Database Desktop User's Guide

Delphi for Windows

Introduction

Copyright Agreement

Database Desktop is a compact relational database application that you can use either as a standalone application on a single computer running Windows or as a multiuser application on a network. It gives users of supported products an easy way to create, edit, and query tables.

This *User's Guide* contains information on using Database Desktop to work with data tables in a variety of formats. Database Desktop is a powerful tool you can use to

- View tables and edit the data in them
- Query databases to retrieve information
- Create new tables or modify the structure of existing tables

SQL

If you have Borland® SQL Link, you can use it with Database Desktop. For more information, see the Database Desktop Help Contents.

Windows note

To gain the full benefits of Database Desktop, you should first familiarize yourself with Microsoft Windows. This manual assumes you already understand Windows and Windows-specific terminology. For help with Windows, see your Windows documentation.

Inside this manual

This manual contains seven chapters and one appendix:

Chapter 1, "Database concepts," describes database concepts and the basics of Database Desktop.

Chapter 2, "The Database Desktop window," describes the Database Desktop window, how to open and close files, and how to work with directories.

Chapter 3, "Creating tables," explains how to create new tables and how to restructure existing tables.

Chapter 4, "Managing objects," explains how to sort, rename, copy, and delete tables.

Chapter 5, "Entering and editing data," explains how to customize your view of a database, and also discusses data-entry tasks you can perform.

Chapter 6, "Running SQL statements," explains how to use the SQL Editor to enter and execute SQL statements.

Chapter 7, "Querying table data with QBE," describes how to extract and manipulate information in databases.

Appendix A, "Using the keyboard," lists the keyboard equivalents for Database Desktop mouse actions.

Conventions

The different typefaces in this manual are used as follows:

`Monospace` typeface represents text as it appears on the screen, and anything you must type. *Italics* are used for table names, for emphasis, and to introduce new terms. *Keycap* typeface indicates a key on your keyboard. It often indicates a key you should press—for example, "Press *Enter* to complete an entry."

Note Notes give additional information on the subject at hand, such as exceptions to a general rule, or more technical detail for advanced users.

Caution Cautions alert you to the potential loss of information.

Tip Tips include helpful suggestions and shortcuts.

When commands appear in full (the name of the command preceded by the "path" used to get to it), the individual commands in the path are separated by vertical bars (|). For example, "File|Open" refers to the command you choose by opening the File menu, then choosing Open.

Using this book

- Start by reading Chapter 1 if you're not familiar with relational databases.
- Consult the remaining chapters for more details on specific operations.
- Use the online Help system.
- Work with the sample tables.

Chapter

1

Database concepts

This chapter introduces some Database Desktop concepts:

- Understanding tables
- Using keys, indexes, and links
- Querying (searching) databases

This manual assumes that you're familiar with basic Windows terminology.

Understanding tables

A database is an organized collection of information or data. An address book is a simple example of a database. It organizes data about people into specific categories: names, phone numbers, and addresses.

In a relational database, the data is organized into tables. Each row of a table contains information about a particular item; this is called a *record*. Each column contains one piece of the information that makes up a record; this is called a *field*. The next figure shows how information is divided into records and fields.

Figure 1.1 A table

Table : CUSTOMER.DB

CUSTOMER	Cust ID	Last Name	Init	Street	City	State	Zip
1	1,386.00	Aberdeen	F	45 Utah Street	Washington	DC	20032
2	1,388.00	Svenwald	I	Gouvernment House	Reykjavik		
3	1,784.00	McDougal	L	4950 Pullman Ave NE	Seattle	WA	98105
4	2,177.00	Bonnefemme	S	128 University Drive	Stanford	CA	94323
5	2,579.00	Chavez	L	Cypress Drive	Palm Springs	FL	32938
6	2,779.00	Fahd	S	The Palace	Riyadh		
7	3,128.00	Elspeth, III	R	1 Hanover Square	London		
8	3,266.00	Hanover	A	15 State Street	Dallas	TX	75043
9	3,271.00	Massey	C	29 Aragona Drive	Oxon Hill	MD	29902
10	3,771.00	Montaigne	L	30 Tauton Drive	Bellevue	WA	98004
11	4,277.00	Matthews	R	P. O. Box 20336	Albuquerque	NM	87234

This row is one record. It contains one value for each field.

This column is one field. It contains one kind of information about a record.

Relational tables

Relational database applications such as dBASE and Paradox give you a way to link tables by comparing values stored in comparable fields in separate tables. The data in a relational database is the information stored in all the related tables. The advantage of a relational database is that you can easily extract or combine data from several tables to get exactly the information you need, without changing the structure of the database. Also, a few small and discrete tables are more convenient to use and maintain than one large table.

The sample database files CUSTOMER.DB and BOOKORD.DB are examples of relational tables. These tables can be linked through the fields containing customers' ID numbers (Cust ID in the *Customer* table, Cust in *Bookord*). When the tables are linked, you can extract information from both tables into one table. For example, you can search for and extract a list of quantities ordered (from the *Bookord* table) and the respective last names (from *Customer*). The results are returned in an *Answer* table (see page 7).

Using indexes

An index is a file that Database Desktop uses to keep track of the location of records in a table. This makes it easy for Database Desktop to

- Maintain a sorted order of a table
- View like values together, if you have Paradox

When you create an index, Database Desktop creates a file that contains the indexed field's values and their locations. Database Desktop refers to the index file when locating and displaying the records in a table. With Paradox tables, this is true of both *primary indexes* (keys) and *secondary indexes*. A secondary index gives you the ability to view a Paradox table's data in an order different from that of the table's key, if you have Paradox.

Database Desktop lets you create secondary indexes for Paradox tables, but it does not let you view tables by secondary index order.

Both Paradox and dBASE tables let you create indexes to specify the order in which records are accessed. However, the way indexes work is different for Paradox and dBASE tables.

Paradox primary indexes

A table can have many indexes defined, but you must identify one of them as the *primary index*. In Paradox tables, the primary index is called the *key*. The key is a field (or group of fields) that contains data that uniquely identifies each record of a table. A key requires a unique value for each record (row) of a table, ensuring that you won't have duplicate records in the table. This means that only one record in the table can be blank in the key field. Tables that have keys are called *keyed tables*.

Paradox uses keys to establish the default sort order for a table: records are sorted by the values in the field(s) defined as the table's key. For example, in the table *Customer*, Cust ID is the key field; the table is sorted by the values in that field.

A primary key can be defined on a single field or group of fields. When a *group* of fields is specified as a table's key, that group is called a *composite key*. With composite keys, duplicate values are allowed in individual fields of the key, as long as values are not duplicated across all fields of the key.

For information on specifying keys, see "Defining keys" on page 26.

Paradox secondary indexes

In Paradox, you can use a secondary index to see an alternate view order for a Paradox table. For example, to view the *Customer* table by City in Paradox, while keeping the table's key order intact, you can use a secondary index on City to temporarily change the view order of the records.

Database Desktop lets you create secondary indexes for Paradox tables, though it does not let you view tables by secondary index order.

You can create an index on a group of fields. This is a *composite secondary index*. It organizes the data by the first field of the index first, then by the second, and so on.

For information on creating secondary indexes, see "Defining secondary indexes" on page 33.

dBASE indexes

As with Paradox tables, indexes for dBASE tables determine the order in which Database Desktop accesses the records in a table. Indexes organize records so that data can be found more quickly. When a dBASE table is indexed, Database Desktop creates a file that contains the indexed field's values and their corresponding record numbers. Database Desktop refers to this index file when locating and displaying the records in a table.

dBASE tables can have maintained or non-maintained indexes.

For more information, search for "dBASE indexes" in the keyword list in Database Desktop Help.

Querying databases

A *query* is a question you ask about the information in database tables. It can be a simple question about information in one table, or a complex question about information in several tables.

You can use queries to

- Find or select information from a table
- Combine information from multiple tables
- Perform calculations on data in a table
- Insert or delete data in a table
- Change values in a table
- Define groups and sets of information on which to perform calculations and comparisons

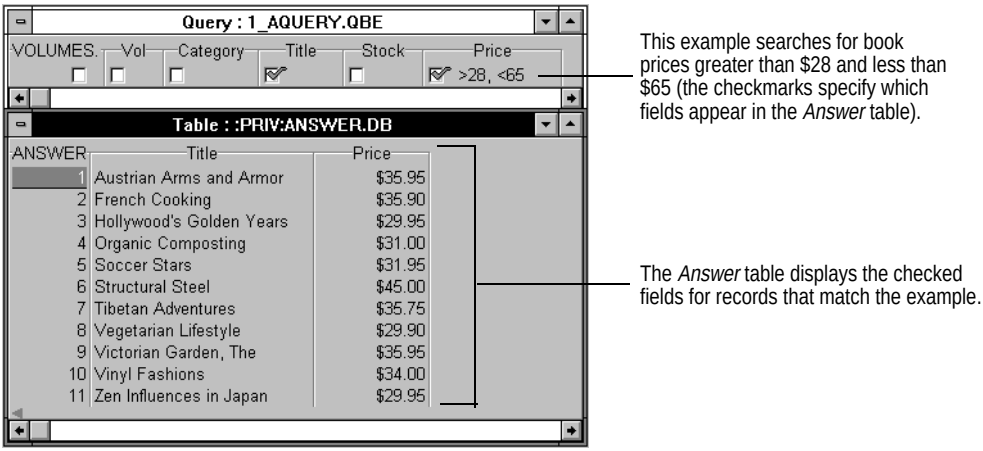
Query by example

Database Desktop uses a technique called *query by example* (QBE) to extract and manipulate data in external database tables. When you run a query in Database Desktop you actually *access* external database tables. You can also search several related tables at a time.

You perform a query in Database Desktop in the Query window. First you choose which table(s) to query. Then you enter an example of the information you want and run the query. Database Desktop looks at the query and the tables involved, and determines the best way to arrive at the result. The result appears in a separate *Answer* table (see the next section).

The next figure shows a query that gives examples of the fields you want to see (and a range of values within one of those fields), and the answer Database Desktop gives.

Figure 1.2 A query and its results



For details on using QBE, see page 93.

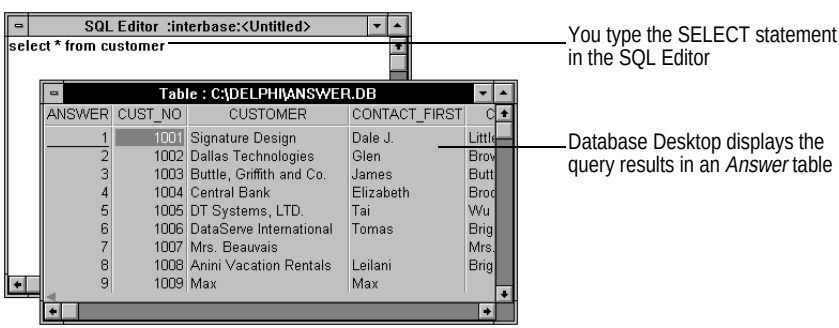
Answer tables

The result of a query is a temporary table called *Answer*. The *Answer* table is overwritten each time a query is run. For information on saving the data in an *Answer* table, see "Saving the query" on page 98.

Other temporary tables are sometimes created when a query is run, such as *Deleted* and *Inserted*.

SQL queries

You can query SQL tables using either QBE or the SQL Editor. The following figure shows a query in the SQL Editor. Database Desktop displays the query results in an *Answer* table.



For details on the SQL Editor, see Chapter 6.

Using DDE links

Database Desktop uses Dynamic Data Exchange (DDE). DDE lets you link data in one application to corresponding data in another application, using the Paste Link command. DDE links are live; when data in one application changes, those changes are automatically updated in the other application.

The application that is the source of the data to be exchanged is called the *DDE server*. The application that receives the exchanged data is the *DDE client*. Database Desktop can be a DDE client, server, or both.

For more information on DDE links, see Database Desktop Help; choose Help | Contents | Tasks | Exchanging Data.

Chapter

2

The Database Desktop window

This chapter discusses the Database Desktop application window and its menus. You'll learn how to

- Start Database Desktop
- Identify and use areas of the Database Desktop window
- Identify file types
- Open and close files
- Establish a working directory
- Establish a private directory
- Create and use aliases

Starting Database Desktop

To start Database Desktop from the Program Manager, double-click the Database Desktop icon or choose File | Run to run DBD.EXE.

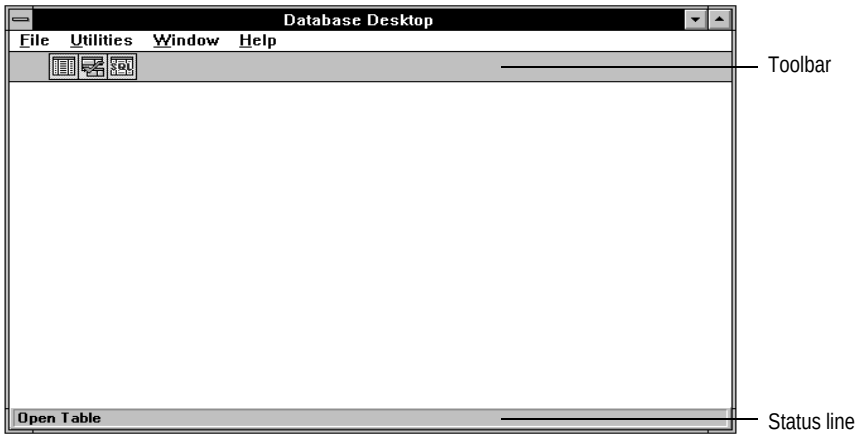
Some applications let you start Database Desktop with a menu command.

Database Desktop has several command-line options that let you control its configuration. For information on each option and its use, search for "command-line configuration" in the keyword list in Database Desktop Help.

The Database Desktop window

The first time you start Database Desktop, the empty Database Desktop application window opens. All Database Desktop windows are opened in and contained by this window.

Figure 2.1 The Database Desktop application window



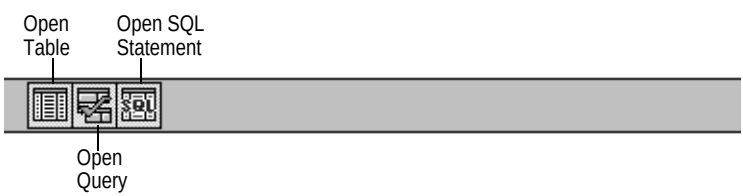
Files you open in Database Desktop appear in their own type of windows. Tables appear in Table windows, and queries appear in Query windows.

Each type of window has some commands and functions that apply only to that type. The File, Utilities, Window, and Help menus are always available in the application window, even when no other windows are open.

The application window Toolbar

Below the menu is a Toolbar. The Toolbar changes when the active window changes. The following figure describes what operations you can perform using the application window Toolbar.

Figure 2.2 Application window Toolbar



Arranging windows

Use commands on the Window menu to arrange windows and icons:

- **Tile** divides the available space in the application window among open table and query windows.
- **Cascade** stacks open table and query windows so their title bars are visible.
- **Arrange Icons** arranges file icons evenly on the application window.

- **Close All** closes all open Database Desktop windows, except the application window.

Managing files

In Database Desktop you work with three types of files: QBE queries, .SQL files, and tables. The other types of files are created automatically by Database Desktop.

The following table lists the file extensions of all files used by Database Desktop.

Table 2.1 File extensions

Extension	Type of file
.CFG	Configuration files, used to store aliases and system configuration settings.
.DB	Paradox table.
.DBF	dBASE table.
.DBT	Memo field values for a dBASE table.
.FAM	Listing of related Paradox files (like a table's .TV file)
.INI	Preference files, used to store working directory settings.
.MB	Memo field values for a Paradox table.
.MDX	Maintained index of a dBASE table.
.NDX	Non-maintained index of a dBASE table.
.PX	Primary index of a Paradox table.
.QBE	Saved QBE query.
.SQL	Saved SQL statement.
.TV	Table view settings for a Paradox table.
.TVF	Table view settings for a dBASE table.
.VAL	Validity checks and referential integrity for a Paradox table.
.X*nn*	Secondary index for a Paradox table, numbered. A secondary index defines an alternate view order to temporarily change the *display* order of the records.
.Y*nn*	Secondary index for a Paradox table, numbered.
.XG*n*	Composite secondary index for a Paradox table.
.YG*n*	Composite secondary index for a Paradox table.

By default, all indexes organize and access data in ascending order (A to Z, or 0 to 9).

Opening files

1. Choose File | Open.
2. Choose the type of file to open—QBE query, SQL statement, or table.
3. Specify the file to open.

The dialog box has a Drive (or Alias) list and a File Type list.

Note To access tables stored on a network, you must tell Database Desktop the location of the network control file. You do this by running the BDE Configuration Utility; double-click

the BDE Configuration Utility icon in the program group where the Database Desktop icon appears. See online Help in the BDE Configuration Utility for more details.

The Drive (or Alias) list

This list indicates what directory Database Desktop is displaying. If you've given a directory path a name (an alias), the list displays that name, rather than the full path. By default, Database Desktop looks in your working directory, which is assigned the alias :WORK: . (See page 13 for more information on working directories. Aliases are discussed further on page 14.)

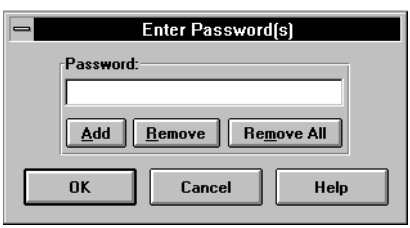
The File Type list

This list shows what type of files are listed: tables, queries, or .SQL files.

Tables with passwords

In most applications, *passwords* can be defined that protect tables from unauthorized access. When you first try to access a password-protected table in Database Desktop, the Enter Password(s) dialog box appears. To access a password-protected table, type the password and then choose OK.

During a session, the password for a table need only be entered once because Database Desktop maintains a *password list* to keep track of all passwords. When you exit Database Desktop, however, the password list is erased. To open the Enter Password(s) dialog box, choose Utilities | Passwords. You can use the Enter Password(s) dialog box to manage the password list:



To enter multiple passwords, choose the Add button after each password you type. This lets you enter passwords for several password-protected tables that you plan to use in a session.

To delete one or more passwords, choose the Remove button after each password you type. This lets you delete passwords from the password list.

To delete every password, choose the Remove All button to delete the password list for the current session.

Note While a password-protected table is open or in use by a query, deleting its password from the password list has no effect until you close the table or query. The next time you try to access the table, however, the Enter Password(s) dialog box will appear.

Creating new files

You can create tables, QBE queries, and SQL statements:

- To create a table, choose File | New | Table or right-click the Open Table button in the Toolbar, then choose New. For information on creating tables, see Chapter 3, "Creating tables."
- To create a query, choose File | New | QBE Query or right-click the Open Query button in the Toolbar, then choose New. For information on queries, see page 93.
- To create a SQL statement, choose File | New | SQL Statement or right-click the Open SQL Script button in the Toolbar, then choose New.

Saving files

To save a query or SQL statement, choose File | Save or File | Save As. Database Desktop saves QBE queries with a .QBE file extension and SQL statements with a .SQL file extension.

