

To include a dollar sign and a comma with the numbers, include those symbols in the PRINT USING statement, as follows:

```
print unit_price * 1.15 using "$#,###"
```

Accordingly, INFORMIX-SQL prints the numbers like this:

```
$    45
$   456
$4,560
```

Making the Dollar Sign Float

In the preceding example, the dollar sign always appears in the same position because the # fill character tells INFORMIX-SQL to print blanks if there are no numbers. It places the dollar sign in a specific position in the number.

If you want the dollar sign to float or change position according to the length of the number, tailor the PRINT USING statement as follows:

```
print unit_price * 1.15 using "$$,$$$"
```

Now the dollar sign appears right before each number, regardless of its length:

```
    $45
   $456
 $4,560
```

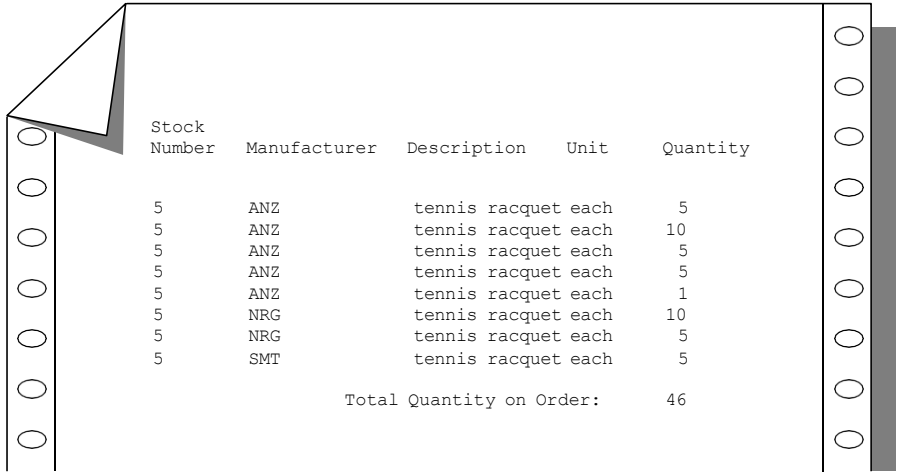
Math Functions

You can perform calculations using any of the following six built-in math functions (or *aggregates*):

TOTAL OF	Adds the values in a column for all rows selected.
MIN OF	Prints the smallest (minimum) value in a column for all rows selected.
MAX OF	Prints the largest (maximum) value in a column for all rows selected.
AVERAGE OF	Calculates the average of the values in a column for all rows selected.
PERCENT	Calculates the percentage of all rows selected that are in a group. (Refer to the *Informix Guide to SQL: Syntax* for details).
COUNT	Counts the number of rows in the table or group you specify.

Totaling a Column

The following sample report lists information about selected items in the **stock** table. In addition, it indicates the total number of these items on order.



```
Stock
Number   Manufacturer   Description    Unit     Quantity

5        ANZ            tennis racquet each      5
5        ANZ            tennis racquet each     10
5        ANZ            tennis racquet each      5
5        ANZ            tennis racquet each      5
5        ANZ            tennis racquet each      1
5        NRG            tennis racquet each     10
5        NRG            tennis racquet each      5
5        SMT            tennis racquet each      5

                    Total Quantity on Order:     46
```

The report specification that produced this report follows:

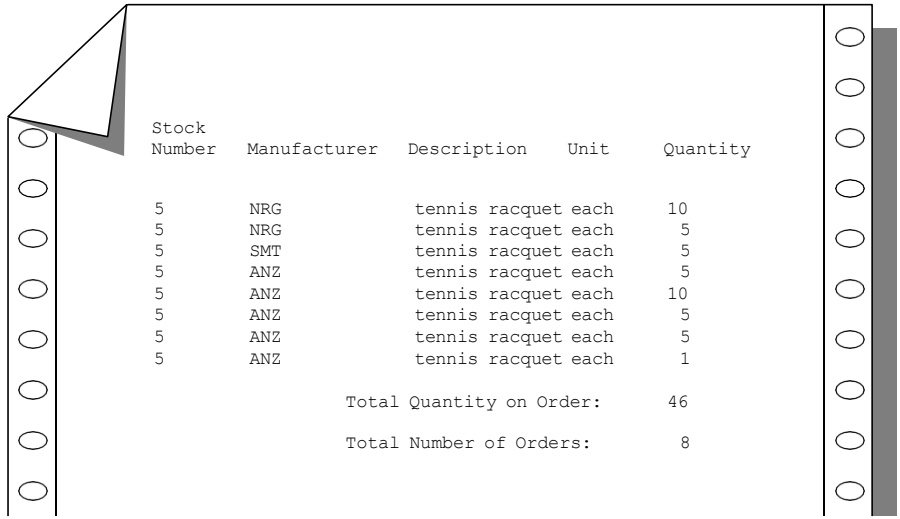
```
database
     stores7
end

select
     items.stock_num, items.manu_code,
     quantity, description, unit
     from items, stock
     where  items.stock_num
     = stock.stock_num and
     items.manu_code
     = stock.manu_code
     and  items.stock_num  =  5
end
format

page  header
     print "Stock Number",
     column 15, "Manufacturer",
     column  30,  "Description",
     column  50,  "Unit",
     column  60,  "Quantity"
     skip 2 lines
on every row
     print stock_num,
     column 15, manu_code,
```

```
        column 30, description,
        column 50, unit,
        column 60, quantity

    on last row
        skip 1 line
        print column 20, "Total Quantity on Order: ",
        column 55, total of quantity using "###"

    end
```

To add all the orders, use the ON LAST ROW control block with a PRINT statement that contains the TOTAL OF function. Specify TOTAL OF followed by the name of the column you want to total.

Counting Rows in a Report

You can use the COUNT function to tally the number of rows that INFORMIX-SQL has selected. For example, the next sample report prints the number of orders, as well as the total quantity of each order, where the stock number is 5.



```
Stock
Number   Manufacturer   Description    Unit     Quantity

5        NRG            tennis racquet each     10
5        NRG            tennis racquet each      5
5        SMT            tennis racquet each      5
5        ANZ            tennis racquet each      5
5        ANZ            tennis racquet each     10
5        ANZ            tennis racquet each      5
5        ANZ            tennis racquet each      5
5        ANZ            tennis racquet each      1

                   Total Quantity on Order:      46

                   Total Number of Orders:        8
```

The report specification that produced this report follows:

```
database
     stores7
end

select
     items.stock_num, items.manu_code,
     quantity, description, unit
     from items, stock
     where  items.stock_num
     = stock.stock_num and
     items.manu_code
     = stock.manu_code
     and  items.stock_num  =  5
end
format

page  header
     print "Stock Number",
     column 15, "Manufacturer",
     column  30,  "Description",
     column  50,  "Unit",
     column  60,  "Quantity"
     skip  2  lines

on  every  row
     print stock_num,
     column 15, manu_code,
     column  30,  description,
     column  50,  unit,
     column  60,  quantity

on  last  row
     skip  1  line
     print  column  20,  "Total Quantity on Order: ",
     column  55,  total  of  quantity  using  "###"
     skip  1  line
     print  column  20,  "Total Number  of  Orders:  ",
     column 55, count using "###"
end
```

The ON LAST ROW control block prints the number of orders and the total quantity of each order.