To save a table you're creating, choose Save As in the Create Table dialog box. Database Desktop saves the table structure with a .DB or .DBF file extension and closes the dialog box. If you didn't complete all the necessary steps or if you've entered conflicting or invalid specifications, Database Desktop prompts you to correct the problem(s).

You don't use the File | Save or File | Save As commands to save a table's data. Database Desktop automatically saves data as soon as you do anything that unlocks the record (such as moving off the record, turning off Edit mode, or choosing Record | Unlock).

To save a table's property changes (column width, arrangement, scroll locks, and row height), choose Properties | Save from the Table window. Changes are saved to a file with the same name as the table, and with a .TV (for Paradox) or .TVF (for dBASE) file extension.

Note If you make changes to a table's properties and don't save them, Database Desktop asks if you want to save them when you close the table.

Setting up a working directory

The *working directory* is where Database Desktop looks first for your files. The Working Directory setting controls what files are listed in File | Open and File | Save dialog boxes. So, for example, if you want to open C:\DBD\SAMPLES\BOOKORD.DB, make C:\DBD\SAMPLES your working directory so that you see BOOKORD.DB when you choose File | Open | Table.

By default, the working directory is the one that holds your program files. To specify a different working directory, choose File | Working Directory, then type the path to the directory.

SQL You cannot set your working directory to an alias on a remote server.

Setting up a private directory

You need to store your temporary tables, such as *Answer*, in a non-shared directory or they could be overwritten by other users or other open applications on your own

computer. Database Desktop therefore gives you a private directory in either a multiuser or standalone environment.

Choose File | Private Directory to establish a private directory.

Files stored in your private directory are listed in File | Open and File | Save dialog boxes, preceded by `:PRIV:` . Private directory files are visible and available to you, but not to other network users.

Note If you don't specify a private directory, Database Desktop creates one on your local hard disk (for example, C:\DBD\DBDPRIV). If you have no local hard disk, the network home directory on the file server should be used as the private directory. Do not use a floppy drive for a private directory.

Aliases

An external database is a collection of files. These files can be kept in a directory on your local hard disk or on a network server. You can assign a name (an*alias*) as a shortcut to a directory using the Alias Manager dialog box.

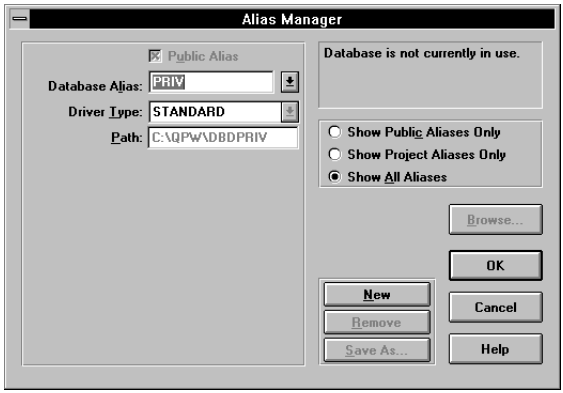
For example, if you have a collection of tables and queries in one directory (called C:\DBD\PROJECTS\CUSTLIST), you can just specify the alias :MYWORK: rather than type the entire path.

Using aliases, you can avoid typing long path names, and you can use the Path list in File | Open and File | Save dialog boxes to list files in a different aliased directory.

Creating an alias

1 Choose File | Aliases. The Alias Manager dialog box appears:

Figure 2.3 The Alias Manager dialog box



2 Choose the New button.

3 In the Database Alias edit field, type the name (alias) you want to give the directory.

4 Leave STANDARD as the driver type. This lets you create an alias for Paradox and dBASE tables.

SQL If you're a user of Borland SQL Link and you want to create an alias for a remote database, see the Database Desktop Help Contents for help on SQL Link.

5 Enter the full path to the directory in the Path edit field.

6 Choose whether you want to show only public aliases, only project aliases, or all aliases. A project alias is displayed only when the directory where it is stored is your working directory, if you choose Only Project Aliases.

7 Choose Keep New. A message in the dialog box tells you that the alias has been added to the session.

8 To make this a permanent alias—available any time you use Database Desktop—choose Save As and save the alias to the file IDAPI.CFG. (Otherwise, the alias will exist only until you exit Database Desktop.)

9 Choose OK to exit the Alias Manager dialog box.

Changing an alias definition

1 Choose File | Aliases.

2 Specify the name of the alias to change in the Database Alias edit field (you can choose it from the list).

3 Edit the path.

4 If you want to make this change permanent, choose Save As and save the new definition to the file IDAPI.CFG.

5 Choose OK.

Removing an alias

1 Choose File | Aliases.
2 Specify the alias to remove.
3 Choose the Remove button.
4 Choose OK.

.SQL files

A .SQL file is an object that contains code you write in SQL (Structured Query Language). Information about using SQL with Database Desktop and about using Database Desktop to work with remote data is located in the Database Desktop Help system.

You can use the SQL Editor to write SQL code to perform operations on remote data using Borland SQL Links. You can also write query scripts using SQL that you can run on local Paradox or dBASE data. For information about using the SQL Editor, see Chapter 6.

Chapter

3

Creating tables

This chapter discusses how to create and restructure Paradox, dBASE, and SQL tables in Database Desktop. It shows you how to

- Define field names, choose field types, and specify field sizes
- Define validity checks and table lookup
- Create keys, indexes, and referential integrity
- Establish password security
- Restructure existing tables
- View table structures

Note You can create tables in Database Desktop with many sophisticated field types, some of which you can enter data in only if you have Paradox or dBASE. In Database Desktop you cannot edit data in the following field types:

- **Paradox**: Memo, Formatted Memo, Graphic, OLE, Autoincrement, Binary, or Bytes
- **dBASE**: Memo, OLE, or Binary
- **SQL**: any BLOB (binary large object) field or a text field that allows more than 255 characters

To enter or edit data in these field types, use Paradox for Windows or dBASE, or an application that supports editing those field types.

Planning tables

Planning is the first step in creating a table. You need to decide what you want the table to contain and how you want to lay it out. When you plan a table, keep these guidelines in mind:

- *Put as little information as possible in each field.* This allows for more flexible data maintenance and more straightforward querying. For example, if you break an address into separate fields for street, city, and state, you can easily query on these specific field values.

- *Be complete*. Try to include fields for all the information you think you'll need, but don't clutter the table with information you don't need. If you discover later that you need another field, you can add it then.
- *Use small tables*. If you have a great deal of information to organize, it's generally better to put it in several small, related tables rather than in one all-encompassing table.
- *Keep your tables familiar*. It's often best to create tables that correspond to the kinds of objects—like forms and files—you already use.
- *Avoid redundancy*. Beyond the common fields necessary for linking tables, don't duplicate information in different tables.
- *Consider what kind of table you need*. Because you can easily create Paradox, dBASE, or SQL tables, weigh the advantages of each. For example, Paradox tables support passwords, validity checks, referential integrity, and a greater variety of field types. dBASE tables support soft deletions and are fully compatible with existing dBASE applications. Determine what your needs are before you choose a table type.

Paradox, dBASE, or SQL table?

Specifying the table type in the Table Type dialog box determines

- Some rules about the table's structure, such as valid field names, types, sizes, and the rules for specifying key fields. For example, Paradox tables can contain spaces and punctuation in names, while dBASE and SQL cannot.
- Where you can create the table. For example, if you specify a Paradox for Windows table, you cannot save that table on an alias for a remote server.

In choosing a table type, consider the features you will need:

- Paradox offers validity checks, referential integrity, and table lookup, while dBASE and SQL tables do not.
- Paradox tables refresh when the data is changed, but dBASE and SQL tables do not.
- Paradox has sequence numbers while record numbers in dBASE might not correspond to the sequence numbers in the table. SQL tables do not have record numbers.
- Paradox has formatted memo fields, as well as long integer, time, timestamp, autoincrement, and bytes fields.

Creating a new table

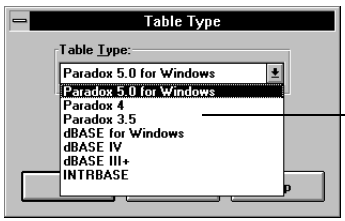
The following sections describe how to create new tables with Database Desktop. To restructure an existing table, see page 47.

Guidelines for creating tables

To create a new table from the Desktop,

1 Choose File | New | Table. Or right-click the Open Table Toolbar button, and choose New.

The Table Type dialog box appears.



Choose the type of table you want to create. Some options discussed in this chapter are available only to Paradox for Windows 5.0 tables.

2 If you want a table type other than Paradox for Windows, click the arrow next to the list box and select from the drop-down list.

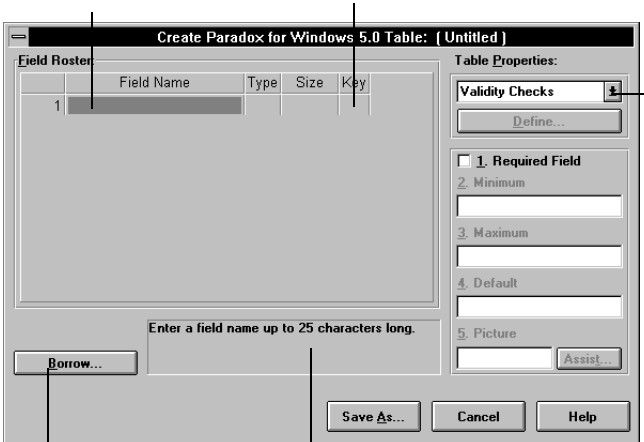
3 Choose OK.

The Create Table dialog box appears, where you can specify the structure of the new table.

Figure 3.1 The Create Table dialog box

Enter the field name, type, and size in the Field Roster.

Press any key or double-click to key the table.



When the dialog box is opened, the Validity Checks table property is selected and all types of validity checks are available.

Choose this to borrow the structure of another table.

The status box gives you guidelines as you create the table.

In this dialog box, you

- Name the fields of the table (page 20)
- Specify field types and sizes (page 22)

Optionally, you can also

- Assign a key to the table (page 26)
- Define validity checks for individual fields (page 27)
- Establish a table lookup to another table (page 31)
- Assign secondary indexes to the table (page 33)
- Establish referential integrity with another table (page 36)
- Specify password security for the table or individual fields (page 40)
- Specify a table language (page 43)

Creating a simple Paradox table

To create a simple, unkeyed Paradox table, follow these steps:

1. Type the name of the first field in the Field Name column of the Field Roster. Rules about field names are discussed in the next section.
2. Move to the Type column.

   You can move among the columns of the Field Roster by pressing *Tab*, *Shift+Tab*, or *Enter*, or by using the arrow keys or the mouse. Database Desktop automatically skips over any columns that are not required.
3. Press *Spacebar* or right-click the Type column to display a list of field types. Type the symbol for the field type you want. For more information on Paradox field types, see page 22.
4. Move to the Size column and type an appropriate field size (if a size is required). Refer to Table 3.1 on page 23 for information on field sizes.
5. Press ↓. Repeat steps 1 through 4 until you've specified as many fields as you want.
6. Choose Save As to name and save the table.

After you save the structure, you can open the table and enter data in your new table.

Defining fields

Use the Field Roster in the Create Table dialog box (page 19) to define the fields of the new table. You can use the mouse, arrow keys, *Enter*, *Tab*, or *Shift+Tab* to move among the columns. (*Shift+Tab* moves backwards.) As you move, a status message at the bottom of the dialog box prompts you for valid entries. A vertical scroll bar appears in the Field Roster if there are more fields than Database Desktop can display.

Field names

Type field names in the Field Name column of the Field Roster. The following lists give you rules for naming fields in Paradox tables and in dBASE tables.

Rules governing Paradox field names

- The maximum length of a field name is 25 characters.
- A field name can't start with a blank space (unless it's enclosed in quotation marks), but it can contain blank spaces.

- Each field name in a table must be unique. (You can't have two identical field names.) You can't make a name unique by
  - Adding a blank space at the end of the name
  - Changing the capitalization of the name
- A field name shouldn't contain these characters:
  - Square brackets [], curly braces {}, or parentheses ()
  - The combination –>
  - The symbol # by itself (you can combine # with other characters, as in the field name Phone #)

Rules governing dBASE field names

- A field name cannot exceed 10 characters.
- A field name cannot contain blank spaces.
- Each field name in a table must be unique. You cannot have two identical field names. You cannot make a name unique by
  - Adding a blank space at the end of the name
  - Changing the case of the name

Rules governing SQL field names

For information on field names for your SQL server, search for "field names" in the keyword list in Database Desktop Help, and choose the topic for your server.

To delete a field

Place the insertion point in any column of the field you want to delete, then press *Ctrl+Del*. Database Desktop deletes the entire row.

If you do not want to delete a whole row, place the insertion point in the column whose value you want to delete and press *Backspace* or select the field and begin typing. Database Desktop overwrites the previous value.

To insert a field

Select the field below which you want to insert the field and press *Ins*. Database Desktop opens a blank row, ready for you to type the field name.

To edit a field name

Select the field. Click again to position the insertion point where you want it, or press *F2*. Then edit as you normally would.

Tip To replace an entire field name, select the name you want. The entire field is highlighted. Begin typing to replace the old name with the new one.

To reorder fields

Click the row number of the field and drag it to its new location. Remember that Paradox key field(s) must occur first in the table's structure.

Specifying field type

Use the Create Table (page 19) or the Restructure Table (page 48) dialog box to specify field types.

To specify the field type,

1 Display the structure of the table in the dialog box.

2 Select the Type column of the field you want.

3 Type the symbol (or name, for SQL tables) for the field type or select from the drop-down list. You can use the list in two ways:

- Right-click the Type column again and click to select the field type.
- Press *Spacebar* to see the list, then choose the field type.

Paradox field types and sizes

A field's type determines the kind of data you can enter in it.

Remember, in Database Desktop you can create all Paradox and dBASE field types and most SQL field types, but you cannot view or edit data in the following field types:

- **Paradox**: Memo, Formatted Memo, Graphic, OLE, Binary, or Bytes
- **dBASE**: Memo, OLE, or Binary
- **SQL**: any BLOB (binary large object) field or a text field that allows more than 255 characters

To enter or edit data in these field types, use Paradox for Windows or dBASE, or an application that supports editing those field types.

To specify a field type, position the insertion point in the Type column of the field list, and use one of these methods:

- Right-click or press *Spacebar* in the type column to display a menu of types. Choose the type you want.
- Type the appropriate symbol (refer to Table 3.1 for type symbols).

The following table lists each field type's symbol and size constraints, and describes the kind of data that each field type can contain.

Table 3.1 Paradox field types

Field type	Symbol	Size values	Description
Alpha	A	1–255 (required)	Contains letters, numbers, special symbols (like %, &, #, and =), or any other printable character.
Number	N	None	Contains numbers in the range 10^{-307} to 10^{308} of up to 15 significant digits.[1] Number fields are best used when you want to perform calculations on the values in the field. Use an alpha field, rather than a number field, when you want to include parentheses or hyphens (as in phone numbers and zip codes).
Money	$	None	Contains numbers in the range 10^{-307} to 10^{308} of up to 15 significant digits. Money fields are exactly like number fields, but are formatted to display decimal places and a money symbol. Regardless of the number of decimal places displayed, Database Desktop recognizes up to six decimal places when performing internal calculations on money fields.
Short	S	None	Contains whole numbers in the range –32,767 to 32,768. Short fields do not allow the same formatting options as number fields, and should be used only by advanced users.
Long Integer[2]	I	None	A 32-bit signed integer. Contains whole numbers (nonfractional) with complete accuracy in the range 2147483647 to –2147483647 (plus and minus 2 to the 31st). Long Integer fields require more space to store than Short fields.
BCD[3]	#	0–32 (number of digits after the decimal point)	Contains numeric data in a BCD (Binary Coded Decimal) format. The BCD field type is provided primarily for compatibility with other applications that use BCD data. Use BCD fields when you want to perform calculations with a higher level of precision than that available with the use of other numeric fields. Calculations on BCD fields are not performed as quickly as those on other numeric fields.
Date	D	None	Contains any valid date from January 1, 9999 B.C. to December 31, 9999.[4] Database Desktop correctly handles leap years and leap centuries and checks all dates for validity.
Time[2]	T	None	Contains times of day, stored in milliseconds since midnight, and limited to 24 hours.
Timestamp[2]	@	None	Contains both date and time values. To enter today's date and the current time, press Spacebar repeatedly until Database Desktop enters the data. Rules for this field type are the same as those for date fields and time fields.
Memo[5]	M	1–240 [1]	Memo fields can contain any printable character and can be virtually any length. The size value you assign refers to the amount of the memo Paradox stores in the table. This can be from 1 to 240 characters. Paradox stores the entire memo outside the table (in the .MB file). Paradox retrieves the data from the .MB file as you scroll through the records of the table. The amount of data a memo field contains is limited only by the disk space available on your system.

Table 3.1 Paradox field types (continued)

Field type	Symbol	Size values	Description
Formatted Memo[5]	F	0–240[1]	Formatted memo fields are similar to memo fields except you can format their text. Paradox recognizes and stores text attributes (different typefaces, styles, colors, and sizes) as well as formatting preferences (such as tabs, line returns, and justification) and stores them with the data.
Graphic[6]	G	0–240 (optional)	Contains graphics (pictures). You can create graphics in a painting or drawing application, or scan existing images and use them as values in a graphic field.
OLE[6]	O	0–240 (optional)	Contains objects placed in your table from other Windows applications that support OLE (Object Linking and Embedding) as a server. The advantage of using the OLE field type to place data is that, through OLE, you can access and make changes to the OLE object from within Paradox.
Logical[2]	L	None	Contains values representing true or false (yes or no). By default, valid entries include T and F (case is not important).
Autoincrement[2]	±	None	Contains long integer values in a read-only (non-editable) field. Paradox begins with the number 1 and adds one number for each record in the table. A table can contain only one autoincrement field.
Binary[6]	B	0–240 (optional)	Contains data that Paradox can't interpret. A common use of a binary field is to store sound. Paradox cannot display or interpret binary fields, but ObjectPAL can access them. Binary fields should be used only by Paradox application developers and advanced users.
Bytes[2]	Y	1–255	Contains data that Paradox can't read or interpret. A common use of a bytes field is to store bar codes or magnetic strips. Paradox cannot display bytes fields, but can access them. Bytes fields should be used only by application developers and advanced users. Unlike binary fields, bytes fields are stored in the Paradox table (rather than in the .MB file), allowing for faster access.

1. In data entry, users would enter the value 10^{-307} in scientific notation by typing `1e-307`; they would enter the value 10^{308} by typing `1e308`. By default, the format of values in scientific notation is established by the Windows Control Panel and the displayed value is a rounded-off version of the actual stored value.
2. Available only in Paradox 5.0 tables.
3. Available only in Paradox 5.0 tables. Database Desktop correctly interprets BCD data from other applications that use the BCD type. However, when Database Desktop performs calculations on BCD data, it converts the data to a numeric float type, then converts the result back to BCD.
4. All B.C. years are treated as leap years. You need Paradox for Windows to change format so B.C. dates display correctly.
5. Not available in Paradox 3.5 tables. Memo and formatted memo fields can be virtually any length. The size value you specify in the Create Table dialog box refers to the amount of the memo Paradox stores in the table. This can be from 1 to 240 characters. The whole memo is stored outside the table. For example, if you assign a size value of 45 to the field, Paradox stores the first 45 characters in the table. It stores the whole memo field in another file (with the extension .MB) and retrieves it as you scroll through the records of the table.
6. Not available in Paradox 3.5 tables.

Tip If all your memos are smaller than a given size (for example, 200 characters), you can save space and time by setting the memo field size to be equal to or larger than this given size. You'll still have an .MB file, but Paradox won't have to access it to display the field's data.

BLOB fields

Paradox memo, formatted memo, graphic, OLE, and binary field types can contain binary large object files (BLOB files). Certain rules apply to this group as a whole, and these field types are sometimes discussed collectively as BLOB fields. You can create these fields using Database Desktop, but to enter or edit data in them you need Paradox for Windows.

Valid dBASE field types

The valid dBASE field types and sizes are given in Table 3.2.

Table 3.2 dBASE field types

Field type	Symbol	Size	Decimal point	
Character	C	1–254 (required)	None	Contains any printable character (including blank spaces).
Float [1]	F	1–20 (required)	0–18, and <=Size –2[2]	Contains numeric data in a binary floating-point format. Use the float type on fields that will not require precise calculations to be performed on them; some degree of precision is rounded or truncated during calculation. Float number fields are best used to contain whole numbers, or numbers of up to two decimal places.
Number	N	1–20 (required)	0–18, and <= Size –2[2]	Contains numeric data in a Binary Coded Decimal (BCD) format. Use number fields when you'll need to perform precise calculations on the field data. Calculations on number fields are performed more slowly, but with greater precision than on float fields.
Date	D	None	None	Contains any valid date.
Logical	L	None	None	Contains a single character representing True or False (Yes or No) values. In dBASE logical fields, logical true can be entered as T, t, Y, or y. Logical false can be entered as F, f, N, or n.
Memo	M[3]	None	None	Contains blocks of text that are too large to be stored in a character field. The contents of memo fields are stored externally to the table. You do not specify a field size for dBASE memo fields.
OLE[4]	O	None	None	Contains objects placed in your table from other Windows applications that support OLE (Object Linking and Embedding) as a server. The advantage of using the OLE field type to place data is that, through OLE, you can access and make changes to the OLE object from within dBASE.
Binary[4]	B	None	None	Used to store binary data such as sound or graphics.

1. Float fields are not available in dBASE III+ tables.
2. You set the number of decimal places in the Dec column of the Field Roster in the Create/Restructure dialog box. In the Dec column, you specify how many decimal places to store. Enter a number at least 2 less than the field size. This is because Paradox counts the decimal point and sign (if any) as part of the field size.
3. Memo field formats differ between dBASE III+ and dBASE IV tables.
4. Available only in dBASE V tables.

Note Remember, you can create dBASE tables in Database Desktop with many sophisticated field types, but you cannot view or edit data in Memo, OLE, or Binary fields using Database Desktop. To enter or edit data in these field types, use dBASE.

Valid SQL field types

For information on valid field types for your SQL server, search for "field types" in the keyword list in Database Desktop Help, and choose the topic for your server.

Keys in Paradox tables

A Paradox table's key establishes the primary index and sort order for the table. A key also requires each value in the field(s) that defines the key to be unique. For example, if the Cust ID field is identified as the key of the *Customer* table, each value in the Cust ID field must be unique. Likewise, if the Cust, Date, and Item # fields are identified as the key of the *Bookord* table, the field values (taken as an ordered group) must be unique. This guards against duplication of data within the table.

Keys are required for linking tables and for using the data integrity features of Paradox tables.

dBASE and SQL tables use indexes similarly to the way Paradox uses keys. For more information on dBASE indexes, search for "dBASE indexes" in the keyword list in Database Desktop Help. For more information on SQL tables, see the Database Desktop Help Contents.

Defining keys

To create a key, display the Create Table (page 19) or the Restructure Table (page 48) dialog box. Then move to the Key column in the Field Roster and double-click (or press any key). The key field indicator (*) appears. Database Desktop keys the table on the selected field.

Remember these rules when defining keys:

- A table can have only one primary key. This key can be made up of one or more fields.
- If a key is defined as a single field, that field must be the first field in the Field Roster.
- If you identify more than one field as keyed, you create a *composite* key. These fields, taken as a group, must be unique for each record of the table. The composite key must be the first fields in the Field Roster.

Tip You can always move a field to a different position in the Field Roster to create the field arrangement you want. See page 21.

Removing keys

To remove a key from a field or group of fields, move to the Key column in the Field Roster and double-click (or press any key). Database Desktop removes the key, and the key field indicator (*) disappears.

If you remove a key that is located above other keyed fields, you must rearrange the fields so the roster begins with the keyed fields.

A dBASE table's index

When working with dBASE tables, Database Desktop uses an index to organize the records in a table according to the values in one or more fields.

When you create an index on a dBASE table, a file is created that contains the indexed field's values and their corresponding record numbers. Database Desktop refers to the index file when locating and displaying the records in a table.

Although Database Desktop supports both .MDX files and .NDX files, it is recommended that you use a dBASE production index (the .MDX file which uses the table's name as its file name) whenever possible. Although you can create nonproduction .MDX files as well as .NDX files, Database Desktop automatically maintains the production index.

A SQL table's index

SQL tables use unique and non-unique indexes, but they do not use the primary keys that Paradox tables use. You can create multiple indexes for a SQL table; for each index, you specify whether it is unique or non-unique. SQL indexes, unlike Paradox and dBASE indexes, are always maintained.

You can use Database Desktop to create and modify indexes on SQL tables, but you cannot specify which index to use in Database Desktop.

When you use a SQL table in Database Desktop, the table should have a unique index. If it does not have a unique index and you edit the table's data, you cannot view the edits as you are making them. To add a unique index to an existing table, choose Utilities | Restructure.

For more information on SQL indexes, see "Creating indexes on SQL tables" on page 45.

Specifying validity checks

Validity checks are rules that govern the values you can enter in a field. If you want Database Desktop to make sure that the values entered in a field are valid, specify a validity check for that field.

Note Validity checks work only on Paradox tables, not on dBASE tables. For SQL tables, the only validity check you can specify is whether a field is required.

The five types of validity checks are listed in Table 3.3.

Table 3.3 Paradox validity checks

Validity check	Meaning
Required field	Every record in the table must have a value in this field. SQL tables can also use this validity check.
Minimum	The values entered in this field must be equal to or greater than the minimum you specify here.
Maximum	The values entered in this field must be less than or equal to the maximum you specify here.
Default	The value you specify here is automatically entered in this field. You can replace it with another value.
Picture	You specify a character string that acts as a template for the values that can be entered in this field. The values entered in this field are automatically formatted according to this picture.

To place a validity check on a field,

1. Select the field in the Field Roster in the Create Table (page 19) or the Restructure Table (page 48) dialog box. The field must have a name, type, and size (if size is required for the field type).
2. Choose Validity Checks from the Table Properties list. All possible validity checks appear in the Create Table dialog box.
3. Type values as necessary to specify the type of validity check you want on the field. You can have any combination of validity checks on a field.

For information specific to SQL tables, see "Creating a SQL table" on page 44.

Picture patterns

A picture acts as a template that formats the value you enter in a field. For example, if you specify the picture **(###)###-####** (a common template for U.S. phone numbers) and enter the value `4085551234`, Database Desktop formats the value into (408)555-1234.

Table 3.4 describes the characters you can use in a picture.

Table 3.4 Picture pattern characters

Character	Stands for
#	Any numeric digit
?	Any letter (case-insensitive)
&	Any letter (convert to uppercase)
@	Any character (case-insensitive)
!	Any character (convert to uppercase)
;	The next character is literal, not a special picture-string character
*	The next character can be repeated any number of times. Or specify how many occurrences of the next letter to require.
[]	Characters inside brackets are grouped
{}	Characters inside braces are grouped
,	Alternative values

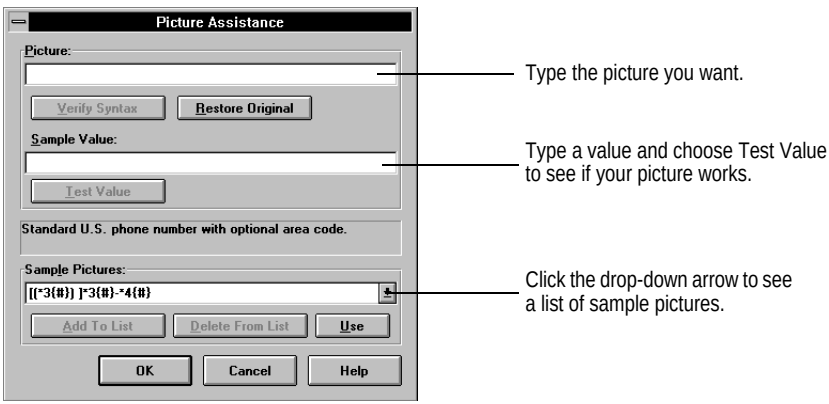
If you use any printable (visible) character in a picture string different from those listed in Table 3.4, Database Desktop treats it as a constant. When you enter a value in a field that has a picture validity check, and you come to a point at which a constant is specified, Database Desktop automatically enters the constant. For example, if you create the picture (408)###-#### and then type `5551234` in the field, Database Desktop inserts (408)555-1234 in the table.

To specify a picture for a field, either type the picture pattern you want in the Picture edit field or choose Assist.

Getting assistance with pictures

When you choose Assist in the Create Table (page 19) or Restructure Table (page 48) dialog box, you'll see the Picture Assistance dialog box, shown in Figure 3.2.

Figure 3.2 The Picture Assistance dialog box



When you use the Picture Assistance dialog box, you can

- Type a custom picture pattern in the Picture edit field.
- Choose Verify Syntax to test the picture you type in the Picture edit field.
- Choose Restore Original to undo any changes you've made to the contents of the Picture edit field.
- Type a value in the Sample Value edit field and choose Test Value to verify that the picture in the Picture edit field works correctly.
- Click the Sample Pictures drop-down arrow to view the sample pictures. Database Desktop displays an explanation of the picture in the panel above the Sample Pictures drop-down list.
- Choose Add To List to place the contents of the Picture edit field on the Sample Pictures drop-down list.
- Choose Delete From List to remove the selected picture from the Sample Pictures drop-down list.

- Choose Use to copy the selected sample picture to the Pictures edit field, where you can modify it.

You can get assistance with pictures you create or with standard pictures Database Desktop provides.

To enter your own picture,

1. Use characters and the symbols shown in Table 3.4 to type the picture you want in the Picture edit field.
2. Choose Verify Syntax to ensure that Database Desktop can interpret the picture.
3. If the syntax is correct, you'll see a message displayed in the message area of the Picture Assistance dialog box confirming that the picture is correct.
4. Choose OK to use the picture and close the dialog box.

To use one of the sample pictures,

1. Choose a picture from the Sample Pictures drop-down list. You'll see an explanation of the picture in the message area of the dialog box. For example, if you choose the picture 5#[-4#], you'll see a message telling you that this picture is for either a 5-digit or a 9-digit US zip code.
2. Choose Use to copy the sample to the Picture edit field.

   If you want, you can modify the standard template when it's in the Picture edit field. If you make a mistake, choose Restore Original to return to the standard template you copied to the Picture edit field.
3. When the picture you want is in the Picture edit field, choose OK.

Note If you create a picture validity check when restructuring a table that contains data, Database Desktop does not reformat existing data to match the picture.

Viewing a field's validity checks

Database Desktop displays each field's existing validity checks whenever you select the field in the Field Roster in the Create Table (page 19) or Restructure Table (page 48) dialog box. For example, if you've specified a default value for the State/Prov field, select State/Prov in the Field Roster to view its validity checks.

Tip You can always view the validity checks on a table from the Structure Information dialog box. Choose Utilities | Info Structure to see all the information about a table's structure.

Removing a validity check

You can remove validity checks using either the Create Table (page 19) or Restructure Table (page 48) dialog box.

Removing a validity check doesn't change any data that exists in the field. It simply removes validity restrictions on future data you enter. To remove a field's validity check,

1. Select the field in the Field Roster.

2 Remove the value from the validity check's edit field. (To clear the Required Field validity check, uncheck its check box.)

SQL You cannot remove the Required validity check from a field on a SQL table.

Creating table lookup

Table lookup helps you enter data in one Paradox table that already exists in the first field of another Paradox table—the *lookup* table. Table lookup lets you

- Require that the values you enter into a field exist in the first field of another table
- Refer to another table to look up the acceptable values for a field
- Copy values in the lookup table to the table you're editing

The difference between table lookup and referential integrity

Table lookup is primarily a data entry tool. Unlike referential integrity, it doesn't track or control changes you make to the lookup table. Table lookup ensures that data is copied accurately from one table to another; referential integrity ensures that the ties between data in separate tables cannot be broken. For more about referential integrity, see page 36.

Why use table lookup?

The major advantage of table lookup is its ability to automatically enter correct values in your table. Table lookup is also valuable when you need to use Paradox 3.5 tables, because referential integrity is unavailable to them.

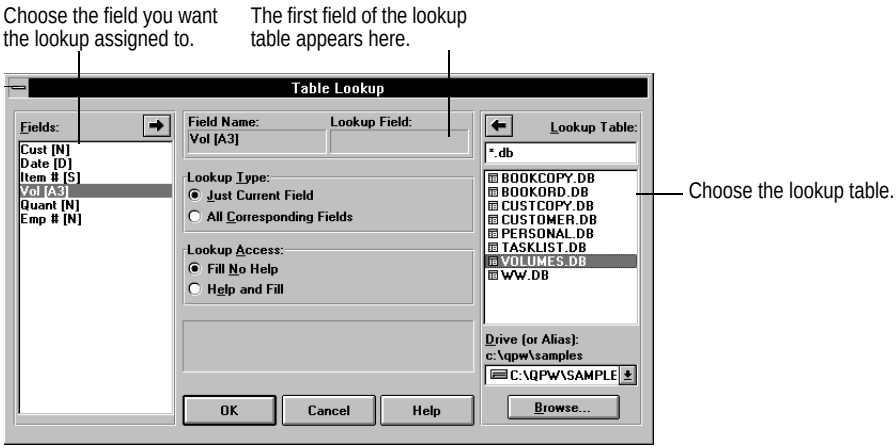
Keep these rules in mind when setting up a lookup table:

- The lookup table contains data you want to copy to another table. That data must be in the lookup table's first field.
- The field that you're assigning the table lookup to must be the same field type as the first field of the lookup table.
- For best performance, the lookup table should be keyed. See "Keys in Paradox tables" on page 26.

To specify a lookup table for a field,

1 Choose Table Lookup from the Table Properties list in the Create Table (page 19) or Restructure Table (page 48) dialog box. The Define button becomes available, and all existing table lookups are listed below it.

2 Choose Define to open the Table Lookup dialog box.



3 Choose the lookup table from the Lookup Table list. (All tables in the working directory are shown.) The first field of the table you choose appears in the Lookup Field edit field.

4 From the Fields list, choose the field that will look to the lookup table. Database Desktop places it in the Field Name edit field.

5 Choose the lookup options you want.

6 Choose OK to close the Table Lookup dialog box. The name of the lookup table appears below the Define button in the Create Table (or Restructure Table) dialog box.

Note You can use a table lookup across different directories. Use the Path drop-down list or the Browse button.

Lookup options

A Paradox table can be assigned one of two types of table lookups, as follows:

- **Just Current Field** checks values you enter in the current field against the values in the first field of the lookup table. If the value is invalid, an error message appears and the value is *not* entered into the table.
- **All Corresponding Fields** does what Just Current Field does, and also fills in values from corresponding fields in the lookup table. Corresponding fields must have *identical field names* and compatible field types in both tables.

Each type of table lookup has two options, as follows:

- **Fill No Help** keeps the lookup table from appearing, so you can't display the lookup table and select an entry from it.
- **Help and Fill** lets you view the lookup table from the table you're editing, by pressing *Ctrl+Spacebar* or choosing Record | Lookup Help.

Tip When the lookup access is Fill No Help, you can't open the lookup table automatically. You can, however, view the lookup table by opening it in its own Table window.

The ways that lookup types can be combined with options are summarized in the following table.

Table 3.5 Lookup types and options

Lookup type	Option	Description
Just Current Field	Fill No Help	When you enter valid data (that exists in the lookup table) into a field of the table you're editing, the data is accepted in that field of the table. Otherwise, an error message appears. The lookup table is protected and *cannot* be viewed during editing.
Just Current Field	Help and Fill	When the pointer is in a lookup field, a message in the status line tells you what keys to press to view the lookup table. To view the values in the lookup table, press *Ctrl+Spacebar*. When the lookup table appears, a scroll lock is placed to the right of the lookup field. As you browse the fields in the lookup table, the values in the first field remain onscreen. To copy a value from the lookup table, select it, then press *Ctrl+Spacebar*. Or, type the value into the field.
All Corresponding Fields	Fill No Help	Same as the Just Current Field type with the Fill No Help option above, except that all values from fields in the lookup table that have the same field name and type as fields in the table you're editing are copied to the table you're editing (instead of a single value).
All Corresponding Fields	Help and Fill	You can enter data into a field by typing it in, but no corresponding values will be filled in. To display the lookup table, press *Ctrl+Spacebar*. When you choose the value you want, it and all corresponding field values are copied from the lookup table to the table you're editing.

Defining secondary indexes

A secondary index lets database users

- Locate values in the specified fields
- Provide a second view order for the table
- Link tables

You can create secondary indexes in Paradox tables in Database Desktop, but you must have Paradox to use them.

To view the records of a keyed table in a different order in Paradox, you *must* use a secondary index. Only a secondary index can temporarily override the primary sort order established by a table's key.

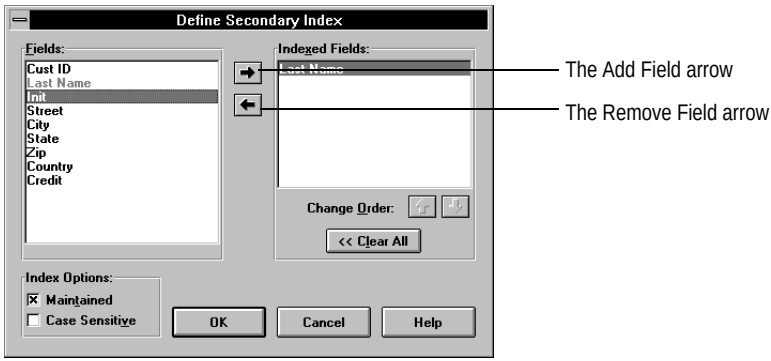
For example, if you sometimes want to view the *Customer* table by City, but need to keep the table's key intact, you can define a secondary index on the City field and use it to temporarily change the view order of the records in Paradox. When you use a secondary index in Paradox, you change only the view order of the records. The physical location of the records in the table does not change.

You can create as many single-field indexes as there are fields in a table, and up to 16 composite secondary indexes.