Combining Arithmetic Operators and Math Functions

You can use both arithmetic operators and math functions in statements in the FORMAT section. To find the average of a column and then multiply the average by a constant, combine the AVERAGE OF function and the multiplication (*) arithmetic operator.

For example, to average the shipping charges (contained in the **ship_charge** column in the **orders** table) and then multiply the result by 1.15, include the following instructions in the FORMAT section:

```
on last row
    skip 1 line
    print (average of ship_charge) * 1.15
```

Group Functions

In addition to arithmetic operators and math functions, INFORMIX-SQL contains functions that you can use to perform calculations on groups of information in a report. The group functions, for example, let you calculate subtotals or interim results.

You can use the following group functions in your report specifications:

- GROUP TOTAL OF
- GROUP COUNT
- GROUP MIN OF
- GROUP MAX OF
- GROUP PERCENT
- GROUP AVERAGE OF

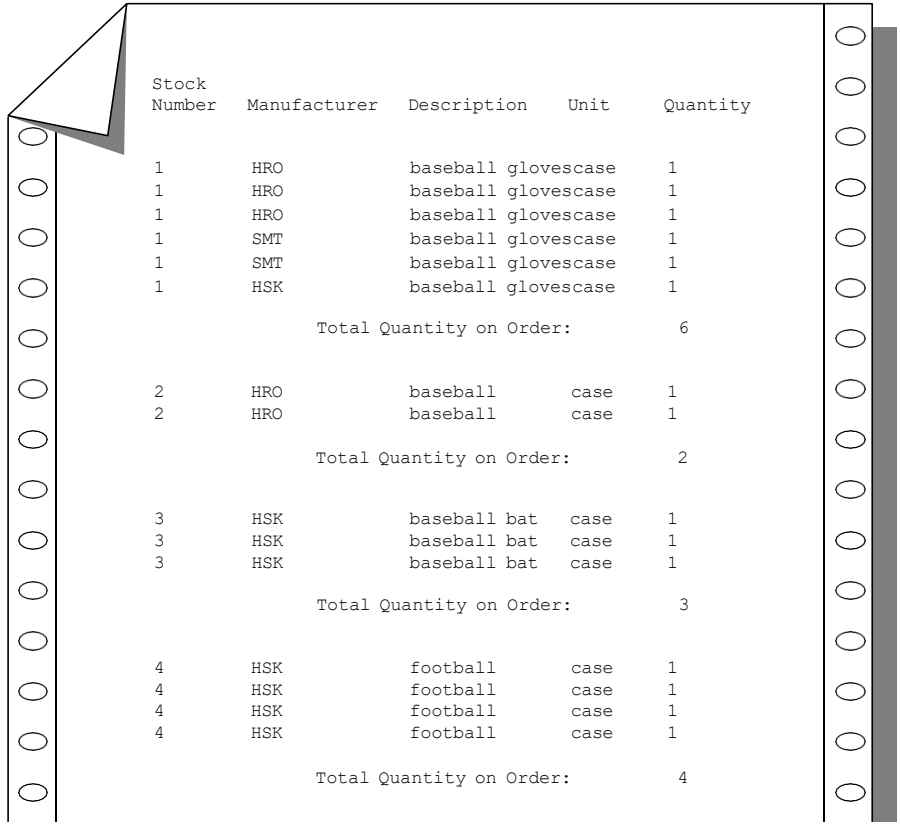
Anytime you use a group function for a column, you also must order by that column in the SELECT or READ section.

To perform calculations on groups of rows in your report, use the group functions in the AFTER GROUP OF control block of the FORMAT section. For example, the following report statement sorts items by stock number and prints the total quantity on order for each item.

In the example, the **stock_num** column in the **items** table is given the display label **stockno**. ACE orders and groups the output by the values in the **stockno** column. (The use of display labels is described in the *INFORMIX-SQL Reference Manual*.)

```
database
     stores7
end

output
     left  margin  0
end

select
     items.stock_num  stockno,  items.manu_code,
     description,    unit, quantity
     from  items,  stock
     where items.stock_num =
     stock.stock_num and
     items.manu_code =
     stock.manu_code
order by stockno
end

format

page  header
     print "Stock Number",
     column 15, "Manufacturer",
     column  30,  "Description",
     column  50,  "Unit",
     column  60,  "Quantity"
     skip  2  lines

on  every  row
     print  stockno,
     column  15,  manu_code,
     column  30,  description,
     column  50,  unit,
     column  60,  quantity

after group of stockno
     skip 1 line
     print  column  20,  "Total  Quantity  on  Order:  ",
     column  55,  group  total  of  quantity  using  "###"
     skip 2 lines

end
```

Here is a portion of the sample report generated by the preceding report specification.



```
Stock
Number   Manufacturer   Description    Unit     Quantity

1        HRO            baseball glovescase     1
1        HRO            baseball glovescase     1
1        HRO            baseball glovescase     1
1        SMT            baseball glovescase     1
1        SMT            baseball glovescase     1
1        HSK            baseball glovescase     1

                Total Quantity on Order:          6

2        HRO            baseball       case     1
2        HRO            baseball       case     1

                Total Quantity on Order:          2

3        HSK            baseball bat   case     1
3        HSK            baseball bat   case     1
3        HSK            baseball bat   case     1

                Total Quantity on Order:          3

4        HSK            football       case     1
4        HSK            football       case     1
4        HSK            football       case     1
4        HSK            football       case     1

                Total Quantity on Order:          4
```

Displaying a Report

You can display report output in several ways:

- Displayed on your screen
- Printed on a printer
- Written to a file
- Piped as input to another program

Specify how you want the report routed in the OUTPUT section of your report specification. You also can use the OUTPUT section to change the margins and page length. The OUTPUT section in a report specification must appear immediately before the SELECT section (or the READ section).

Printing the Report on a Printer

To print the report on a printer, use the REPORT TO PRINTER statement in the OUTPUT section.

The following example shows an OUTPUT section that prints the report on a printer. In addition, it changes the left margin to 10 and the page length to 50.

```
database
    company
end

output
    left  margin  10
    page length 50
    report to printer
end

select lname, company, city
    from customer
end

format
    every  row
end
```

Writing the Report to a File

To send the report results to a file, use the REPORT TO *filename* statement in the OUTPUT section, enclosing the name of the file in quotation marks. For example, to write the report to a file named **cust_rep**, specify the OUTPUT section as follows:

```
output
     report  to  "cust_rep"
end
```

When you run the report, INFORMIX-SQL creates a file named **cust_rep** and writes the report results in it.

Piping the Report to Another Program

For a detailed explanation of piping report results as input to another program, refer to the discussion of the REPORT statement in the *INFORMIX-SQL Reference Manual*.

Calling C Functions from ACE

ACE usually can handle all your database report needs without modification. However, you might occasionally find it necessary to add a feature. For example, a C function called. from ACE might make statistical computations on the data presented in a report and add these to the report. For more information on calling C functions from ACE, see the *INFORMIX-SQL Reference Manual*.

After you have written the file that contains your C functions, you must compile the files and link the necessary library functions to create a custom version of **sacego**. For information on the methods that you can use to simplify the creation and compiling process, see the *INFORMIX-SQL Reference Manual*.