Note You can't create a secondary index on a Memo, Formatted Memo, Graphic, OLE, Logical, Binary, or Bytes field.

To create a secondary index for a table,

1. Choose Secondary Index from the Table Properties list of the Create Table (page 19) or Restructure Table (page 48) dialog box. The Define button becomes available and any existing secondary indexes are displayed.
2. Choose Define to open the Define Secondary Index dialog box.



   The Fields list displays the fields you can use as a secondary index. BLOB fields are dimmed.

3. Double-click the field on which you want to create the secondary index (or select it, and then choose the Add Field arrow or press *Alt+A*). Database Desktop moves the field to the Indexed Fields list.
4. Check any options you want. Index options are discussed later in this section.
5. Choose OK to create a secondary index on the field and close the dialog box.

Database Desktop automatically names case-sensitive indexes you create on a single field with the field's name. To create another secondary index, choose Define again.

The Maintained option

If you check the Maintained check box in the Define Secondary Index dialog box, Database Desktop maintains the secondary index automatically. This means that every time the table is changed, the index is also changed. The file that contains the table and the file that contains the index are never out of sync. This speeds up certain operations like queries. Also, using Paradox for Windows, you can link Paradox tables in form and report data models only on maintained indexes.

If you don't check Maintained, Database Desktop updates the index only when you use it (for example, when you run a query). The operation that uses the secondary index takes slightly longer using a non-maintained index, because Database Desktop must first update the index to recognize values that you've added, deleted, or changed, and then perform the requested operation.

Note You can create a maintained secondary index only on a keyed table. This option is dimmed if the table does not have a key specified.

The Case Sensitive option

If you check the Case Sensitive check box in the Define Secondary Index dialog box, Paradox recognizes the case (capitalization) of words as it sorts them. (Database Desktop does not sort on secondary indexes.)

Capitalizing a value doesn't make it unique in a case-insensitive index.

Note Some language drivers do not support a case-insensitive index.

Database Desktop automatically names single-field, case-sensitive indexes with the field's name. You must name a case-insensitive index when you save it. This enables you to create two indexes on the same field, one case sensitive and one case insensitive.

Composite secondary indexes

You can create a composite secondary index by adding more than one field to the Indexed Fields list. Open the Define Secondary Index dialog box (page 34) and add the fields you want to use in the index to the Indexed Fields list. To add a field, double-click it (or choose it in the Fields list, then choose the Add Field arrow or press *Alt+A*). Database Desktop adds the field below the selected field in the Indexed Fields list.

Paradox creates the composite index in the order that the fields appear in the Indexed Fields list. When you use this index, Paradox sorts the table by the top field first, then by the next, and so on.

You can change the order of the fields to change the sort order of the index. To move a field in the Indexed Fields list, select the field and use the Change Order arrows to move it up or down. These arrows become available when two or more fields are in the Indexed Fields list.

To remove one field from the Indexed Fields list, select it and choose the Remove Field arrow (or press *Alt+R*). To remove all fields from the Indexed Fields list, choose Clear All (or press *Alt+L*).

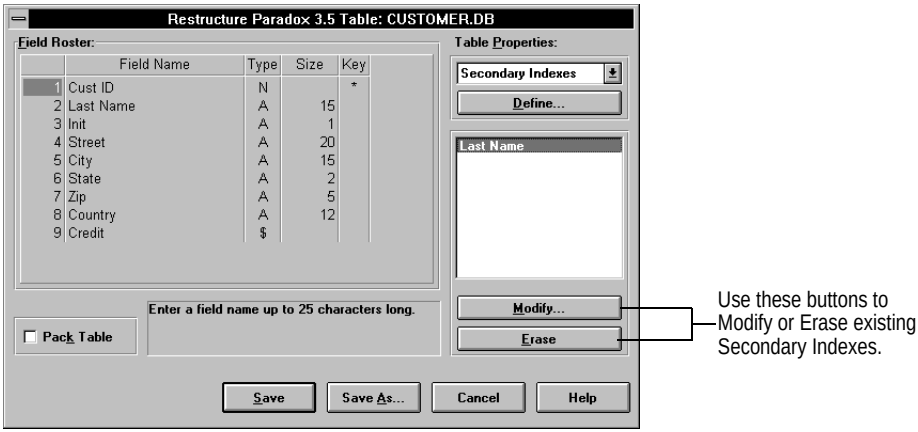
Choose OK to create and name a composite secondary index. You'll see the Save Index As dialog box. Paradox automatically names single-field, case-sensitive indexes with the field's name. For composite indexes or case-insensitive indexes, type the name you want in the Index Name edit field. A secondary index name can be up to 25 characters and include any printable character.

When you choose OK from the Save Index As dialog box, both it and the Define Secondary Index dialog box close and the name appears in the list of secondary indexes in the Create Table (or Restructure Table) dialog box. Database Desktop warns you when you might overwrite an existing index.

Modifying secondary indexes

To change a secondary index definition, select it from the list of secondary indexes in the Create Table (page 19) or Restructure Table (page 48) dialog box and choose Modify. The Define Secondary Index dialog box opens with the selected index specification filled in. Make the changes you want and choose OK.

Figure 3.3 Changing a secondary index



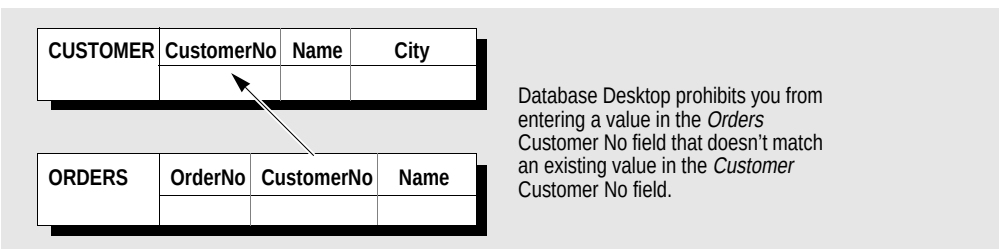
Erasing secondary indexes

If you want to remove a secondary index definition, select its name from the list of secondary indexes in the Create Table (or Restructure Table) dialog box and choose Erase. Database Desktop deletes the index.

Defining referential integrity

Referential integrity means that a field or group of fields in one Paradox table (the "child" table) must refer to the key of another Paradox table (the "parent" table). Database Desktop accepts only those values that exist in the parent table's key as valid values for the specified field(s) of the child table.

Figure 3.4 Referential integrity



You can establish referential integrity only between like fields that contain matching values. For example, you can establish referential integrity between *Customer* and *Orders* on their CustomerNo fields. In both cases, the values contained in the specified fields are the same. The field names don't matter as long as the field types and sizes are identical.

Note You can establish referential integrity only between tables in the same directory.

Using referential integrity, Database Desktop checks the validity of a value before accepting it in the referential integrity table. If you establish referential integrity between

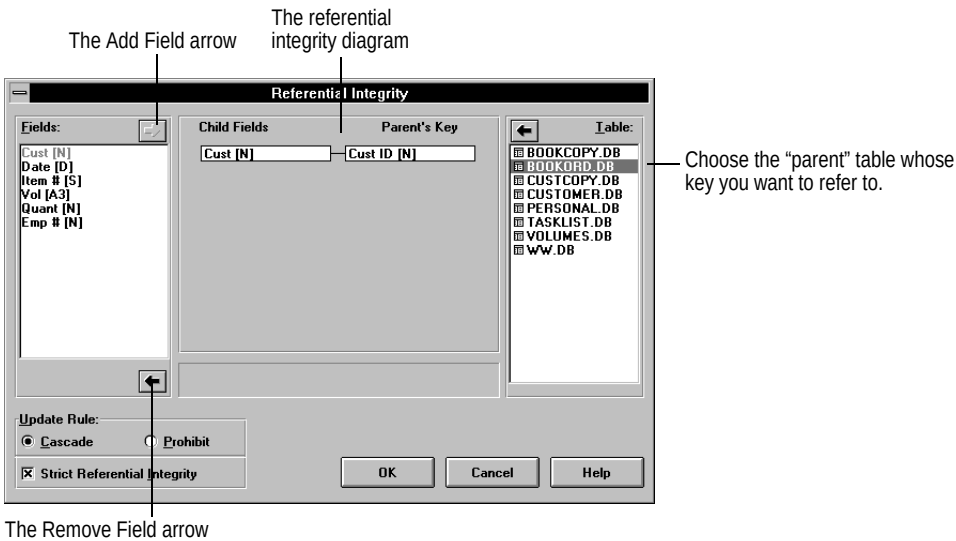
Customer and *Orders* on their CustomerNo fields, then enter a value in the CustomerNo field of *Orders*, Database Desktop searches the CustomerNo field of *Customer* and

- Accepts the value in *Orders* if it exists in *Customer*
- Rejects the value in *Orders* if it doesn't exist in *Customer*

Note If you define referential integrity on a table that already contains data, some existing values might not match a value in the parent's key field. When this happens, Database Desktop places the existing records that don't match into the temporary *Keyviol* table in your private directory.

To define a referential integrity relationship,

1. In the Create Table (page 19) or Restructure Table (page 48) dialog box, choose Referential Integrity from the Table Properties list. The Define button becomes available.
2. Choose Define to open the Referential Integrity dialog box.



Database Desktop displays all tables in the working directory in the Table list.

3. Choose the parent table from the Table list. The table's key field appears in the Parent's Key area of the referential integrity diagram.

Note If the parent table you choose isn't keyed, or if there is a problem with the key, a message appears on the status bar.

The Fields list displays all the fields from the child table. (BLOB and autoincrement fields are dimmed in the Fields list. You cannot create referential integrity on these fields.)

4. Double-click the child table's field in the Fields list (or *Tab* to it and click the Add Field arrow or press *Alt+A*). The field name appears in the Child Fields area of the referential integrity diagram.

If you choose a field that isn't of the identical type as the parent's key field, Database Desktop displays a message on the status bar, and doesn't add the field to the diagram.

Tip If you make a mistake and add the wrong field, click the Remove Field arrow or press *Alt+R*.

5 Choose the update rule you want.

6 Choose whether you want to enforce strict referential integrity. (See "Using Strict Referential Integrity" later in this section.)

7 Choose OK to name and save the referential integrity relationship.

You can establish referential integrity with a composite key. If the parent table has a composite key, add fields from the Fields list to match one, some, or all of the parent's key fields.

Update Rule options

Database Desktop provides two update rules for tables that use referential integrity. You must use one of these rules when you define referential integrity.

- **Cascade.** Any change you make to the value in the key of the parent table is automatically made in the child table. Cascade is the default update rule.

Note To cascade an update across tables, Database Desktop must place a lock on the parent record and all its child records. If the lock is denied (because another user has already placed a lock), Database Desktop can't perform the cascaded update.

- **Prohibit.** You *cannot* change a value in the parent's key if there are records that match the value in the child table. For example, if the value 1356 exists in the CustomerNo field of *Orders*, Database Desktop prohibits you from changing that value in the CustomerNo field of *Customer*. (You can change it in *Customer* only if you first delete or change all records in *Orders* that contain it.) If, however, the value *doesn't* exist in any records of the child table, Database Desktop permits the change in the parent table.

Using Strict Referential Integrity

The Strict Referential Integrity option lets you control how DOS versions of Paradox access a table on which you've defined referential integrity.

Suppose you use a DOS version of Paradox to open a Paradox for Windows table that uses referential integrity. You could add data that violates the referential integrity, because the version of Paradox you're using doesn't recognize the referential integrity. To prevent DOS versions of Paradox from opening the table, check the Strict Referential Integrity check box in the Referential Integrity dialog box.

Saving the referential integrity relationship

When you've established the referential integrity you want, choose OK to name and save it. You'll see the Save Referential Integrity As dialog box.

Type the name you want to give the referential integrity relationship. Referential integrity names can be up to 31 printable characters and require no file extension. When you choose OK, the Referential Integrity dialog box closes, and the referential integrity name appears in the list area below the Define button in the Create Table (page 19) or Restructure Table (page 48) dialog box.

Note Database Desktop saves referential integrity definitions in a file with the table's name and the .VAL file extension when you save the table's structure.

When you save the referential integrity, Database Desktop checks to see if there is an index on the referential integrity field(s). If there isn't an index, Database Desktop creates one for you, giving it the name of the field (if it's a single-field case-sensitive definition) or the name you gave the referential integrity (if it's a multiple-field case-insensitive definition). You'll see the index appear in the list of secondary indexes when you choose Secondary Indexes from the Table Properties list in the Create Table (page 19) or Restructure Table (page 48) dialog box. If you delete the referential integrity, Database Desktop does *not* automatically delete this index. You must delete it manually.

Changing or deleting referential integrity

You can choose any referential integrity name from the list of named referential integrity relationships in the Create Table (page 19) or Restructure Table (page 48) dialog box to either modify or erase it.

- Choose Modify to open the Referential Integrity dialog box with the selected referential integrity relationship filled in. Database Desktop must obtain locks on all tables involved in the referential integrity when you modify it. You can change
  - The name of the referential integrity (save the referential integrity with a different name)
  - The update rule
  - The Strict Referential Integrity setting
- Choose Erase to delete the selected referential integrity relationship.

Creating self-referential integrity

You can create referential integrity on a table so that one field refers to the table's key field. For example, suppose you have a table that lists employees. The key field of this table is Employee ID. There is also a Supervisor field. The supervisors are also employees. You can create referential integrity to make sure that the value you enter in Supervisor is also a value in Employee ID.

When you create self-referential integrity, you must use the Prohibit update rule.

Note You can't create a circular reference. That is, you can't create referential integrity in which a field refers to itself.

Establishing password security

Sometimes it's important to ensure that the Paradox table you create is protected from access by unauthorized users. Not only can you establish a password for a Paradox table as a whole, but you can also assign specific rights to the table or individual fields.

Once you specify password security, only those users who know the password can access the table. This includes you, so don't forget your password!

Whenever a user tries to access a password-protected table, Paradox prompts them to supply the password (if they haven't already done so).

To create a master password for the table,

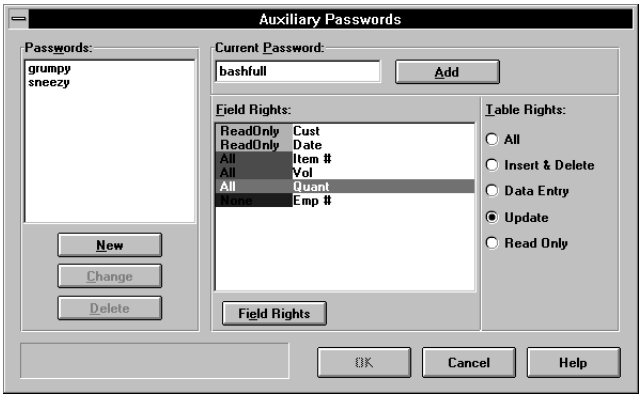
1. Choose Password Security from the Table Properties list in the Create Table (page 19) or Restructure Table (page 48) dialog box. The Define button becomes available.
2. Choose Define to display the Password Security dialog box.

3. Type the password you want in the Master Password edit field. You'll see asterisks (*) representing the characters you type. A password can be from 1 to 31 characters long and can contain spaces. Passwords are case sensitive.
4. Type the same password in the Verify Master Password edit field. Again, you'll see asterisks in place of the characters you type.
5. If the two passwords aren't identical (including capitalization), you'll see an error message prompting you to enter the password again.
6. Choose OK to close the dialog box and return to the Create Table dialog box.

Database Desktop saves the password when you choose OK from either the Password Security or the Auxiliary Passwords dialog box.

If you want more specific security, you can choose Auxiliary Passwords from the Password Security dialog box. You'll see the Auxiliary Passwords dialog box, shown in Figure 3.5.

Figure 3.5 The Auxiliary Passwords dialog box



Use this dialog box to assign specific table and field rights to specific auxiliary passwords.

Note You can return to the Password Security dialog box and change the table's master password by choosing Cancel or pressing *Esc*. If you do this, any auxiliary passwords you've specified are lost.

Table rights

The master password provides all rights to the table and its fields. Using auxiliary passwords, you can be more selective and specific about the kind of table operations each user can perform.

- **All** gives a user all rights to any function of the table, including the ability to restructure or delete it. The only operation prohibited is a change to the master password.
- **Insert & Delete** gives a user the right to insert or delete records or empty the table, but not to delete the table.
- **Data Entry** gives a user the right to insert records in the table, but not to delete records, restructure, or empty the table.
- **Update** gives a user the right to view the table and change non-key fields, but not to insert or delete records or change key fields.
- **Read Only** gives a user the right to view the table, but not to change it in any way.

Field rights

In addition to specifying rights for the table as a whole, you can assign rights to individual fields. The default option in the Field Rights list is All. To choose another option, double-click the field (or choose the Field Rights button).

Double-click once to choose Read Only. Double-click again to choose None. If you double-click again, the field right is reset to All.

- Choose All to give the user all rights to the data in that field (within the limits of the table rights you specify).
- Choose Read Only to give the user the right to view—but not to change—the data in that field.
- Choose None to prevent the user from viewing or changing the data in that field. Database Desktop hides the values in the field.

Table 3.6 shows how you can combine table rights and field rights.

Table 3.6 Auxiliary password field and table rights

	Field rights		
Table rights	**All**	**Read Only**	**None**
All	✓		
Insert & Delete	✓		
Data Entry	✓	✓	✓
Update	✓	✓	✓
Read Only	✓	✓	✓

To specify an auxiliary password,

1. Type the password in the Current Password edit field.
2. Choose the level of table rights for the password from the Table Rights panel.
3. Assign the field rights (All, Read Only, or None) for the password.
4. Choose Add to place the password in the Passwords list.
5. Repeat the process to specify as many auxiliary passwords as you need.
6. Choose OK to save the auxiliary passwords and close the dialog box. (This also saves the master password.)

Remove a password by selecting it in the Passwords list and choosing Delete. Change the rights of a password by selecting it from the Passwords list and choosing Change. Make the changes you want, then choose Accept to save them, or choose Revert to return the password to the list unchanged.

Using passwords

When you try to open a password-protected table, Database Desktop prompts you for the password. You must enter the password to open the table.

Suppose you close the table, then attempt to open it again. If you haven't exited Database Desktop, you'll be allowed to open the table *without* giving the password another time. Database Desktop stores the password and assumes you're allowed to open the table again. Database Desktop releases all passwords when you exit the program.

If you want to release a password without exiting Database Desktop, choose Utilities | Passwords. You'll see the Enter Password(s) dialog box, shown in Figure 3.6.

Figure 3.6 The Enter Password(s) dialog box

You don't create passwords from this dialog box; you allow or remove access to a password-protected table.

Enter the password you want Database Desktop to release from memory in the Password text box. Asterisks (*) represent the characters you type. Choose Remove to remove this password from Database Desktop memory. You'll be required to supply the password the next time you open the table.

Tip You can choose Remove All to remove all passwords from Database Desktop memory. This means any table you've opened using a password, then closed, will again be protected. (Tables that are still open are not affected.)