Chapter

9

Database Structure and Integrity

In This Chapter 9-3

Changing the Structure of a Database 9-3
- Removing a Table 9-4
- Removing an Index 9-4
- Removing a Database 9-5
- Renaming a Table 9-5

Changing the Structure of a Table 9-6
- Using the Schema Editor 9-6
  - Adding a Column to a Table 9-7
  - Modifying a Column in a Table 9-8
  - Dropping a Column from a Table 9-8

Transactions 9-10

Audit Trails 9-10
- Comparing Audit Trails and the Transaction Log 9-11

Views 9-11

In This Chapter

This chapter explains how to modify the structure of an INFORMIX-SQL database and table using the INFORMIX-SQL menu system. It also describes ways to ensure database integrity and discusses data types and indexes. The following topics are discussed:

- Deleting a database
- Removing a table or index
- Renaming a table
- Renaming, adding, or dropping a column from a table
- Protecting the integrity of your database with transactions
- Protecting the integrity of a table with audit trails
- Creating dynamic *windows* on your database with views
- Selecting the appropriate data type for a column
- Calculating column and row length in a table
- Using strategies when indexing columns

Changing the Structure of a Database

You can change the structure of a database by:

- Removing a table
- Removing an index
- Renaming a table
- Removing the database

You must have the appropriate database or table privileges to perform these operations. See the *Informix Guide to SQL: Reference*, the *Informix Guide to SQL: Tutorial*, and the *Informix Guide to SQL: Syntax* for more information on privileges and for the syntax and usage for the SQL statements you use to change the structure of a database.

Before you rename a table or remove a table or index, be sure the database containing the table or index is the current database.

Removing a Table

You can remove a table from a database by running the DROP TABLE statement or by using the Drop option on the TABLE menu.

When you select the Drop option on the TABLE menu, INFORMIX-SQL prompts you to enter the name of the table you want to drop. You are then asked to confirm that you do want to drop the table. If you change your mind, press N and the table is untouched. Press Y and INFORMIX-SQL drops the table from the database.

Tip: *INFORMIX-SQL deletes the table you specify from the database. If you accidentally delete the wrong table, you must re-create the table and restore all the data from a backup copy; if you do not have a backup, you must re-enter the data.*

Removing an Index

Use the DROP INDEX statement to remove an index. The general syntax is

```
DROP INDEX index-name
```

where index-name is the name of the index you want to remove.

If you accidentally remove the wrong index, you can easily re-create it with the CREATE INDEX statement.

Removing a Database

You remove a database by running the DROP DATABASE statement or by using the Drop option on the DATABASE menu.

You must own all the tables in the database or have DBA status to drop a database.

To drop the current database, first run the CLOSE DATABASE statement to close it. The syntax for this statement is:

```
CLOSE  DATABASE
```

The Drop option on the DATABASE menu removes a database. When you select the Drop option, INFORMIX-SQL prompts you to enter the name of the database you want to drop. You are then asked to confirm that you do want to drop the database. If you change your mind, press N and the database is untouched. Press Y and INFORMIX-SQL drops the database.

Important: *Be very careful when you drop a database.* INFORMIX-SQL *deletes all tables, indexes, and data from the database.* INFORMIX-SQL *also removes the database directory if all the files in the directory belong to the database. If the database directory contains files that do not belong to the database,* INFORMIX-SQL *removes only the database files and leaves the rest of the directory intact.*

Renaming a Table

Use the RENAME TABLE statement to change the name of an existing table. The general syntax is

```
RENAME  TABLE  oldname  TO  newname
```

where oldname is the current name of the table and *newname* is the name you want to assign to it.

Tip: *When you rename a table you have referred to in reports, screen forms, or INFORMIX-SQL command files, you need to change those references after you rename the table. For reports and screen forms, update the specifications and then recompile them. For INFORMIX-SQL command files, you can just change the references.*

Changing the Structure of a Table

You can change the structure of a table at any time, even if you have stored data in the table. You can:

- Rename a column
- Add a column
- Delete a column
- Change the data type of a column
- Add a unique constraint to a column
- Drop a unique constraint from a column

You can make any of these changes to a table that you create. However, you cannot change the structure of a table created by another person unless you are granted ALTER privilege for that table.

You can only change the structure of tables that belong to the current database. If you try to change a table that does not belong to the current database, INFORMIX-SQL displays an error message.

You can change the structure of a table by using an SQL statement or by using the table schema editor available from the TABLE menu. This section demonstrates the use of the table schema editor. To use SQL statements to change the structure of a table, see the discussion of ALTER TABLE and other SQL statements in the *Informix Guide to SQL: Syntax.*

Using the Schema Editor

You can use the Alter option on the TABLE menu to make changes to the structure of a table with the schema editor.

The examples in this section are based on the **customer** table from the demonstration database. The CREATE TABLE statement that created the **customer** table follows:

```
create table customer
(
    customer_num      serial(101),
    fname             char(15),
    lname             char(15),
    company           char(20),
    address1          char(20),
    address2          char(20),
    city              char(15),
    state             char(2),
    zipcode           char(5),
    phone             char(18)
);
```

Adding a Column to a Table

Use the Alter option on the TABLE menu to add a column to a table schema. The process of adding a column is identical to that of creating a table with the schema editor.

1. Select the Alter option on the TABLE menu. INFORMIX-SQL prompts you for the name of the table you want to change. Select the table, and the ALTER TABLE menu appears. Use the Arrow keys to move the highlight to the part of the schema where you want to position the new column. The new column appears immediately above the highlight.
2. Select the Add option. Enter the information necessary to add the column. When you finish, press RETURN. INFORMIX-SQL again displays the ALTER TABLE menu.
3. Select the Exit option. Then select Build-new-table to make the table addition permanent. INFORMIX-SQL makes the changes to the database table and returns you to the TABLE menu.

Modifying a Column in a Table

You can also use the Alter option on the TABLE menu to change a column definition in the table schema. You can change one part of a column or several parts.

1. Select the Alter option on the TABLE menu. INFORMIX-SQL prompts you for the name of the table you want to change. Select the table, and the ALTER TABLE menu appears. Use the Arrow keys to move the highlight to the part of the schema you want to change.
2. Select the Modify option. INFORMIX-SQL displays the appropriate screen for the part of the schema you highlighted. Make the desired changes. To modify another part of the schema, use the Arrow keys to move the highlight. Make menu selections by using the SPACEBAR to highlight the option you want and then pressing RETURN. The screen changes as you move the highlight.

   If your changes decrease the length of a column, the MODIFY ANYWAY menu appears, with a warning that you may lose data. Select Yes if you want to continue with the change or No if you do not.

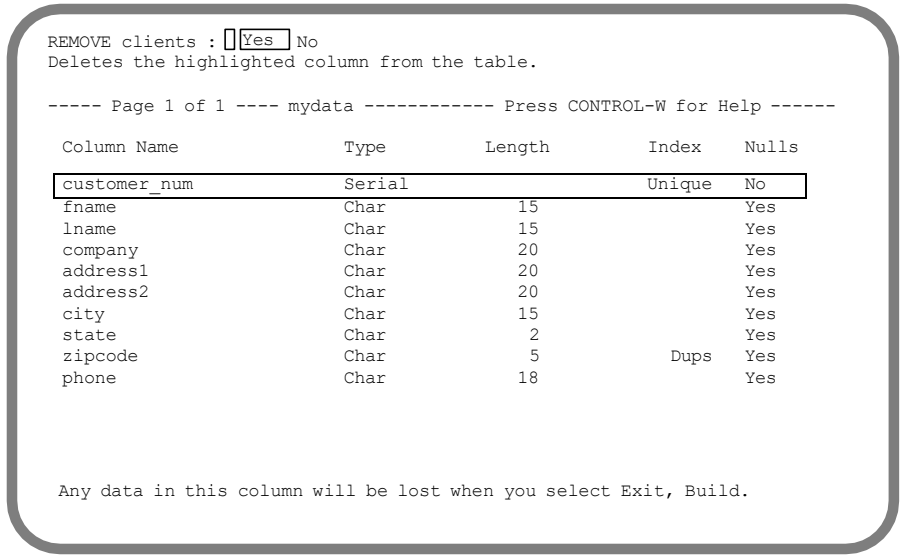
   When you finish your changes, press RETURN. INFORMIX-SQL displays the ALTER TABLE menu.
3. Select the Exit option. Then select Build-new-table to make the change permanent. INFORMIX-SQL makes the changes to the database table and returns you to the TABLE menu.

Dropping a Column from a Table

You can remove any column from the table schema. Use the Drop option on the ALTER TABLE menu when you decide a column is no longer useful.

1. Select the Alter option on the TABLE menu. INFORMIX-SQL prompts you for the name of the table you want to change. Select the table, and the ALTER TABLE menu appears. Use the Arrow keys to move the highlight to the column you want to delete.

2. Select the Drop option by pressing the d key. INFORMIX-SQL displays the REMOVE menu, with a message reminding you that this change will delete data when you select Build-new-table.



3. Select the Yes option from the REMOVE menu to delete the column; select No and the table is untouched. If you select the Yes option, INFORMIX-SQL removes the column from the screen and returns to the ALTER TABLE menu.
4. Select Exit. Then select Build-new-table. If you decide you do not want to remove the column, select the Discard-new-table option. INFORMIX-SQL restores the table schema to its prior state.

Transactions

A *transaction* is a series of operations on a database that you want INFORMIX-SQL to complete entirely or not at all. For example, if you are writing an accounting program, you might want INFORMIX-SQL to perform several operations as a unit to ensure a correct balance. Multiple tables might be affected in the course of completing a series of updates or inserts, and data in the tables must remain i*n balance*. Transactions give you the ability to treat multiple operations as a single transaction. By using transactions, you can ensure the integrity of the data throughout the database.

If you are working within the INFORMIX-SQL menu system on a database with transactions, a TRANSACTION menu appears whenever you attempt to open a new database, run a form, select a user-menu, run a report, or leave the INFORMIX-SQL menu system without terminating the current transaction. The TRANSACTION menu asks you to commit or roll back (discard) any transactions executed since the transaction log was last updated.

For more information about transactions and the SQL statements used with transactions, refer to the *Informix Guide to SQL: Tutorial* and the *Informix Guide to SQL: Syntax*.

Audit Trails

An *audit trail* is a file that contains a history of all additions, deletions, updates, and other manipulations of a database table. The purpose of an audit trail is similar to that of a transaction log. Each is used to maintain a record of modifications to a database, and each can be used to update backup copies of a database.

Comparing Audit Trails and the Transaction Log

A comparison of audit trails and transactions (or a transaction log) follows:

- INFORMIX-SQL records all modifications to a *database* in a transaction log. An audit trail records modifications to a single *table*.
- To use transactions, you must create a database *with transactions*. Thus, you cannot easily add and drop the transaction log, but you can add and delete audit trails as needed.
- With audit trails, you can record modifications to a single important table without incurring the system expense of maintaining a transaction log on the entire database.
- Unlike transactions, audit trails do not protect against statements that fail to run to completion.

For more information about the use of audit trails, refer to the *Informix Guide to SQL: Syntax*.

Views

In certain instances, you might want a user to work with only part of a database, especially if it contains information that you want to remain confidential. With INFORMIX-SQL, you can control what the user sees by creating a dynamic window on the database called a *view*.

Views have the following two advantages over tables:

- Because a view resembles a table in many respects, you can substitute a view name for a table name in most INFORMIX-SQL statements. You can allow the user to insert, select, update, and delete information through a view.
- You can create a view so that INFORMIX-SQL makes sure that any data entered or updated by the user satisfies one or more preconditions.

For example, you can create a view that allows the user to work only with customers in California. The user cannot query on or enter data about a customer who lives in Oregon. This way, you can prevent a user from entering information that will not be accessible through the view.

For more information about the use of views, refer to the *Informix Guide to SQL: Tutorial* and the *Informix Guide to SQL: Syntax*.

Index

A

Abbreviated years 3-15
ACE report writer
  addition 8-41
  AFTER GROUP OF control block 8-39
  aggregates 8-40
  arithmetic 8-41
  AVERAGE aggregate 8-45
  BEFORE GROUP OF control block 8-39
  BOTTOM MARGIN statement 8-29
  calculations 8-40
  calling C functions from 8-53
  clauses in 8-23
  column headings 8-33
  comments 8-24
  compiling a report specification 8-14, 8-21
  control block 8-23, 8-30
  correcting a report specification 8-21
  COUNT aggregate 8-45, 8-47
  creating a report specification 8-12
  custom report 8-30
  DATABASE section 8-9
  default report 8-30
  definition of 1-25, 8-4
  displaying a report 8-52
  division 8-41
  END keyword 8-23
  EVERY ROW statement 8-30
  fill characters 8-43, 8-44
  FORMAT section 8-10, 8-26, 8-29
  formatting a report 8-28
  formatting number expressions 8-42
  generating a default report specification 8-12
  GROUP aggregate 8-49
  grouping data 8-37
  headers 8-31
  joins 8-27
  leading currency symbol 8-44
  LEFT MARGIN statement 8-29
  math functions 8-45
  MAX aggregate 8-45
  MIN aggregate 8-45
  multiplication 8-41
  ON EVERY ROW control block 8-35
  OUTPUT section 8-29, 8-52
  PAGE HEADER control block 8-32
  PAGE LENGTH statement 8-29
  page numbers 8-34
  PAGE TRAILER control block 8-11
  PAGENO expression 8-34
  parentheses 8-42
  PRINT USING statement 8-42
  running a report 8-15
  sample report specification 8-8
  SELECT section 8-9, 8-26
  selecting all rows and columns 8-26
  selecting data for a report 8-25
  selecting data from several tables 8-27
  subtraction 8-41
  summary of sections 8-23