If you've assigned the same password to several tables, you can use the Enter Password(s) dialog box to give Database Desktop the password once to access all applicable tables. Type the password and choose Add or OK (or press *Enter*).

Choosing a table language

A table's language driver determines the table's sort order and available character set. You choose a default language driver for Paradox and dBASE tables from the BDE Configuration Utility. (Refer to the BDE Configuration Utility Help system for more information.)

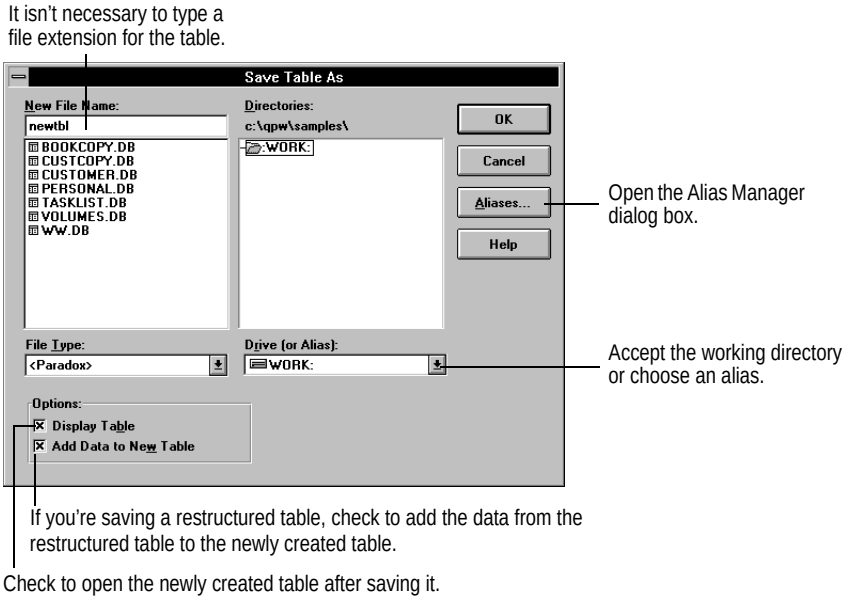
You can override the default table language when you create a new table by choosing Table Language from the Table Properties drop-down list in the Create Table (page 19) or Restructure Table (page 48) dialog box. Choose Modify to change the default table language. You'll see the Table Language dialog box. Choose a different language from the Language drop-down list.

Saving the new table

When you finish specifying your table's structure, choose Save As in the Create Table (page 19) or Restructure Table (page 48) dialog box to create the table and close the Create Table dialog box. If you didn't complete all the necessary steps or if you've entered conflicting or invalid specifications, Database Desktop prompts you to correct the problem(s).

When you choose Save As, Database Desktop displays the Save Table As dialog box.

Figure 3.7 The Save Table As dialog box



Type the table's name in the New File Name edit field. You don't need to type an extension; Database Desktop recognizes the type of file you want based on the table type you chose in the Table Type dialog box.

Unless you specify otherwise, Database Desktop saves the table to the working directory. Use the Drive drop-down list if you want to save the table to an aliased directory. If you want to save the table to a directory that isn't available from the Drive drop-down list, type the full path you want, with the table name, in the New File Name edit field.

If you check the Display Table check box, Database Desktop opens the new table when the Create Table dialog box is closed.

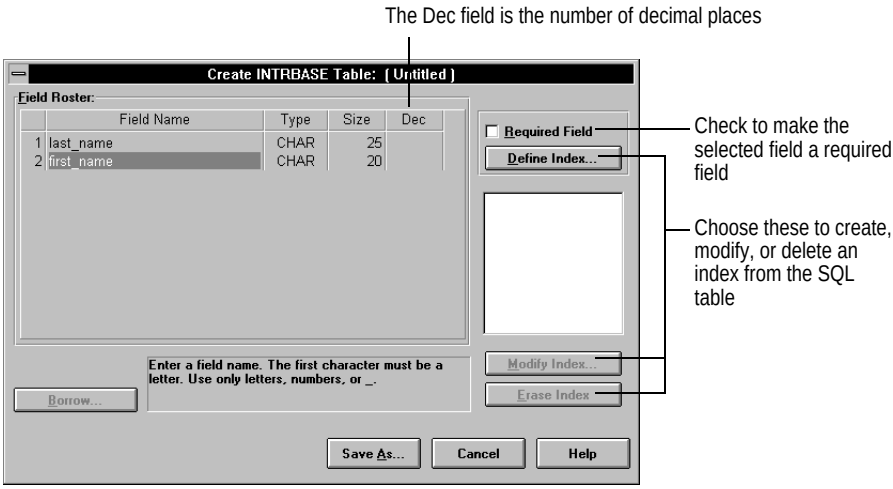
Creating a SQL table

When you create a SQL table,

- You specify the driver type in the Table Type dialog box after choosing File | New | Table.
- You can define the table structure (fields & types), specify required fields, and define indexes. Other features of Paradox tables, such as validity checks and referential integrity, are not supported on SQL tables.

The Create Table dialog box for SQL tables looks as shown in Figure 3.8.

Figure 3.8 The Create Table dialog box for SQL tables



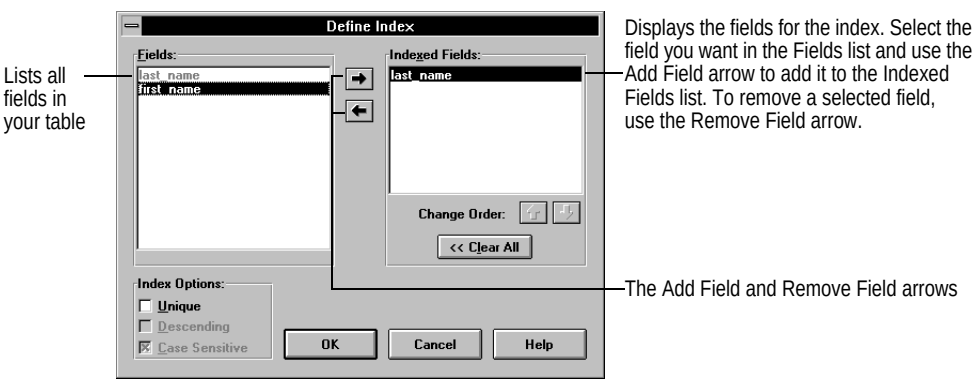
For information on valid field types for your SQL server, search for "field types" in the keyword list in Database Desktop Help, and choose the topic for your server.

Creating indexes on SQL tables

You can use Database Desktop to create and modify indexes on SQL tables.

To create an index for a SQL table, display the Create Table (page 45) or the Restructure Table (page 54) dialog box. Then, choose Define Index. Database Desktop displays the Define Index dialog box, shown in Figure 3.9.

Figure 3.9 The Define Index dialog box for SQL indexes



For SQL servers that do not support case-insensitive or descending indexes; the Descending and Case Sensitive fields are dimmed.

When you use a SQL table in Database Desktop, the table should have a unique index. If it does not have a unique index and you edit the table's data, you cannot view the edits as you are making them.

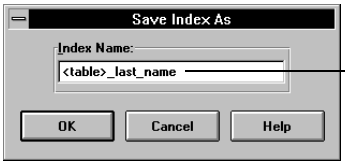
For detailed information on using the Define Index dialog box, search for "Define Index Dialog Box" in the keyword list in Database Desktop Help, and choose the topic "Define Index Dialog Box (SQL Tables)".

Naming SQL indexes

For most database servers, index names must be unique for all tables in database (or in some other predefined workspace). When you create an index on a SQL table, Database Desktop prefixes the index name with the table name to ensure that the index name is unique.

Sybase note Sybase index names do not need to be unique within a database, so Database Desktop does not prefix Sybase index names with table names.

When you create a SQL index and choose OK from the Define Index dialog box, Database Desktop supplies the prefix "<table>_" for the index name as follows:



This index on the *Customer* table will be named "customer_last_name"

You can include the table name with the index name or omit it:

- If you type the index name following "<table>_", Database Desktop prefixes the index name with the table name and an underscore.
- If you delete "<table>_", Database Desktop omits the table name from the index name. If the index name is not unique, an error will occur when Database Desktop saves the table.

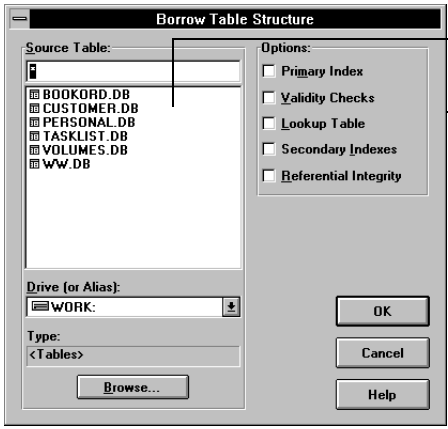
This index naming scheme also affects restructuring, as described in "Restructuring a SQL table" on page 53.

Borrowing a table structure

When creating a table similar to one you already have, you can borrow its structure. Then you can either use it as is or change it. You must begin from a blank table structure to borrow another table's structure.

To borrow a table structure,

1 Choose Borrow from the Create Table dialog box (page 19). You'll see the Borrow Table Structure dialog box.



Choose the table whose structure you want to borrow.

For Paradox tables, you can borrow these properties. For other table types, you can borrow indexes.

2 Choose the table whose structure you want to borrow.

3 Choose the options (if any) you want to borrow along with the table.

4 Choose OK to return to the Create Table dialog box. The borrowed table's structure appears in the Field Roster. You can now change the borrowed structure.

Borrow options

In addition to borrowing the structure of a table, you can also borrow its primary or secondary indexes, validity check definitions, referential integrity, table lookup definitions, or any combination of these options. Use the Options panel to check the definitions you want to borrow with the table.

If you are borrowing from a table of a different driver type, Database Desktop borrows the indexes when possible.

Database Desktop borrows validity checks into Paradox tables when possible. For example, if you borrow from a SQL table into a Paradox table and the SQL table has a required field, Database Desktop borrows the required validity check into the Paradox table.

Restructuring Paradox tables

If, sometime after creating a table, you need to change its structure, you can restructure it. Restructuring a table is very much like creating it for the first time.

This section discusses restructuring Paradox tables.

- For information on restructuring dBASE tables, search for "Restructure Table Dialog Box" in the keyword list in Database Desktop Help.

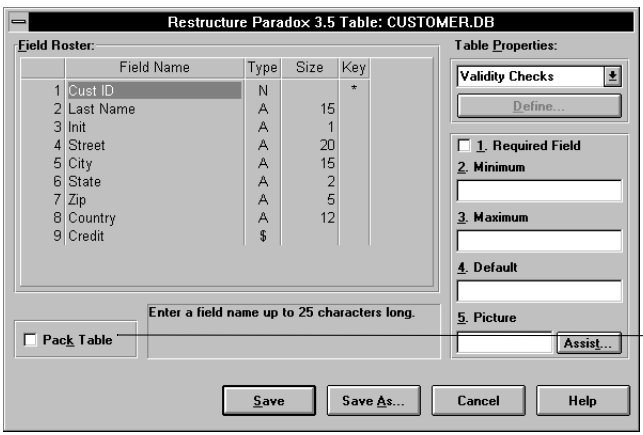
- For information on restructuring SQL tables, see "Restructuring a SQL table" on page 53.

Tip If you want to just rename a table (not restructure it), use the Rename utility. See "Renaming objects" on page 64.

To restructure a table, choose Utilities | Restructure, then choose the table you want. (If the table you want to restructure is already open in the active window, use Table | Restructure.) The Restructure Table dialog box opens, with the table's existing type and structure specified.

The following figure shows the *Customer* table in the Restructure Table dialog box.

Figure 3.10 The Restructure table dialog box



Work in the Restructure Table dialog box the same way you work in the Create Table dialog box.

Pack a table to reuse disk space left over from deleting records. Some restructure operations automatically pack your table. You can check Pack Table and choose OK when you want to be sure Paradox packs the table.

Note Restructuring sometimes results in the creation of temporary tables, such as a *Problems* table, that Database Desktop uses to store records that are incompatible with the table as you've restructured it. Database Desktop numbers these temporary tables consecutively (up to 99) and stores them in your private directory. For example, if you restructure twice, and both operations cause data loss, Database Desktop creates both a *Problems* and a *Problem1* table.

General rules of restructuring

When you restructure a table, you might make changes that could result in a loss of data. Changes such as shortening field sizes, creating validity checks, or changing field types can cause existing data to become invalid. Whenever this is the case, Database Desktop opens the Restructure Warning dialog box upon leaving the Restructure Table dialog box.

Figure 3.11 The Restructure Warning dialog box

Database Desktop asks you to confirm each change. Use the three panels below to specify preferences.

Check field Trim to choose the trimming preference you want without asking for confirmation on each field.

Check to delete fields without asking for confirmation on each one.

Choose whether to apply new validity check requirements to existing data.

Restructuring a table differs from creating a table in the following ways:

- You cannot change a table's type in Database Desktop. For example, you cannot change a Paradox table to a dBASE table. Use the Copy utility to do this, as described in "Copying objects" on page 60.
- If you restructure a table that was created in a previous version of Paradox in such a way that Database Desktop must convert it to a Paradox for Windows table, the Restructure Warning dialog box warns you of the conversion and asks you to confirm it.
- If you add a primary key to a table that was previously unkeyed or had different keys, you might cause *key violations*. This means that there might be data already entered into the table that violates the rules established by the new key. Database Desktop moves the records with the key violations to a special temporary table called *Keyviol*, located in your private directory.

Note If there is already a *Keyviol* table, Database Desktop adds a number to the new temporary table, so it might appear as *Keyviol1* or *Keyviol2*. Database Desktop can create up to 100 temporary tables of the same name. (The first is not numbered and the last is number 99.)

Database Desktop deletes key-violating records from your table. You can change the records in *Keyviol* so they comply with the key requirements, and then add them back to your original table using an the Add utility (see page 79).

- If you change a field's type, and Database Desktop can't convert some of the data in the field to the new type, Database Desktop prompts you to confirm the change. If you do, Database Desktop moves the records containing data that could not be converted into a special temporary table called *Problems*.

  You can change the records in *Problems* so they comply with the new structure of the table, and then add them back into the table using an INSERT query (see page 125).
- If you decrease a field's size, Database Desktop prompts you to trim existing data in the Restructure Warning dialog box. If you choose not to trim data, Database Desktop moves the records containing data that doesn't fit in the new field size to the *Problems* table.

- If you add or change a validity check, you have the option of enforcing the new validity check on existing data (make this choice from the Restructure Warning dialog box). If you choose to enforce the new validity check on existing data, and any data that doesn't comply with it, Database Desktop places the non-compliant data in the *Keyviol* table. You can change the records in *Keyviol* and then add them back to the table using an INSERT query (see page 125).

Note Changes to pictures are not enforced.

- If you add a new field that has a default validity check on it, and choose to enforce the validity check on existing data, Database Desktop creates the new field and places the default value in each record of the table. If you define a default validity check on an existing field that contains data, Database Desktop does not overwrite the existing data with the new default value.
- If you change a table's language driver when restructuring a table, you risk losing any special characters that might exist in the table.

Shortening a field

When you shorten a field that already has data in it, you might lose some data. When this is the case, Database Desktop displays the Restructure Warning dialog box (see Figure 3.11), which lets you choose whether to trim existing data, or to save records that contain data too long for the new field size in the *Problems* table.

Adding fields to an existing table

Adding fields to an existing table is exactly like adding fields when you create a new table. See "To insert a field" on page 21.

When you add fields to an existing table, Database Desktop does not automatically add those fields to any queries that use the table or to any Paradox forms or reports associated with the table. If you want the new fields added to associated objects, you must explicitly add them.

Deleting fields from an existing table

Deleting fields from an existing table is similar to deleting fields when you create a new table. (See "To delete a field" on page 21.) However, deleting a field in a table with data in it can result in a loss of data unless that field is empty. Database Desktop displays a dialog box warning you of the loss and asking you to confirm the deletion.

Editing a field name in an existing table

To change the name of a field, select the field name from the Field Roster in the Create Table (page 19) or Restructure Table (page 48) dialog box. Then position the insertion point in the field (click it or press *F2*), and use standard text-editing techniques to make the change.

To replace (overwrite) the old name completely, select the field and type the new name.

Converting a non-keyed field to a keyed field

When you convert a field from non-keyed to keyed, remember that keyed fields must be consecutive and start with the first field in the Field Roster. (You can move a field, if necessary; see page 21.)

Changing field types in Paradox tables

Changing the field type of a field in an existing table is mechanically identical to specifying the field type when you create the table. You overwrite the existing field type symbol with a new one.

However, changing the field type of a field in an existing table could result in the loss or corruption of data. When that's the case, Database Desktop prompts you to confirm the change. Field type changes have other consequences as well, illustrated in Table 3.7.

Table 3.7 Changing field types in Paradox tables

	To A	To N	To $	To D	To S	To M	To F	To B	To G	To O	To L	To I	To T	To @	To #	To +	To Y
From A	✓[1]	P[2]	P	P	P	✓	[3]				P	P	P	P	P		✓
From N	✓	✓	✓		P						✓	✓			✓	✓[4]	
From $	✓	✓	✓		✓						✓	✓			✓		
From D	✓			✓										✓			
From S	✓	✓	✓		✓						✓	✓			✓	✓[4]	
From M	✓					✓	✓	✓									
From F						✓	✓	✓									
From B								✓									
From G								✓	✓								
From O								✓		✓							
From L	✓	✓	✓		✓						✓	✓			✓	✓[4]	
From I	✓	✓	✓		✓						✓	✓			✓		
From T	✓												✓	✓			
From @	✓			✓									✓	✓			
From #	✓	✓	✓		✓						✓	✓			✓		
From +	✓	✓	✓		✓						✓	✓			✓	✓	
From Y	✓																✓

1. ✓ means Database Desktop allows the conversion, but might trim data. If Database Desktop must trim data, you'll see the Restructure Warning dialog box, which asks you to confirm the conversion.
2. P means the conversion is allowed, but might generate the *Problems* table.
3. Blank means the field type conversion is not allowed.
4. Conversion to autoincrement is allowed only from a single-field key containing data that is <2147483647.

Note Table 3.7 shows compatible field types. This definition of compatibility applies to other table operations that can be done only in Paradox for Windows (such as Add and Subtract) as well as to Query and Restructure.

Alpha field conversions

The result of converting another field type to an alpha field varies. All formatting and other definitions associated with the other field type are lost.

When you convert a field of another type to an alpha field, you must specify a length for the field. If some data already in the field contains more characters than the newly specified length of the alpha field, you can trim the data or move records containing such data to the *Problems* table.

Number, money, and short field conversions

You can convert among number, money, and short field types without data loss, except in the case where a number or money value is too large for a short field or includes decimals. In that case, you can either trim the values or move records containing those values to the *Problems* table.

Date field conversions

The following table shows which kinds of alpha strings can be converted to dates.