TOP MARGIN statement 8-29
TOTAL aggregate 8-45
trailers 8-31
USING expression 8-42
ACE report, steps in creating 8-5
Active table
definition of 3-6, 5-4
selecting a different 5-4
ADD NUMBER menu, schema editor 2-17
Add option
ALTER TABLE menu 2-29
CREATE TABLE menu 2-13, 2-15
PERFORM 3-12
ADD STARTING NUMBER screen, schema editor 2-21
ADD TYPE menu, schema editor 2-15, 2-20
Adding data, in PERFORM 3-12
AFTER GROUP OF control block 8-11
AFTER GROUP OF control block, in ACE reports 8-39
Alter option, TABLE menu 2-11, 2-28
ALTER TABLE menu
Build-new-table Exit option 2-29
Drop option 9-8
options 2-29
ASCII file
and reports 4-19
and UNLOAD statement 4-19
as input to an application 4-19
retrieving data for a report 8-27
ASCII statement, in ACE reports 8-28
ATTRIBUTES section
COLOR 6-21
COMMENTS 6-23
DEFAULT 6-25
definition of 6-18
description of 6-13, 6-17
DOWNSHIFT 6-24
for multiline editor 3-20
form-specification file 6-13
how to enter 6-19
INCLUDE 6-26
INVISIBLE 6-22
keywords 6-18
rules for assigning 6-20
UPSHIFT 6-24
VERIFY 6-28
WORDWRAP 3-20
Audit trail
definition of 9-10
versus a transaction log 9-11

B

BEFORE GROUP OF control block, in ACE reports 8-39
Boldface type Intro-5
BOTTOM MARGIN statement, in ACE reports 8-29

C

C functions
calling from ACE 8-53
calling from PERFORM 6-34
Calculations in reports 8-40
Calling C functions
from ACE 8-53
in PERFORM 6-34
CENTURY field attribute 3-16
CHANGE ANYWAY menu, schema editor 9-8
CHAR data type
definition of 2-16
determining length 2-16
CHOOSE DATABASE screen, selecting options from 2-9
Choose option, SQL menu 7-15
CLOSE DATABASE statement 9-5
Clustered indexes 2-24
COLOR attribute, in PERFORM 6-21
Columns
adding, using the TABLE menu 9-7
assigning a data type 2-15, 2-20
CHAR data type 2-16
column names 1-7
correspondence between data type and PERFORM field entry 3-13
DATE data type 2-19
DATETIME data type 2-19
DECIMAL data type 2-17
defining a column 2-13
defining for a table 2-13
definition of 1-6
displaying with the Info option 7-18
dropping, using the ALTER TABLE menu 9-8
FLOAT data type 2-18
indexing 2-22
INTEGER data type 2-17
INTERVAL data type 2-20
join 1-11
joining in multiple-table forms 5-8
MONEY data type 2-19
naming conventions 2-15
naming with the Add option 2-15
NULL value in 2-24
number data type 2-16
searching for 1-9
searching for within rows 1-10
SERIAL data type 2-18
SMALLFLOAT data type 2-18
SMALLINT data type 2-17
Command files
guidelines for naming 7-14
how to choose 7-15
saving statements 7-14
Comment icons Intro-6
Comments
displaying on screen in PERFORM 6-23
in SQL statements 7-16
including in a report specification 8-24
COMMENTS attribute, in PERFORM 6-23
COMPILE REPORT menu 8-22
Compiling
definition of 6-7
report specifications 8-21
Contact information Intro-11
Control blocks, in ACE specification file 8-23
Create Database screen 2-6

Create option, DATABASE menu 2-6
CREATE TABLE menu
  building a schema with 1-20, 2-13
  correcting mistakes 2-14
  definition of 2-12
CREATE TABLE statement 7-8
CREATE VIEW statement, WITH CHECK OPTION 9-11
Criteria, for database query 4-10
Current database
  changing 2-9
  definition of 2-8
Current list
  definition of 4-8
  scanning rows with Next and Previous options 4-9
  with multiple-table forms 5-16
Current option
  changing 1-15
  definition of 1-15
  in multiple-table forms 5-16
Current statement, definition of 7-4
Cursor movement, in PERFORM 6-18
customer table, columns in 1-8

D

Data
  adding to a database with PERFORM 3-12
  checking in PERFORM fields 3-17
  confirming your entry in PERFORM fields 3-17
  definition of 1-4
  displaying on the screen with PERFORM 5-11
  entering in a table using a screen form 3-3
  entering in fields with PERFORM 3-13
  how it is stored 1-5
  how to query on 4-3
  including from multiple tables in one form 3-5
  retrieving from a database 1-9
  verifying 5-11
Data type
  assigning to a column 2-15, 2-20
  CHAR 2-16
  DATE 2-19
  DATETIME 2-19
  DECIMAL 2-17
  FLOAT 2-17, 2-18
  for number columns 2-16
  INTEGER 2-17
  INTERVAL 2-20
  MONEY 2-19
  SERIAL 2-18
  SMALLFLOAT 2-17, 2-18
  SMALLINT 2-17
  table of 2-18
  what it does 2-16
Database
  adding data using PERFORM 3-12
  changing the current 2-9
  changing the structure 9-3
  creating tables in 2-10
  creating through the Main menu 2-5
  current 2-8
  definition of 1-4
  how to retrieve data 1-9
  naming conventions 2-6
  querying on 1-9, 4-3
  relational 1-10
  removing a table 9-4
  removing an index 9-4
  renaming a table 9-5
  retrieving information from 1-9
  selecting a different current 2-9
  system files in 2-8
  tables in demonstration 5-5
  user-menu for 1-30
Database management system
  definition of 1-4
  relational 1-10
DATABASE menu, options 2-6
DATABASE section
  in form specifications 6-12
  in report specifications 8-9
DATE data type
  default in form specifications 6-25
  definition of 2-19
  explanation of 2-19
  formatting 3-15
DATETIME data type, definition of 2-19
DBCENTURY environment variable 3-15
dbschema utility, definition of 1-20
DECIMAL data type
  definition of 2-17
  scale and precision 2-17
DEFAULT attribute, in PERFORM 6-24
Delete option, in PERFORM 3-22
Deleting data, in PERFORM 3-22
Delimiters, definition of 3-4
Demonstration database, tables in 1-6, 5-5
Dependencies, software Intro-5
Detail option, in PERFORM 5-13
Detail table, selecting 5-13
Directory, database 2-8
Documentation
  on-line manuals Intro-9
  types of
    related reading Intro-10
Documentation notes, program item Intro-9
DOWNSHIFT attribute, in PERFORM 6-24
DROP INDEX statement 9-4
Drop option
  ALTER TABLE menu 2-29, 9-8
  CREATE TABLE menu 2-13
  DATABASE menu 2-6

E

Editor
  multiline 3-20
  specifying with WORDWRAP attribute 3-20
  SQL 7-9
  system 7-10
Environment variables Intro-5
  DBCENTURY 3-15
Errors, correcting in SQL statements 7-12

EVERY ROW statement, in ACE reports 8-35
Exit option
ALTER TABLE menu 2-29
CREATE TABLE menu 2-13
DATABASE menu 2-6

F

Facilities
for database management 1-20
in INFORMIX-SQL 1-20
report writing 1-25
schema editor 1-20
screen form 1-21
structured query language 1-28
User-menu 1-30
Feature icons Intro-7
Field tag
assignments 6-18
definition of 6-13
Field width, explanation of 6-17
Fields
character, in PERFORM 3-14
checking your data entry in PERFORM 3-17
confirming your data entry in PERFORM 3-17
correspondence to columns 1-22, 3-4
DATETIME, in PERFORM 3-16
DATE, in PERFORM 3-14
definition of 1-21, 3-3
displaying text in color 6-21
editing in PERFORM 3-19
entering data, in PERFORM 3-13
INTERVAL, in PERFORM 3-16
join 5-4
making text invisible 6-22
MONEY, in PERFORM 3-16
number, in PERFORM 3-14
querying on in PERFORM 4-12
too short for search 4-16
File
command 7-14
output, formatting in PERFORM 4-22
perform.out 4-21
saving current statements in 7-14
sending query results with the Output option 7-17
writing reports to 8-53
File extension
.DAT 2-28
.DBS 2-8
.FRM 6-9
.IDX 2-28
.OUT 4-21
.PER 6-9
.SQL 7-15
Fill characters, definition of 8-43
finderr script Intro-9
FLOAT data type, definition of 2-17
Form specifications
ATTRIBUTES section 6-13
changing default values 6-24
choosing tables for 6-6
compiling 6-7
controlling case of input 6-24
correcting errors in 6-8
DATABASE section 6-12
default 6-3, 6-7
definition of 6-3
field tag assignments 6-18
field tags in 6-13
files that make up 6-9
INSTRUCTIONS section 6-33
joining fields in 6-29
JOINING keyword in 6-32
lookup joins on 6-31
maximum number of tables in 6-7
modifying 6-4, 6-10
naming 6-6
SCREEN section 6-12
sections in 6-11
specifying acceptable values in 6-26
TABLES section 6-13
using today as default date 6-25
verify joins with 6-31
verifying input in 6-28
FORMAT section of report specification
AFTER GROUP OF control block 8-11
ON EVERY ROW control block 8-11
ON LAST ROW control block 8-11
PAGE HEADER control block 8-11
PAGE TRAILER control block 8-11
sample section 8-10
Format string, definition of 8-43
Formatting
date values 3-15
report 8-28
Forms
ATTRIBUTES section 6-13
choosing which tables to include 6-6
compiling the default specification 6-7
DATABASE section 6-12
entering data on a screen form 3-3
how to create 6-5
how to name 6-6
modifying an existing one 6-10
multiple-page 4-18
multiple-table 5-3
orderform 5-5, 5-9
querying on 4-10
screen form 1-21
SCREEN section 6-12
specifying acceptable values 6-26
TABLES section 6-13
using default field values 6-24
verifying input 6-28
Functions
C, calling from ACE 8-53
Execute, in PERFORM 3-18
Help, in PERFORM 3-18
Interrupt, in PERFORM 3-18
math, for ACE reports 8-45

G

GENERATE REPORT screen 8-12

H

HELP menu
guidelines for using 1-19
options of 1-19
Help, how to use 1-19
Highest value operator, in PERFORM 4-15
Highlight, how to move 1-15

I

Icons
feature Intro-7
Important Intro-6
platform Intro-7
product Intro-7
Tip Intro-6
Warning Intro-6
Important paragraphs, icon for Intro-6
INCLUDE attribute, in PERFORM 6-26
Indexes
ascending 2-24
clustered 2-24
creating 2-22
descending 2-24
displaying with the Info option 7-18
for columns with duplicate values 2-23
removing from a database 9-4
unique 2-23
when to index a table 2-23
INFO menu
options 7-18
with SQL 7-18
Info option
SQL menu 7-18
TABLE menu 2-11
Information lines, in PERFORM 3-9
INFORMIX-SQL
facilities 1-20
Main menu 1-14
what is 1-4
Insert mode, in PERFORM 3-19
INSTRUCTIONS section, in form specification 6-33
INTEGER data type, definition of 2-17
INTERVAL data type, definition of 2-20
INVISIBLE attribute, in PERFORM 6-22

J

Join
definition of 1-10
fields 6-29
fields on screen forms 5-8
field, definition of 5-8
in multiple-table form 5-4
in PERFORM 5-8
in the stores7 demonstration database 1-10
JOINING keyword 6-32
lookup 5-12, 6-31
verify 5-11, 6-31
Join columns, definition of 5-8
JOINING keyword in a form specification 6-32

K

Keys
arrow, with SQL editor 7-9
backspace, in PERFORM 3-18, 3-20
change mode, in PERFORM 3-20
clear screen, in PERFORM 3-20
cursor positioning, in PERFORM 3-18
delete a character, in PERFORM 3-20
delete a character, with SQL editor 7-9
delete forward, in PERFORM 3-20
delete rest of line, with SQL editor 7-10
editing keys, in PERFORM 3-20
editing keys, in SQL 7-9
escape, with SQL editor 7-9
execute, in PERFORM 3-18
fast backspace, in PERFORM 3-18
fast forward, in PERFORM 3-18
field editing, in PERFORM 3-19
for displaying HELP menu 1-19
for leaving a menu 2-10
for moving the highlight 1-15
forward, in PERFORM 3-18, 3-20
help, in PERFORM 3-18
insert mode, with SQL editor 7-9
interrupt, in PERFORM 3-18
next field, in PERFORM 3-18
repeat data, in PERFORM 3-20
Keywords in ACE
COLUMN 8-33
DELIMITER 8-28
END 8-21, 8-23
SPACES 8-33
USING 8-42
Keywords in PERFORM
ATTRIBUTES 6-13
DATABASE 6-12
END 6-12
JOINING 6-32
LOOKUP 6-32
SCREEN 6-12
TO 6-27
WHERE 6-22
WITHOUT NULL INPUT 6-25

L

Leading currency symbol 8-45
LEFT MARGIN statement, in ACE reports 8-29
Lookup fields
definition of 5-12, 6-32
in form specifications 6-32
Lookup joins
definition of 6-31
in multiple-table forms 5-12
in PERFORM 5-12
on a screen form 5-12
Lowest value operator, in PERFORM 4-15

M

Main menu
  accessing 1-14
  creating a database 2-5
  creating a table 2-10
  Database option 1-17, 2-6
  Exit option 1-17
  Form option 1-15, 1-16, 3-6, 6-5
  Query-language option 1-16, 7-5
  Report option 1-16
  Table option 1-17
  User-menu option 1-17
Master option, in PERFORM 5-16
Master-detail relationship, introduction to 5-12
Match, exact in PERFORM query 4-4
Math functions, in ACE reports 8-45
Menus
  ADD TYPE 2-15
  ALTER TABLE 2-29
  calling with the User-menu facility 1-30
  CHANGE ANYWAY 9-8
  CREATE TABLE 2-12
  creating custom with User-menu facility 1-30
  current option 1-15
  custom (user-menus) 1-30
  DATABASE 2-6
  HELP 1-19
  INFO 7-18
  INFORMIX-SQL Main menu 1-14, 2-5
  PERFORM 3-11
  REPORT 8-6
  selecting options 1-16
  SQL 7-5
  SYNTAX 7-6
  TABLE 2-11
Message line of a menu 1-15
Mode
  insert, in PERFORM 3-19
  real, invoking INFORMIX-SQL 1-14
  typeover, in PERFORM 3-19
Modify option
  ALTER TABLE menu 2-29, 9-8
  CREATE TABLE menu 2-13
MODIFY REPORT menu
  Compile option 8-20
  Discard-and-exit option 8-20
  Save-and-exit option 8-20
MODIFY REPORT screen 8-19
Modifying data, with PERFORM 3-21
MONEY data type
  definition of 2-19
  determining column length 2-19
Moving the highlight 1-15
Multiline editor, in PERFORM 3-20
Multiple-table form
  active table for 5-4
  current option 5-16
  definition of 5-3
  join fields 5-8
  lookup join 5-12
  master-detail relationship 5-12
  orderform 5-9
  selecting the detail table 5-13
  selecting the master table 5-16
  verify join 5-11