Table 3.8 Converting strings to dates

Strings that can be converted	Strings that cannot be converted
7/04/1776	July 4, 1776
3/30/91	The 30th of March, 1991
25-Dec-1066	Christmas Day, 1066
11-Nov-18	Armistice Day
1.01.2000	New Year's Day, the year 2000
13.06.80	Herb's 29th birthday

Restructuring tables that are linked by referential integrity

When restructuring the parent table in a referential integrity relationship, you might be prohibited from performing certain restructure operations.

Tip To see if the table you're restructuring is the parent in a referential integrity relationship, choose Dependent Tables from the Table Properties drop-down list in the Restructure Table dialog box (page 48). Database Desktop lists all child tables that depend on the table you're restructuring.

The basic rule to remember when restructuring a parent table is that *you cannot perform any operation that causes records to be removed from the table*. If you remove records from the parent table, you risk orphaning records in the child table. This is in violation of the rules of referential integrity. Each record in the child table must have a valid parent record.

Follow these guidelines as you restructure tables that are linked by referential integrity:

- If you resize any field in the parent table, you must choose to trim data that doesn't fit in the new field size, rather than save such data in the *Problems* table.
- You cannot change the parent table's key definition or the child table's foreign key definition in such a way that will cause records to be saved in the *Keyviol* table.
- You can change field names, but not types or sizes, of fields that are part of the referential integrity definition.
- You can add a validity check to either table, but you must choose not to enforce it on existing data. (Use the Restructure Warning dialog box to make this choice.) The exception to this rule is the creation of a default validity check on a new field in the table.
- To make a parent table the child of another table, that table and all its existing child tables must be empty. For example, if *Orders* is the parent table of *Stock*, you cannot make *Orders* the child of *Customer* unless both *Orders* and *Stock* are empty.
- When working with tables that contain data, if you link more than two tables by referential integrity you must create the first link to the table that has no parent. For example, to define referential integrity among the *Customer*, *Orders*, *Lineitem*, and *Stock* tables, you must
  1. First create the link from *Orders* to *Customer*.
  2. Then create the link from *Lineitem* to *Orders*.
  3. Then create the link from *Stock* to *Lineitem*.
- To create a cyclic referential integrity relationship (as in "Table A refers to Table B, which refers to Table C, which refers back to Table A") all the tables must be empty.

Saving the restructured table

When you finish restructuring the table and want to save it, you have two options:

- **Save** overwrites the old structure with the new structure. At this point, Database Desktop warns you of potential data losses based on the changes you've made.
- **Save As** creates a new table with the structure as you have specified it. Database Desktop leaves the original table intact. Check the Add Data To New Table check box in the Save Table As dialog box if you want to place data from the original table into the new table you're saving.

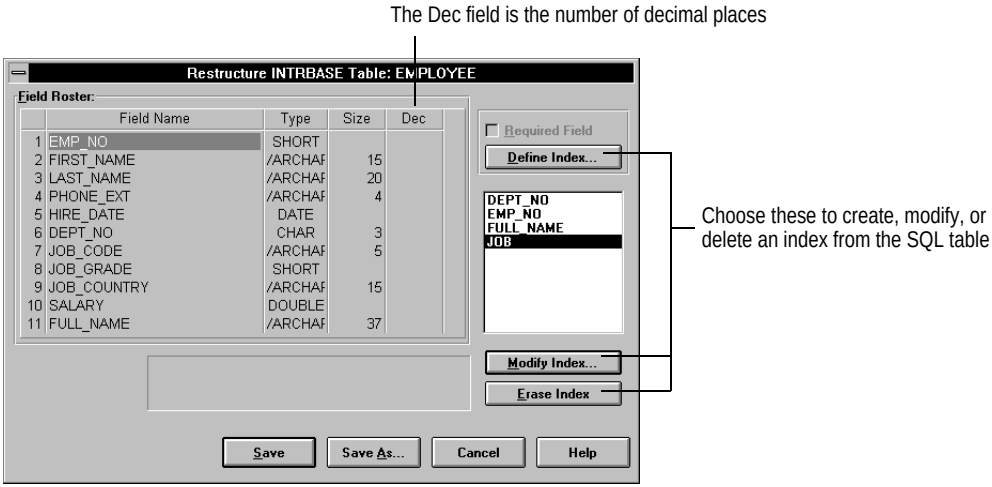
Caution If you use Save As and type an existing table's name, Database Desktop overwrites the existing table with the new table. Database Desktop displays a dialog box that warns of the possible overwrite and asks for confirmation. When the Restructure Table dialog box closes, Database Desktop opens the restructured table as well as any *Keyviol*, *Problems*, or other temporary tables created as a result of the restructure.

Restructuring a SQL table

When you restructure a SQL table, you can add, modify, and drop indexes. You cannot otherwise change the structure of a table on a server.

The Restructure Table dialog box for SQL tables looks as shown in Figure 3.12.

Figure 3.12 The Restructure Table dialog box for SQL tables



Note When you use a SQL table in Database Desktop, the table should have a unique index. If it does not have a unique index and you edit the table's data, you cannot view the edits as you are making them. To add a unique index, click Define Index.

Prefixing the index name with the table name

Database Desktop prefixes some index names with the table name, as described in "Creating indexes on SQL tables" on page 45. These index names are also affected when you restructure a SQL table as follows:

- If you create a new index during a restructure, Database Desktop prefixes the index name with the table name unless you delete the "<table>_" string from the index name.
- If you modify an index during a restructure, Database Desktop does not modify the index name, unless you rename the index as part of your modification.
- If you choose Save As during a restructure, Database Desktop renames all index names with the new table name, even if the index names are not prefixed with the current table name. (Otherwise, a duplicate index name would be guaranteed.) For example, suppose the EMPLOYEE table contains the following indexes:

  EMPLOYEE_DEPT_NO
  EMPLOYEE_EMP_NO
  FULL_NAME
  JOB

  If you restructure the table and save it as MY_DEPT, Database Desktop renames the indexes as follows:

  MY_DEPT_DEPT_NO
  MY_DEPT_EMP_NO
  MY_DEPT_FULL_NAME
  MY_DEPT_JOB

Getting table information

Database Desktop provides an easy way to get information about a table.

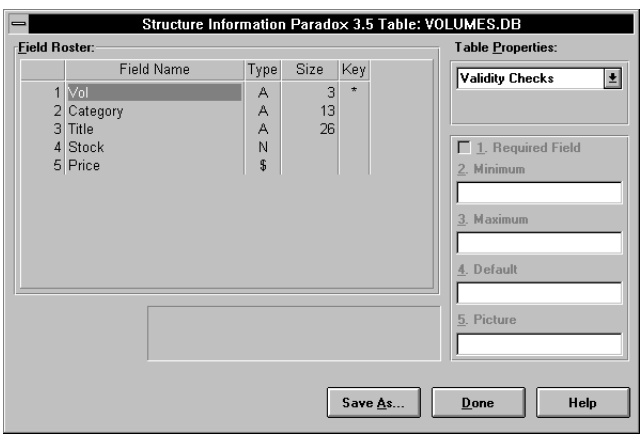
You can either choose Table | Info Structure to see the structure of an open table or choose Utilities | Info Structure and specify the table whose structure you want to see.

Database Desktop opens the Structure Information dialog box, which shows the table's structure, as well as any key, validity check, index, table lookup, or referential integrity information.

Note You can't make changes to the table's structure from the Structure Information dialog box; you must choose Restructure for that.

Figure 3.13 shows the Structure Information dialog box for Paradox and dBASE tables; Figure 3.14 shows the dialog box for SQL tables.

Figure 3.13 The Structure Information dialog box

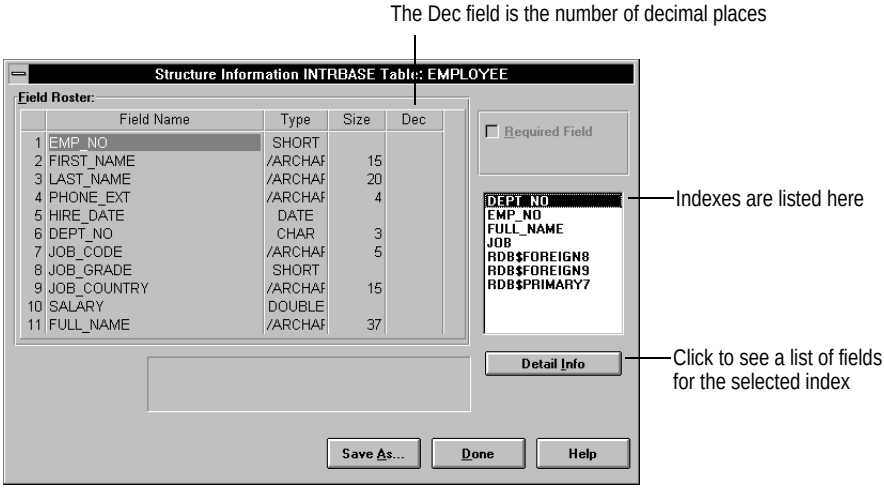


Use the Table Properties drop-down list to view information about the table.

- **Validity Checks** shows each field's defined validity checks. Move through the fields in the Field Roster to see each one's validity checks.
- **Table Lookup** shows any tables that this table uses as a lookup table.
- **Secondary Indexes** shows all the table's secondary indexes.
- **Referential Integrity** shows whether this table refers to a parent table for valid data.
- **Table Language** shows the table's language driver.
- **Dependent Tables** shows any table that this table recognizes as a child in a referential integrity relationship.

Choose Done to close the Structure Information dialog box when you're finished viewing the table's structure.

Figure 3.14 The Structure Information dialog box for SQL tables



Chapter

4

Managing objects

This chapter describes how to use the following commands on the Utilities menu to manage your objects and tables:

- Sort
- Copy
- Rename
- Delete

Sorting a table

When you sort a table, you tell Database Desktop to rearrange the order of the records in the table and display them in the order you specify.

SQL You cannot sort SQL tables.

Sorting keyed tables

If a table is keyed, Database Desktop always keeps its records sorted according to the values in the key field (or fields). When you sort the table, Database Desktop creates a new, unkeyed table containing the sorted data. The original table remains unchanged.

Sorting unkeyed tables

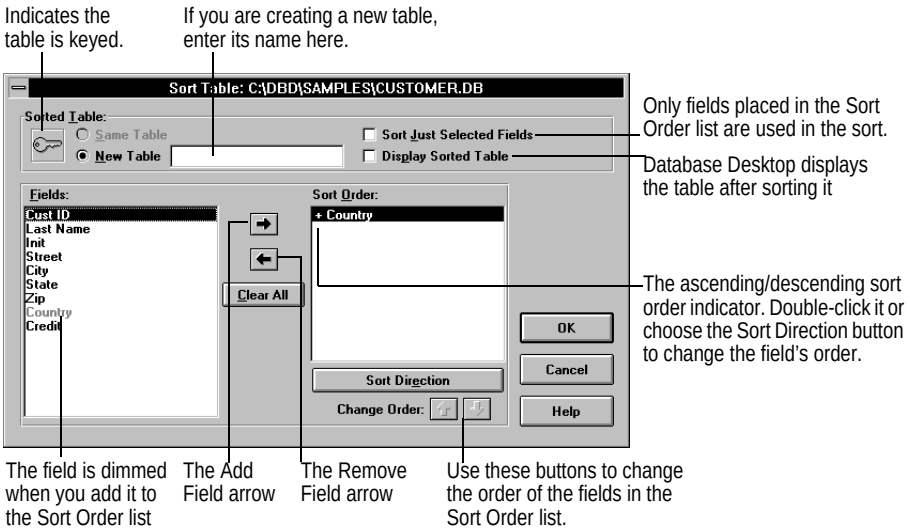
If a table is not keyed, records appear in the table in the order in which you entered them. (See "Keys in Paradox tables" on page 26 for information on creating keys.)

When you sort an unkeyed table, you change the actual location of the records in the table. You tell Database Desktop the fields on which you want the table sorted. Database Desktop then rearranges the records based on field values. You can sort an unkeyed table to itself, or create a new sorted table, leaving the original intact.

Using Sort

To sort a table, choose Utilities | Sort, then choose the table you want to sort from the Select File dialog box. Database Desktop opens the Sort Table dialog box, shown in Figure 4.1.

Figure 4.1 The Sort Table dialog box



Specifying the sort order

You specify the order in which you want to sort the records of the table by selecting fields in the Fields list and adding them to the Sort Order list. When Database Desktop performs the sort, it sorts records based on the values in the first field in the Sort Order list, then on the values in the second field, and so on.

You don't have to put all the fields from the Fields list in the Sort Order list. Database Desktop adds any fields you don't explicitly put in the Sort Order list to the end of that list before performing the sort (unless you've checked Sort Just Selected Fields). In any case, Database Desktop includes all fields in the result (whether the result is the same or a new table).

Note If you don't add any fields to the Sort Order list, Database Desktop sorts the table in the order of the fields in the Fields list. If you check Sort Just Selected Fields, you must place at least one field in the Sort Order list.

Database Desktop can't sort on the following field types:

- BLOB, BCD, logical, or bytes fields in Paradox tables
- Memo, binary, OLE, or logical fields in dBASE tables

Fields of these types are displayed in the Fields list, but are dimmed and cannot be selected for placement in the Sort Order list.

Adding fields to the Sort Order list

To add a field from the Fields list to the Sort Order list, either double-click the field in the Fields list or select the field in the Fields list and choose the Add Field arrow or press *Alt+A*. The field appears in the Sort Order list immediately below the selected field. The field name remains in the Fields list, but is dimmed to indicate that it's no longer available.

To insert a field at the top of the Sort Order list,

1. Select the top field in the Sort Order list.
2. Add the field you want. It appears selected below the top field.
3. Use the Change Order up arrow to move the field to the top position.

To add a consecutive group of two or more fields from the Fields list to the Sort Order list, click a field at one end of the range and drag to the other end of the range. (Or, using the keyboard, move to the top field in the range, hold *Shift* and press ↓ until all the fields you want are selected.) You can also hold *Shift*, then click the fields at the beginning and end of the range. Then choose the Add Field arrow or press *Alt+A* to move the selected range of fields to the Sort Order list. If the range of fields you select extends over fields that can't be sorted on, or over fields that have already been added to the Sort Order list, Database Desktop ignores them.

Removing selected fields from the Sort Order list

To remove a single field from the Sort Order list, select it and choose the Remove Field arrow (or press *Alt+R*). The field returns to the Fields list. To remove a range of fields, select the range, then choose the Remove Field arrow (or press *Alt+R*). The Remove Field arrow is available only when a field is selected in the Sort Order list.

Removing all fields from the Sort Order list

To remove all fields from the Sort Order list, making those fields available again in the Fields list, choose Clear All (or press *Alt+C*). Clear All is available whenever a field appears in the Sort Order list.

Rearranging fields in the Sort Order list

To move a field or group of fields to a different position in the Sort Order list, select the field(s) you want, then click the up or down Change Order arrows.

The Change Order arrows are available only when two or more fields are on the Sort Order list.

Using ascending or descending sort order

Each field in the Sort Order list is preceded by a sort order indicator that shows whether the sort order within the field is ascending (shown as +) or descending (shown as –). The default is ascending. To reverse the sort order for a field, double-click the sort order indicator or select the field and choose the Sort Direction button.

The Same Table/New Table option

Database Desktop gives you an option for how to save the results of a sort:

- Select Same Table if you want the result of the sort to overwrite the existing sort order of the table you are sorting. Same Table is available only if you're sorting an unkeyed table.
- Select New Table if you want the result of the sort to create a new table. Type the name of the new table in the New Table text box.

The Sort Just Selected Fields option

When you check Sort Just Selected Fields, Database Desktop sorts only those fields that appear in the Sort Order list. All the fields of the source table are included in the resulting sorted table, but are not sorted beyond the fields listed in the Sort Order list.

Note If you check Sort Just Selected Fields and two or more records have identical values in their sorted fields, Database Desktop cannot sort those records and places them in the table as a group, but unsorted within the group.

If you *don't* choose Sort Just Selected Fields, Database Desktop performs the sort first on the fields in the Sort Order List, then—if two or more records have identical values in their sorted fields—on the fields remaining in the Fields List (in the order in which they appear).

The Display Sorted Table option

When you check Display Sorted Table and perform the sort, Database Desktop opens the sorted table when the Sort Table dialog box is closed.

Performing the sort

When you finish specifying the sort and the options you want, choose OK to perform the sort.

Sorting on a network

When you sort tables in a multiuser environment, Database Desktop automatically places a lock on the table you're sorting. This means other users can't modify the table's contents or structure. If another user has a lock on the table, you won't be able to begin sorting until that user finishes working with it.

When you sort to a new table, Database Desktop automatically places a lock on that table as well as the original table for the duration of the sort.

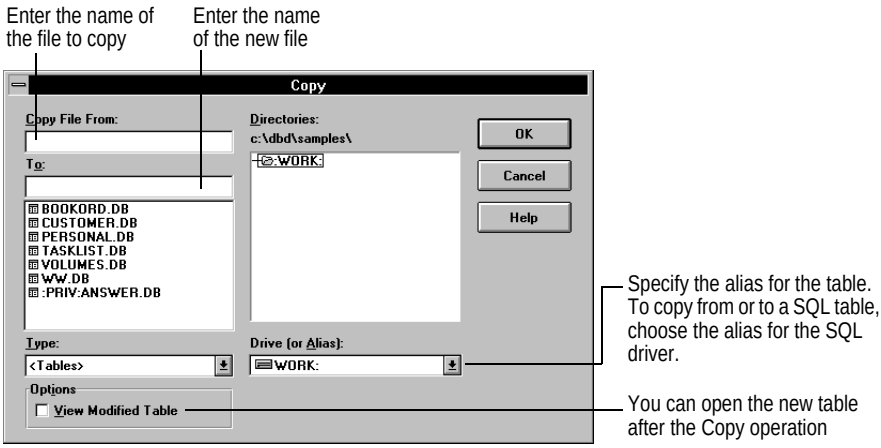
Copying objects

You can copy tables, queries, SQL tables, and .SQL files from within Database Desktop.

Always use the Database Desktop Copy utility to copy Paradox or dBASE tables. Using the DOS COPY command or the Windows File Manager may not copy all related files that make up a table (for example, the files containing a table's primary index, secondary indexes, validity checks, or BLOB data). The Database Desktop Copy utility, however, copies all files correctly.

To copy an object, choose Utilities | Copy. Database Desktop opens the Copy dialog box, shown in Figure 4.2.

Figure 4.2 The Copy dialog box



All tables in your working and private directories are shown in the file list. Use the Type drop-down list to display other object types.

You can use the Drive Or Alias drop-down list or the Directories panel to access files in different directories.