N

Naming conventions
  for a database 2-6
  for a table 2-12
  for columns 2-15
  for report specifications 8-13
Next option, in PERFORM 4-9
NULL values, in a column 2-24
Numbers, appearance in reports 8-42

O

ON EVERY ROW control block, in ACE reports 8-11, 8-35
ON LAST ROW control block, in ACE reports 8-11
On-line error messages Intro-9
On-line manuals Intro-9
Operators
  arithmetic, combining with math functions 8-49
  arithmetic, in reports 8-41
  range, in query 4-10
  relational, in query 4-10
  value, in query 4-10
Option, how to select 1-16
OUTPUT FORMAT screen, in PERFORM 4-22
Output option
  PERFORM 4-19
  SQL menu 7-17
OUTPUT section
  displaying for reports 8-52
  for a report 8-29
  in ACE reports 8-29, 8-52

P

PAGE HEADER control block, in ACE reports 8-11, 8-32
PAGE LENGTH statement, in ACE reports 8-29
PAGE TRAILER control block, in ACE reports 8-11
PERFORM
  active table in 5-4
  checking your data entry 3-17
  COLOR attribute 6-21
  COMMENTS attribute 6-23
  comparison with SQL 7-7
  confirming your entry in a field 3-17
  current list 4-8
  Current option 5-16
  DEFAULT attribute 6-25
  Detail option 5-12
  DOWNSHIFT attribute 6-24
  entering search criteria 4-10
  entering search criteria for query 4-4
  Execute function 3-18
  field editing feature 3-19
  Help function 3-18
  how to access 3-6
  how to exit 3-22

INCLUDE attribute 6-26
information lines section 3-9
Interrupt function 3-18
INVISIBLE attribute 6-22
locating all rows in a table 4-9
Master option 5-12
menu options 3-11
Output option 4-19
positioning the cursor 3-18
query operators and short fields 4-16
Query option 4-4
querying on a range of values 4-14
querying on alternative values 4-16
querying on CHAR fields 4-13
querying on DATE fields 4-12
querying on DATETIME fields 4-12
querying on detail table 5-13
querying on FLOAT and SMALLFLOAT fields 4-17
querying on highest value 4-15
querying on INTERVAL fields 4-12
querying on lowest value 4-15
querying on MONEY fields 4-12
querying on number fields 4-12
removing a row 3-22
screen for orderform form 5-5
screen form section 3-10
Screen-format option 4-22
searching for a group of rows 4-7
storing a row 3-21
three sections of screen 3-8
updating a row 3-21
UPSHIFT attribute 6-24
VERIFY attribute 6-28
PERFORM screen transaction processor
Add option 3-12
adding data with 3-12
CHAR fields in 3-14, 4-13
current list 4-8
cursor positioning keys 3-18
DATE fields in 3-14, 4-12
DATETIME fields in 3-16, 4-12, 4-15
Delete option 3-22
deleting data with 3-22
Detail option 5-13
displaying comments on the screen 6-23
editing keys in 3-20
equal to operator 4-11
exiting from 3-22
explanation of queries in 4-10
highest value operator 4-15
how to access 3-6
Information lines in 3-9
insert mode 3-19
INTERVAL fields in 3-16, 4-12, 4-15
introduction to 3-5
lowest value operator 4-15
Master option 5-16
menu options 3-11
modifying data with 3-21
MONEY fields in 3-16, 4-12
multiple-page forms 4-18
Next option 4-9
number fields in 3-14, 4-12
pictures in 3-17
Previous option 4-9
range operator 4-14
relational operators in 4-11
sample queries 4-18
Screen option 4-18
search criteria in 4-4
SERIAL fields in 3-14
Status lines in 3-10
Table option 5-4
typeover mode 3-19
Update option 3-21
what it is 3-5
wildcard characters in 4-13
PICTURE attribute 3-17
Platform icons Intro-7
Previous option, in PERFORM 4-9
PRINT USING statement 8-42
Printing query results with SQL 7-17
Privilege, displaying with the Info option 7-18
Product icons Intro-7
Program group
Documentation notes Intro-9
Release notes Intro-9
Program, calling with the User-menu facility 1-30

Q

Query
by example 4-4
definition of 1-9, 4-3
examples with PERFORM 4-18
finding all rows in a table 4-9
interactive 7-4
locating a single row 4-4
moving between screens 4-18
on detail table 5-13
range operators 4-10
relational operators 4-10
saving results in PERFORM 4-19
saving results with SQL 7-17
searching for a group of rows 4-7
searching for a range of values 4-14
searching for alternative values 4-16
searching for an exact match 4-4
searching for columns 1-9
searching for columns within rows 1-10
searching for highest value 4-15
searching for lowest value 4-15
searching for rows 1-9
searching in FLOAT and SMALLFLOAT fields 4-17
sending results to a file 7-17
sending results to a printer 7-17
specifying search criteria for 4-10
structured 1-28
using SQL 1-28
using wildcard characters 4-10, 4-13
value operators 4-10
Query language, definition of 1-28, 7-4
Query option, in PERFORM 4-4
QUERY screen, in PERFORM 4-4

R

Range operators
  in PERFORM 4-14
  with DATETIME fields, in PERFORM 4-15
  with INTERVAL fields, in PERFORM 4-15
READ section, of report specification 8-27
READ statement, in a report specification 8-27
Real mode, invoking INFORMIX-SQL 1-14
Related reading Intro-10
Relational DBMS
  advantages of 1-12
  definition of 1-10
Relational operators
  CHAR fields, in PERFORM 4-13
  DATE fields, in PERFORM 4-12
  DATETIME fields, in PERFORM 4-12
  in PERFORM 4-11
  INTERVAL fields, in PERFORM 4-12
  MONEY fields, in PERFORM 4-12
  numeric fields, in PERFORM 4-12
Release notes, program item Intro-9
Remove option, in PERFORM 3-22
RENAME TABLE statement 9-5
REPORT menu
  Compile option 8-6
  Drop option 8-6
  Generate option 8-6, 8-12
  Modify option 8-18
  New option 8-6
  Run option 8-15
  summary of options 8-6
Report specification
  compiling 8-21
  correcting 8-21
  DATABASE section 8-9
  default, compiling 8-14
  default, creating 8-12
  definition of 8-5
  errors in 8-21
  FORMAT section 8-10
  group functions 8-49
  including comments in 8-24
  modifying 8-18
  OUTPUT section 8-29
  READ section 8-27
  saving 8-20
  SELECT section 8-9
  sequence of sections 8-23
  summary of sections 8-23
  what it contains 8-23
REPORT TO statement, in ACE reports 8-53
Report writer, ACE 8-3
Reports
  arithmetic operators in 8-41
  basic formatting 8-28
  calculations in 8-40
  counting rows in 8-47
  definition of 8-3
  displaying output 8-52
  dollar sign in 8-45
  formatting 8-26
  group control blocks 8-38
  grouping data 8-37
  headers 8-31
  how numbers appear in 8-42
  modifying 8-18
  naming conventions 8-13
  numbering pages automatically 8-34
  organizing the data 8-37
  printing a heading 8-32
  printing column headings 8-33
  printing rows of data 8-35
  retrieving data from an ASCII file 8-27
  running 8-15
  sample report 8-7
  sample SELECT statements 8-26
  selecting data for 8-25
  steps in creating with ACE 8-5
  trailers 8-31
  using fill characters 8-43
  writing 1-25
  writing to a file 8-53
  writing with ACE 8-3
Row
  adding rows to a table 1-7
  changing the contents with PERFORM 3-21
  counting in ACE report 8-47
  definition of 1-6
  in the stores7 demonstration database 1-7
  locating all in a table 4-9
  locating with a query, in PERFORM 4-4
  removing with PERFORM 3-22
  searching for 1-9
  searching for a group, in PERFORM 4-7
  storing with PERFORM 3-21
Run option, SQL menu 7-12
RUN REPORT screen 8-16
Running SQL statements 7-12