When you copy a table, Database Desktop copies both its structure and the data contained in it. Database Desktop also copies the table's

- Key (primary index)
- Secondary index(es) (except .NDX files on dBASE tables)
- Validity checks (see "Copying referential integrity" on page 62)
- Table properties (as you've set them in the Table window)

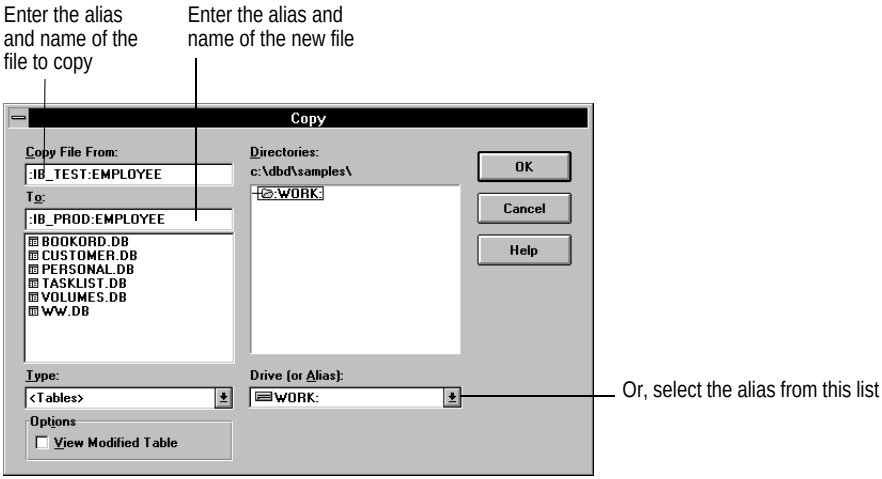
Copying SQL tables

To copy a from or to a SQL table,

1 Select the alias for the source table.
2 Choose the source table.
3 Select the alias for the destination table.
4 Type the destination table name.

You can type the alias name or select it from the list of aliases, as shown in Figure 4.2.

Figure 4.3 Copying to or from a SQL table



When you copy a SQL table to a new table in the same database (alias) or to a different database of the same type (such as InterBase to InterBase), Database Desktop copies

- The field structure exactly (including required field settings)
- Indexes, renaming them for uniqueness in the database as described in "Creating indexes on SQL tables" on page 45 and "Prefixing the index name with the table name" on page 54.

Copying on a network

When you copy a table, Database Desktop must acquire a read lock on the original table and an exclusive lock on the copy. This means

- No user can change the contents or the structure of the table you're copying during the Copy operation.
- If you copy to an existing table, no locks can be open on that table.

If a record lock, write lock, or exclusive lock exists on the table you're copying, you won't be able to make the copy until the lock is removed.

Copying referential integrity

When you define referential integrity (see Chapter 3), you create a *parent/child relationship* between two tables.

- If you copy the *parent* table, Database Desktop doesn't copy the referential integrity.
- If you copy the *child* table, Database Desktop copies the referential integrity. This means the copied table must meet the requirements of the referential integrity. To delete the referential integrity, you must restructure the table.

- Both tables in the referential integrity relationship must be in the same directory. When you copy the child table to a different directory, you break the referential integrity link.

Copying to a different table type

You can copy tables to and from any table type.

To copy a Paradox table to a dBASE table, or a dBASE table to a Paradox table, type the file extension you want (.DB for Paradox and .DBF for dBASE) for the copied table. For example, if you want to copy the Paradox *Customer* table to a dBASE *Customer* table, type `customer.dbf` as the name of the copied table.

Field type conversions

Database Desktop automatically changes field types when you change table types.

The following tables show what to expect when you copy between Paradox and dBASE tables. For information on field type conversions for SQL tables, see your SQL Links documentation.

Table 4.1 Copying from a Paradox to a dBASE table

From Paradox type	To dBASE type	Side effects
Alpha	Character	
Number	Number	Assigns size (20) and dec. (4)
Money	Number	Assigns size (20) and dec. (4)
Short	Number	Assigns size (6) and dec. (0)
Long Integer	Number	Assigns size (11) and dec. (0)
BCD	Number	Assigns size (20) and dec. (4)
Date	Date	
Time	Character	Assigns size (8)
Timestamp	Character	Assigns size (30)
Memo	Memo	
Formatted Memo	Memo	Formatting is lost
Graphic	Binary	
OLE	OLE	
Logical	Logical	
Autoincrement	Number	Assigns size (11) and dec. (0)
Binary	Memo	Data cannot be displayed
Bytes	Memo	Data cannot be displayed

Note If the new dBASE table contains no production index (.MDX file), no float number field type, and no memo field type, Database Desktop creates a dBASE III+ table. If the

dBASE table contains an OLE or binary field, Database Desktop creates a dBASE for Windows table. Otherwise, Database Desktop creates a dBASE IV table.

Table 4.2 Copying from a dBASE to a Paradox table

From dBASE type	To Paradox type	Side effects
Character	Alpha	
Float	Number	Removes size
Number	Number	Removes size
Logical	Logical	
Date	Date	
Memo	Memo	Adds size (1)[1]
OLE	OLE	
Binary	Graphic	

1. Paradox assumes the data in the dBASE memo is in text form. If the memo contains a different type of data, you should use the Add utility and add the memo to the appropriate Paradox BLOB field type.

Index conversions

Whenever possible, Database Desktop copies indexes when you change table types. Database Desktop chooses the closest matching index type for the destination table. For example:

- When copying from a SQL table to a Paradox table, only one unique index can be copied, so Database Desktop copies the primary index. Unique indexes on the SQL table become secondary indexes on the new Paradox table.
- dBASE expression indexes cannot be copied.

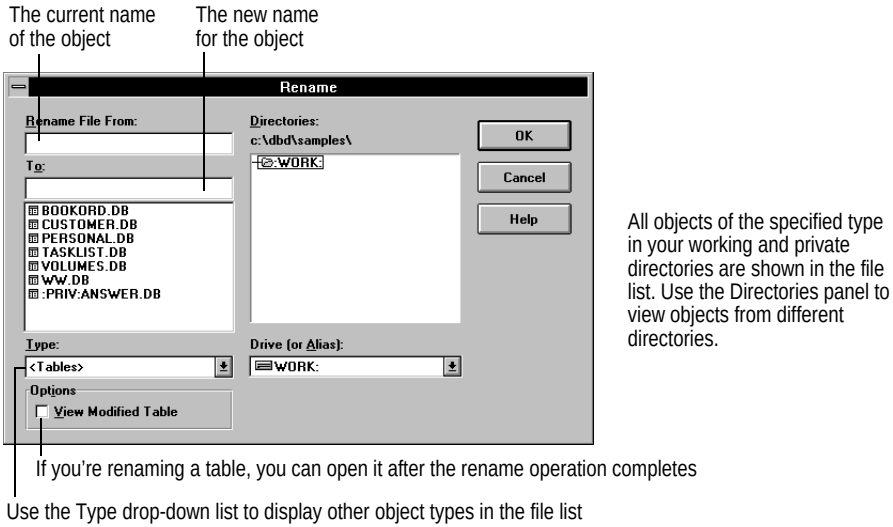
Renaming objects

You can rename tables, queries, and .SQL files from within Database Desktop. You cannot rename SQL tables.

Always use the Database Desktop Rename utility to rename tables. Using the DOS RENAME command or the Windows File Manager may not rename all related files that make up a table (for example, the files containing a table's primary index, secondary indexes, validity checks, or BLOB data). The Database Desktop Rename utility, however, renames all files correctly.

To rename an object, choose Utilities | Rename. Database Desktop opens the Rename dialog box, shown in Figure 4.4.

Figure 4.4 The Rename dialog box



Rules for renaming objects

Follow these rules when renaming objects:

- You cannot rename a table to change its type. A Paradox table must be renamed as a Paradox table, and a dBASE table must be renamed as a dBASE table.

  You can copy a table to changes its type. See "Copying objects" on page 60.

- You cannot rename a table that is identified as the parent table in a referential integrity relationship.
- When renaming an object, you can type a full path when you type the object's new name. This both renames the object and moves it to a new location.

Caution Be careful when renaming tables. Once renamed, a table can't be found by associated documents. Queries that refer to a table under one name won't be bound to the table under its new name.

Using Rename on a network

When you use Rename, Database Desktop must acquire an exclusive lock on the table. This means

- No user can access the table in any way.
- If there is a lock of any type open on the table, you must wait until it's released before you can use Rename.
- If you rename an object with an existing object's name, Database Desktop deletes the existing object.

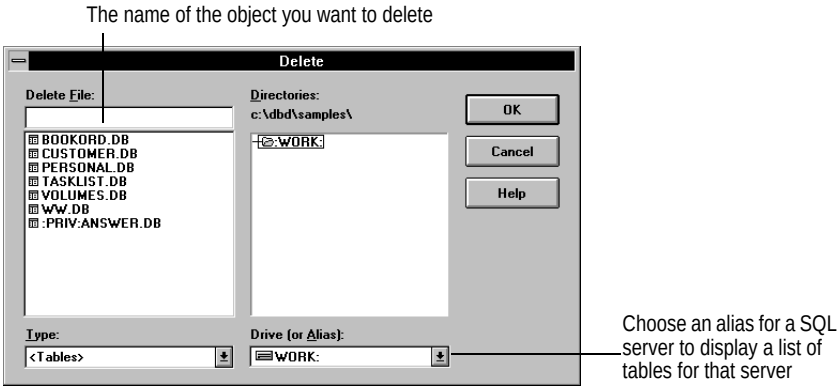
Deleting objects

You can delete tables, queries, SQL tables, and .SQL files from within Database Desktop.

Always use the Database Desktop Delete utility to delete tables. using the DOS DELETE command or the Windows File Manager may not delete all related files that make up a table (for example, the files containing a table's primary index, secondary indexes, validity checks, referential integrity, or BLOB data). The Database Desktop Delete utility, however, deletes all files correctly.

To delete an object, choose Utilities | Delete. Database Desktop opens the Delete dialog box, shown in Figure 4.5.

Figure 4.5 The Delete dialog box



All tables in your working and private directories are shown in the file list. Use the Type drop-down list to display other object types in the file list.

You can use the Directories panel to access files in different directories.

Enter the name of the object you want to delete in the Delete File text box. Choose OK to delete it. Database Desktop displays a message asking you to confirm the deletion.

SQL To delete a SQL table, select the alias associated with the SQL server, then either select or type the table name.

To delete more than one object, hold *Shift* and click the names of the objects you want to delete.

Note You cannot delete a table that is identified as the parent in a referential integrity relationship. You must first either delete the referential integrity (from the child table), empty the child table, or delete the child table.

Caution Be careful when deleting objects. You can't undo a deletion. Be sure that a table isn't used in any queries before you delete. Queries that depend on the table are *not* deleted when the table is deleted.

Using Delete on a network

When you delete a table with Delete, Database Desktop must acquire an exclusive lock on the table. This means

- No user can access the table in any way.
- If a lock of any type is open on the table, you must wait until it's released before you can use Delete. This means you cannot delete a table that is open on your Desktop.

Chapter

5

Entering and editing data

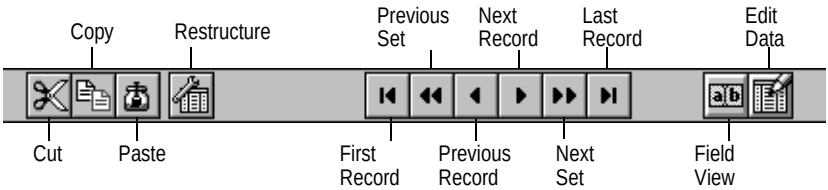
This chapter discusses tables and how you can work with them, such as

- Customizing your view of a table
- Using menu commands and the Toolbar
- Working in Edit mode
- Cutting, copying, and pasting data
- Inserting, deleting, and locking records
- Working with validity checks and table lookup
- Using the Add, Subtract, and Empty utilities to change entire tables

Viewing tables

To open a table, choose File | Open | Table. Or, if the application window is empty, click the Open Table button in the Toolbar. In the Open Table dialog box, choose the table to open. The table you chose opens in a Table window, and the Toolbar appears as shown in the following figure.

Figure 5.1 The Table window Toolbar



When you first open a table, its data appears in a Table window in View mode. The data appears in column and row format and displays any formatting properties included in the file. Each Table window contains an independent view of a table, so different views of a single table can be open at the same time. Up to 24 tables can be open at one time.

Note To be able to simultaneously access tables stored on a network, you must tell Database Desktop the location of the network control file. You do this by running the BDE Configuration Utility; double-click the BDE Configuration Utility icon in the program group where the Database Desktop icon appears. See online Help in the BDE Configuration Utility for more details.

Moving through a table's records

Use the Toolbar navigation buttons or the Record menu to move through the records of the table. The buttons and commands work as follows:

First moves to the first record in the table.

Last moves to the last record in the table.

Next moves to the next record.

Previous moves to the previous record.

Next Set moves to the next screenful of records.

Previous Set moves to the previous screenful of records.

Using scroll bars

To scroll left or right through the columns of the table, use the horizontal scroll arrows. To scroll up or down one record at a time, use the vertical scroll arrows.

When you drag the box on the vertical scroll bar, a range of record numbers appears in the status line. These numbers represent the records that will appear in the window when you release the mouse button; the view isn't updated until you release the mouse button.

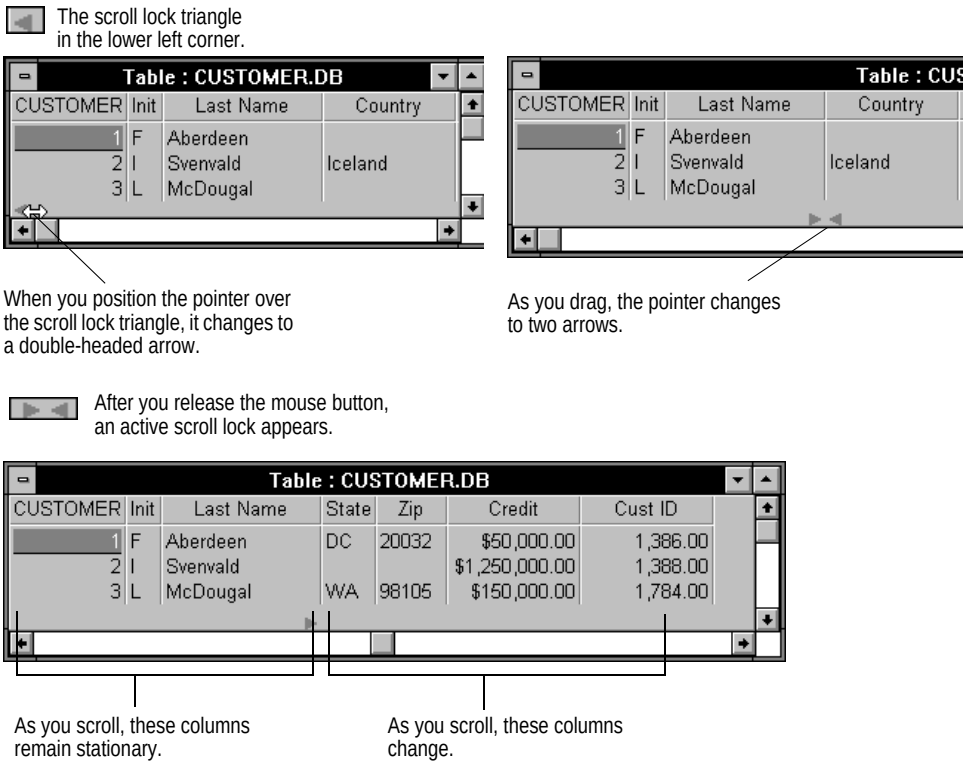
Note In a keyed table, when you move the vertical scroll box, the status line uses entries in the key field (or the first field of a composite key) rather than record numbers to indicate which records will be displayed.

Using scroll lock

To lock one or more columns in place as you move horizontally through the table's columns, use a *scroll lock*. All columns to the left of the lock remain stationary as you move through the table's columns.

The scroll lock is a triangle in the lower left corner of the Table window. To place a lock, drag the triangle to the right side of the column(s) you want to lock. An active scroll lock appears as two triangles when you release the mouse button, as shown in the next figure.

Figure 5.2 A scroll lock in the Table window

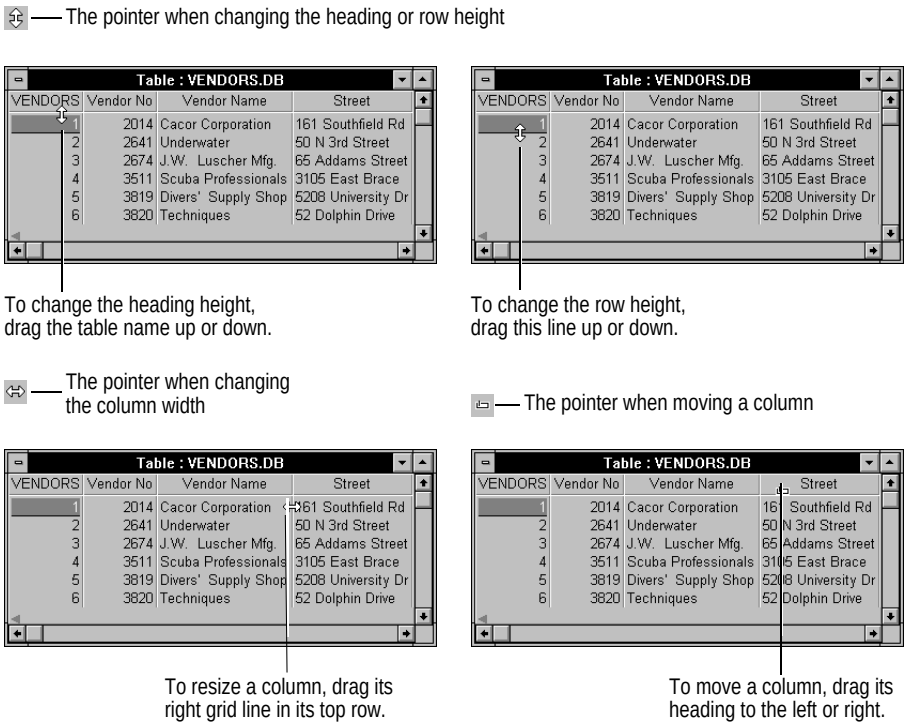


Customizing a table view

The *view* of a table is how it appears onscreen; you can modify and save a custom view of a table. Changing the view makes it easier to see specific fields; the actual *structure* of the table (its definition of field order and size) remains the same. To customize a view, you can rearrange, resize, and lock columns, and resize rows or table headings.

The following figure shows the *hot zones* on an open table view. Hot zones indicate areas on a table where you can drag to modify the view of the table. As the pointer passes over a hot zone, the pointer changes shape.

Figure 5.3 Hot zone pointers in the Table window



Rearranging and resizing columns

To move a column, position the pointer on a column's heading. When the pointer changes shape (shown at left), drag the column to its new position.

To resize a column, position the pointer on its right boundary line (either the heading area or the top row of data). When the pointer changes shape (shown at left), drag the boundary line to increase or decrease the width of the column.

Resizing rows

To resize the height of all of the rows in a table, drag the line under the first record number. Move the line up to decrease the row height, or down to increase the row height.

Resizing column headings

To resize the height of all the column headings, drag the table name. The table name is located above the leftmost column (which contains the record numbers).

Saving a custom view

To save property changes for an active table, choose Properties | Save. All display changes you make in the table view (except scroll locks) are saved to a file. The file that

holds table view properties has the same name as the table, but the extension is either .TVF (for a dBASE table) or .TV (for a Paradox table).