S

sacego, and C function calls in ACE 8-53
Saving query results with SQL 7-17
Schema editor
  ADD NUMBER menu 2-17
  ADD STARTING NUMBER screen 2-21
  ADD TYPE menu 2-20
  adding a column to a table 9-7
  ALTER TABLE menu 9-7
  assigning data types 2-15
  CHANGE ANYWAY menu 9-8
  changing a table 2-28
  correcting mistakes 2-14
  creating tables 1-20
  defining a column 2-13
  definition of 2-13
  dropping a column from a table 9-8
  exiting 2-26
  modifying a column in a table 9-8
  naming a table 2-12
Schema, for a table 1-20, 2-13

Screen
  CHOOSE 7-15
  INFO FOR TABLE 7-18
  NEW 7-9
  PERFORM 3-8, 5-5
  RUN FORM 3-7
  SAVE 7-14
  USE-EDITOR 7-11
Screen forms
  active table on 5-4
  choosing tables when creating 6-6
  choosing the one you want 3-7
  creating multiple 3-6
  default 6-9
  definition of 1-21, 3-3
  definition of multiple-table 5-3
  entering data in a table 3-3
  features 1-21
  how to use 3-6
  in the demonstration database 3-4
  including data from multiple tables 3-5
  join fields on 5-8
  master-detail relationships on 5-4
  maximum number of tables on 3-5, 6-7
  modifying 6-4
  naming 6-6
  verify joins on 5-11
Screen option
  ALTER TABLE menu 2-29
  CREATE TABLE menu 2-13
  in PERFORM 4-18
Search criteria
  definition of 4-4
  in PERFORM 4-4
Select option, DATABASE menu 2-6
SELECT section, of report specification 8-9
SELECT statement, in a report specification 8-9
SERIAL data type
  assigning starting number 2-21
  definition of 2-18
SMALLFLOAT data type, definition of 2-17, 2-18
SMALLINT data type, definition of 2-17
Software dependencies Intro-5
SQL
  command file 7-15
  commenting 7-16
  comparison to PERFORM 7-7
  correcting errors in statements 7-12
  creating a table 7-8
  current statements 7-4
  definition of 7-4
  dropping an index 9-4
  editing keys 7-9
  editor 7-9, 7-11, 7-12
  entering statements 7-8
  errors, correcting 7-12
  formatting statements 7-16
  how to access from the Main menu 7-5
  how to use 7-3
  introduction to 2-4
  learning how to use 7-6
  long statements 7-11
  naming current statements 7-14
  options on SQL menu 7-5
  renaming a table 9-5
  running statements 7-12
  saving statements 7-14
  separating statements with semicolons 7-16
  specifying a system editor 7-11
  statements, how to use 1-28, 7-4
  SYNTAX menu 7-6
  system editors 7-11
  Use-editor option 7-11
SQL menu
  Choose option 7-5, 7-15
  Drop option 7-6
  Exit option 7-6
  Info option 7-6, 7-18
  introduction to 7-5
  Modify option 7-5, 7-13
  New option 7-5, 7-9
  options 7-5
  Output option 7-5, 7-17
  Run option 7-5, 7-12
  Save option 7-5, 7-14
  Use-editor option 7-5, 7-11
SQL statements
  correcting errors in 7-12
  current 7-4
  examples 7-16
  formatting 7-16
  how to enter 7-8
  how to run 7-12
  how to use 7-4
  indicating comments 7-16
  separating with semicolons 7-16
Statements
  correcting errors 7-12
  CREATE TABLE 7-8
  current 7-4
  definition of 7-4
  editing in SQL 7-9
  how to format 7-16
  how to run 7-12
  in ACE report specification 8-23
  saving in a command file 7-14
  SQL, definition of 1-28
Status
  displaying with the Info option 7-18
  lines, in PERFORM 3-10
Status lines, in PERFORM 3-10
stores7 demonstration database
  customer table 9-7
  examples 1-41
  join columns 1-10
  report specifications 8-7, 8-8
  summary of tables 1-6
System catalog, definition of 2-9
System editor 7-11
System files
  extensions for 2-28
  where they are stored 2-28
System requirements
  database Intro-5
  software Intro-5

T

Table
  active, in PERFORM 3-6, 5-4
  changing the structure 2-28, 9-6
  connecting through column joins 1-10

creating through menus 1-20, 2-12
defining columns for 2-13
definition of 1-5
detail, selecting 5-13
displaying information in 7-18
dropping a column 9-8
finding all rows with Query option 4-9
how to build with the schema editor 2-13
how to create with SQL 7-8
in system catalog 2-9
in the stores7 demonstration database 1-6
joining columns 1-11
listing those in a database 7-18
master, returning to 5-16
modifying a column 9-8
multiple, in a form 5-3
naming 2-12
naming conventions 2-12
removing a row 3-22
removing, using the TABLE menu 9-4, 9-5
renaming 9-5
schema 1-20, 2-13
selecting data for a report 8-26
storing a row 3-21
system files that make up a table 2-28
using the schema editor 9-6
TABLE menu
schema editor 9-6
summary of options 2-11
Table option, in PERFORM 5-4
Table schema, how to reproduce 1-20
TABLES section, in form specifications 6-13
Tip icons Intro-6
TOP MARGIN statement, in ACE reports 8-29
Transaction log versus an audit trail 9-11
TRANSACTION menu, in an ANSI-compliant database 9-10
Transactions, definition of 9-10
Typeover mode, in PERFORM 3-19

U

Update option, in PERFORM 3-21
UPSHIFT attribute, in PERFORM 6-24
Use-editor option, SQL menu 7-11
User-menu, definition of 1-30
USING expression, in ACE reports 8-42
Utilities, calling with the User-menu facility 1-30

V

Values
NULL, in a column 2-24
querying on a range in PERFORM 4-14
querying on highest in PERFORM 4-15
querying on lowest in PERFORM 4-15
VERIFY attribute, in PERFORM 6-28
Verify joins
definition of 6-31
in form specifications 6-31
in multiple-table forms 5-11
on a screen form 5-11
View
advantages of 9-11
definition of 9-11

W

Warning icons Intro-6
Wildcard characters, in PERFORM 4-13
Writing query results to a file with SQL 7-17

Y

Y2K compliance 3-15
Years, abbreviated 3-15