Undoing changes to a view

To erase any unsaved changes you've made to a view and restore the last-saved view, choose Properties | Restore. Database Desktop reinstates the view properties stored in the associated .TV or .TVF file. If there is no view properties file, the default view of the table is reinstated.

Restoring the default view

If you've changed and saved the view of the active table, but now want to return to the default view, choose Properties | Delete. This deletes the .TV or .TVF file associated with the active table. If there's a DEFAULT .TV or .TVF file in your private directory, the properties of this file will be used for all tables that don't have their table view file. (You will have a DEFAULT.TV or .TVF file in your private directory if you've ever saved a table view and then used the DOS COPY command to copy it to a file called DEFAULT.TV or .TVF.)

Tip If, after you run a query, the resulting Answer table looks different than expected (for example, if the columns are in a different order than they should be), try choosing Properties | Delete, and then run the query again. There might be an old ANSWER.TV file that Database Desktop is applying to your new Answer table.

Editing data

To change data in a table, you must be in Edit mode. To enter Edit mode, do one of the following:

- Click the Edit Data button in the Toolbar.
- Choose View | Edit Data.
- F9 Press *F9*.

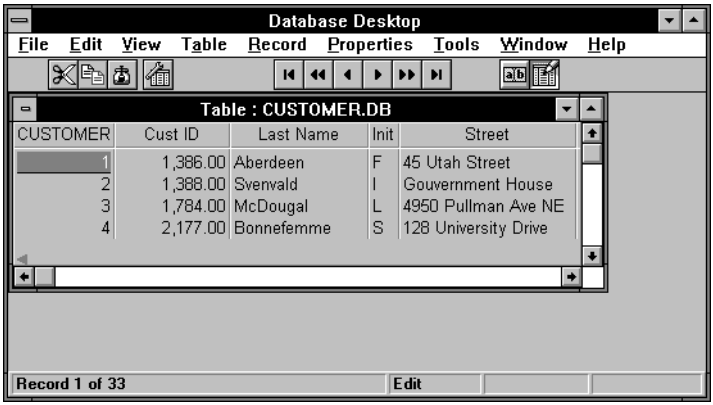
In Edit mode, you can select any field and begin typing to replace its existing entry. When you enter Edit mode, the Edit Data button remains pressed in and the status line tells you Edit mode is active (as shown in the next figure).

Note In Database Desktop you cannot edit data in the following field types:

- **Paradox:** Memo, Formatted Memo, Graphic, OLE, Autoincrement, Binary, or Bytes
- **dBASE:** Memo, OLE, or Binary
- **SQL**: any BLOB (binary large object) field or a text field that allows more than 255 characters

To enter or edit data in these field types, use Paradox for Windows or dBASE, or an application that supports editing those field types.

Figure 5.4 The *Customer* table in Edit mode



Note To position the insertion point within the entry so you can change a spelling or typing error, use field view (see the next section).

Selecting fields and records

When you move to a field or click it, the field is highlighted. This indicates that the field is *selected*. In Edit mode, if you type anything into a selected field, you'll replace the existing entry with the value you type. The cut, copy, and paste operations affect the entire field entry when it's selected.

You can select more than one field at a time, or select a portion of a single field entry.

Note In Edit mode, if a field is already selected, clicking the same field again enters field view (see page page 74). To exit field view, move off the field by clicking another field or by pressing *Tab* or an arrow key.

Selecting multiple fields

To select multiple fields across rows and columns, drag from one corner to the other (be sure you're not in field view). During multiple selection, the pointer appears as a four-headed arrow. Fields selected with this method must be contiguous.

To select a group of fields using the keyboard, select the field where you want to begin (don't enter field view) and hold down *Shift*, while pressing the arrow keys to place a box around the fields you want.

Selecting all records

To select the entire table, choose Edit | Select All. A selection box surrounds the whole table.

Field view

In Edit mode, you can change a field's entry in one of two ways:

- Select the field and type a new value. When you begin typing, the new value replaces the old entry.
- Select the field and edit the existing entry.

Data you enter into a field must match the field's data type. See Tables 3.1 and 3.2.

When you revise field entries in Edit mode, you can insert or delete characters without retyping the whole entry. Begin by selecting the field you want to change. Then enter *field view* in one of the following ways:

- Select the field, then click the Field View button in the Toolbar.
- Select the field, then press *F2*.
- Select the field, then choose View | Field View.
- Double-click an unselected field (or click a selected field again). This method places the insertion point in the field where you double-click.

When you enter field view, you can edit within a field entry. The insertion point appears at the end of the field or at the place where you clicked.

You can move the insertion point by clicking in the field, pressing editing keys such as *Home* or *End*, or pressing arrow keys. *Backspace* deletes characters to the left of the insertion point, and *Del* deletes characters to the right. You can also drag in the field to select characters.

See Table A.5 on page 152 for a list of keyboard actions in field view. In general, when you're in field view you can combine *Alt* with the navigation keys to get the same result as the key alone gives you when you're not in field view.

Exiting field view

When you move off a field, you exit field view. This happens whenever you click another field, or press *Enter*, *Tab*, or *Alt* plus an arrow key. To exit field view and remain on the current field, you can click the Field View button, press *F2*, or choose View | Field View.

To select another field *and* remain in field view, use persistent field view, described next.

Persistent field view

Unlike field view, which ends as soon as you leave the field you were editing, *persistent field view* lets you move among fields without leaving field view. In Edit mode, press *Ctrl+F2* to enter persistent field view.

Ctrl F2

In persistent field view you can use *Home*, *End* and the arrow keys just as in standard field view. Press *Tab*, *Enter*, or *Alt* plus an arrow key to move from field to field.

When you first select a field in persistent field view, the entire field is highlighted. Replace mode is still the default mode for data entry, until you click the mouse button (or press an arrow or editing key) to position the insertion point.

To exit persistent field view, press *Ctrl+F2*.

Table A.5 on page 152 lists the keyboard actions in persistent field view.

Cutting, copying, and pasting data

In addition to typing a field value while in Edit mode, you can also cut or copy a value, then paste it into a field. Also, blocks of records or portions of records can be copied to the Clipboard, then pasted into a notebook or other Windows applications.

Note Changes you make in Edit mode are posted (saved in the table) when you move off the record.

The following table lists the cut, copy and paste operations performed by the Edit menu commands while in Edit mode.

Table 5.1 Cut, copy, and paste operation commands

Command	Operation it performs
Cut	Deletes a single field entry from the table, or any number of whole records selected as a block, and places the data on the Clipboard.
Copy	Places a duplicate of the selected field entry(s) on the Clipboard. To select only a portion of an entry in a field, first enter field view. To select all entries in a column, double-click the column heading.
Paste	Inserts a *single* field value from the Clipboard into a selected field in the table. Multiple values on the Clipboard are pasted only into a spreadsheet or a word-processing application.
Paste Link	Inserts a Dynamic Data Exchange (DDE) link from the Clipboard to a field in the *query image*. Paste Link is available in the Table window, but you can only paste a link into an alpha field.
Delete	Removes the selected entry *without* placing it on the Clipboard.
Select All	Selects *all entries* in the active table.

Note In Edit mode, you can paste a value into a field only when its data type matches the field type. For a list of field types, see the tables starting on page 23.

Using Undo

In Edit mode, edits are *posted* (saved in the table) as soon as you move off a record. To discard any edits and restore the original record, choose Edit | Undo or Record | Cancel Changes *before* moving off of the record.

After the original record is restored, a message in the status line lets you know the changes were discarded.

To discard changes to a single field value and restore the original field entry, press *Esc* before you move off the field.

Caution You cannot use Edit | Undo or Record | Cancel Changes to retrieve a Paradox record you've deleted (but you can retrieve a dBASE record, if you have dBASE).

Removing entries from fields

In addition to the Cut command in Edit mode, Database Desktop provides the Delete command to remove the selected entry *without* placing it on the Clipboard.

Note Edit | Delete empties the selection.

To remove a single field entry in Edit mode, select a field, then choose Edit | Delete. (If multiple fields are selected, the command is unavailable.) If field view is also active, you can select specific text, then choose Edit | Delete to remove it.

Note Record | Delete removes the entire record.

To remove an entire record (row), select all fields in the record, including the record number, then choose Record | Delete. A dialog box confirms the action; after you choose OK, the record and all field entries in it are removed.

Tip You can use Edit | Delete only on records (rows) or single field entries of the table. You can't delete a field (column) from a table. To work with only specific fields in a table, you can create a query and use the *Answer* table that results (see page 93).

Inserting and deleting records

In Edit mode, you can insert new blank records, or delete existing records from a table.

Ins Choose Record | Insert (or press *Ins*) to insert a blank record *above* the selected record.

When you insert a record into a keyed Paradox table, then enter values into its fields, the record immediately moves to its proper position according to the sort order. Records inserted in non-keyed tables remain at the positions where they are entered. Records inserted in dBASE tables move to the end of the table.

Ctrl Del Choose Record | Delete (or press *Ctrl+Del*) to delete the selected record.

Caution Make sure you want to delete the *entire record* before you choose Delete; you can't retrieve a deleted record in Database Desktop.

Exiting Edit mode

Exit Edit mode using any of these methods:

- Choose View | End Data Entry.
- Click the Edit Data Toolbar button.
- Press *F9.*

Field types

When you work with tables, you enter and edit data in a variety of *field types.* A field type determines the kind of data you can enter in a field. Field types and possible values are listed in Table 3.1 for Paradox tables and in Table 3.2 for dBASE tables.

In Database Desktop you cannot enter data in any of the following field types:

- **Paradox**: Memo, Formatted Memo, Graphic, OLE, Autoincrement, Binary, or Bytes
- **dBASE**: Memo, OLE, or Binary
- **SQL**: any BLOB (binary large object) field or a text field that allows more than 255 characters

To enter or edit data in these fields, use Paradox for Windows or dBASE.

Locking records

In Edit mode, as soon as you begin editing a record, Database Desktop *locks* it. When a record is locked, other users can view it, but can't edit it or delete it. A message in the status line lets you know when a lock is active. Database Desktop removes the lock and *posts*, or saves, the record as soon as you move off the record or turn off Edit mode.

Manually locking records

F5 To manually lock a selected record while in Edit mode, choose Record | Lock (or press *F5* or *Ctrl+L*). You might want to manually lock a record when you

- Work in a multiuser environment
- Use different views of the same table in one session

Shift F5 With manual locks, a message in the status line lets you know when a lock is active, and the Lock command is replaced by the Unlock command. Before other users can access manually locked records, the records must be unlocked. Press *Shift+F5* or *Ctrl+Shift+L* to post the record and unlock it.

Ctrl F5 After you choose Record | Lock, the Record | Post/Keep Locked command is available on the Record menu. Use this command if you want to save your edits without leaving the record. When this locking option is used, the insertion point remains on the record, even if the record is relocated due to the table's sort order. If necessary, your view of the table is updated.

Fields with validity checks

Validity checks are a feature of Paradox tables that prevent data from being entered in a field unless the data meets certain requirements. Validity checks are defined when a table is created or restructured. See page 28 for a description of the types of Paradox validity checks and how to create them.

Multiple validity checks—such as a minimum, maximum, and default value—can be active for a single field.

An error message appears (if the value in any field is invalid) *after* Database Desktop attempts to post, or save, the record. An attempt is made to post the record after you try to move off the record or unlock it.

Looking up table values

A *table lookup* is a defined relationship between two Paradox tables—a lookup table and the table you're editing. These table relationships are defined when a table is created or restructured.

For example, if you're not sure how a customer's name is spelled, you can use table lookup to browse in the lookup table. The correct customer name, and corresponding values such as a customer identification number and address, can be copied from the lookup table.

The status line in the table window tells you if Lookup Help is available for the field.

See page 32 for more information on Paradox lookup options.

Using Move Help

In certain situations, you might have a record in one table that corresponds to a record in another table. This can happen in a referential integrity relationship, where one record in a parent table is related to one or more records in a child table.

In this case you can use Move Help to move a dependent record from one master to a different master.

Choose Record | Move Help in a Table window. Then in the Move Help dialog box select the new master record from the master table and choose OK. The detail record you select is now assigned to the new master record.

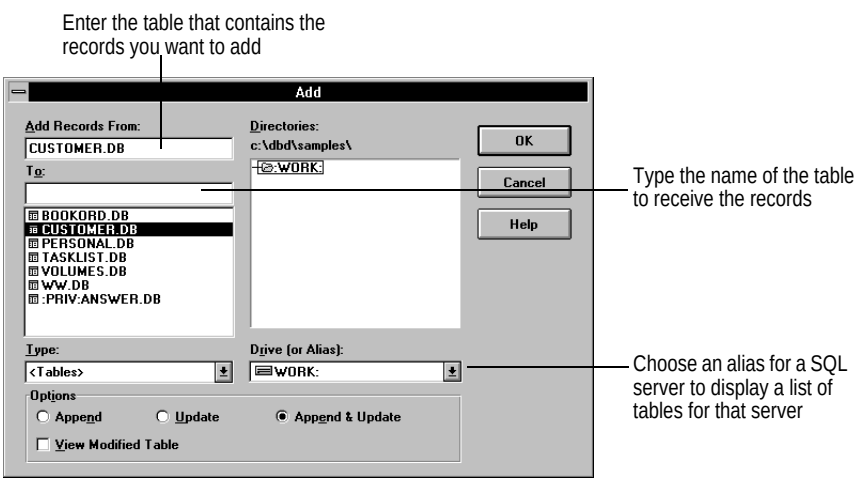
Move Help is available only in fields for which a referential integrity relationship is defined.

Adding records to a different table

You can add the records from one table to another table. You can use Add on all table types.

Choose Utilities | Add. Database Desktop opens the Add dialog box, shown in Figure 5.5.

Figure 5.5 The Add dialog box



All tables in the working and private directories are shown in the list on the left. Use the Directories panel to choose a table from a different directory. You can perform an Add operation across directories.

1. Position the insertion point in the Add Records From text box, then choose the table you want to add records from.
2. Position the insertion point in the To text box, then choose the table you want to add records to.
3. Choose Append or Update. (These options are discussed later in this section.)
4. Check View Modified Table if you want to open the table you added records to when the Add operation is complete.
5. Choose OK.

The two tables you use in the Add operation must have *compatible* (though not necessarily identical) field types in the same order.

For fields to be compatible, Database Desktop must be able to change from the existing field type to the new field type in a Restructure operation. For example, number and currency field types are compatible, but number and graphic fields are not.

- For a complete chart of compatible Paradox field types, see Table 3.7 on page 51.
- For a complete chart of restructure-compatible dBASE field types, search online Help for the subject "dBASE field types" and the topic "Compatible dBASE Field Types."
- For information about compatible field types for SQL tables, see your SQL Links documentation.

When performing an Add operation, keep these rules in mind:

- You can add records from one table type to another only if the tables have a compatible structure. This means compatible field types in the same order.
- The table you add records to can have more fields than the source table, as long as the first fields of the table you add the records to are compatible with all fields of the source (compatible fields types in the same order). Database Desktop places null values in the extra fields.
- The source table can have more fields than the table you add the records to, as long as the fields of the table you add the records to are compatible with the first fields of the source (compatible field types in the same order). Database Desktop ignores the extra fields.

Adding records to a different table type

When you add records from one table type to another, consider whether the field types in the table you add records to are compatible with the field types in the table you add records from. The rules for adding records from one type to another are the same as those for restructuring from one table type to another.

Note Some field type conversions can result in invalid records being written to the temporary *Problems* table. If this happens, edit the records in the *Problems* table and then add them

again. The *Problems* table is not generated for SQL tables; the invalid records are dropped.

Make sure you add dBASE data to an appropriate Paradox field type. For example, if your dBASE memo contains graphic images, it is valid to add them to the graphic field type. It isn't valid to add graphics to a formatted memo. Database Desktop doesn't interpret the data in the dBASE memo before the addition, so make sure that the data in the dBASE field is compatible with the Paradox field type.

Adding records to keyed tables

If the table you add the records to is keyed, the added records must conform to the rules of the key. Database Desktop places records that don't conform in the temporary *Keyviol* table in your private directory. The source table is never changed during an Add operation.

Append and Update options

In the Add dialog box, you can use the Options area to add new records, update existing records, or do both.

- **Append:** Adds new records without affecting any existing records.
  - If the target table is keyed, Database Desktop adds records in their proper position in the table. Database Desktop places records that violate the key in the temporary *Keyviol* table in your private directory. (You can edit these records to conform to the key, then use Add again to place them in the table.)
  - If the target table isn't keyed, Database Desktop places the added records at the end of existing records.
- **Update:** Updates records that already exist in the table you're adding records to. Any records in the source table that don't match an existing record aren't added.

  When you choose Update, the records of the source table overwrite matching records in the table you're adding records to. Database Desktop places the records that are overwritten in the temporary *Changed* table in your private directory.

  Update is not allowed on dBASE and SQL tables.

Note The table you add records to *must* be keyed to use Update.

- **Append & Update:** Both adds new records (following the rules just stated) and updates existing records (following the rules just stated).

  Append & Update is not allowed on dBASE and SQL tables.

Adding on a network

When you use Add, Database Desktop must acquire a read lock on the source table and a write lock on the table you add the records to. This means other users can't

- Change the contents or structure of either table

- Perform any operation that requires a write or exclusive lock on either table

If another user has locked a record or placed a write or exclusive lock on either table, you must wait until the lock is removed before using Add.

Because dBASE tables don't use read locks, when you use Add on dBASE tables, Database Desktop places write locks on both tables.

Subtracting records

You can remove records that exist in one table from a different table by using the Subtract utility. You can subtract records only from a keyed table.

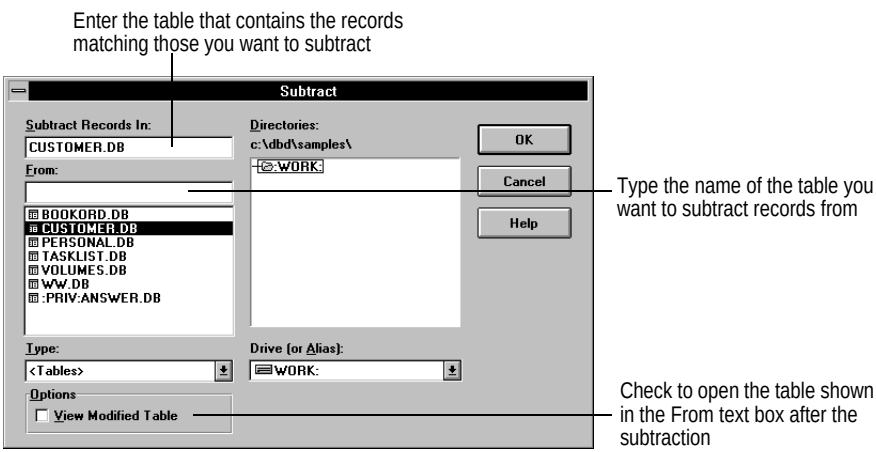
Because the table you subtract records from must be keyed, and dBASE and SQL tables do not support Paradox keys, you cannot subtract records from dBASE or SQL tables. Instead, use a DELETE query (see Chapter 7).

SQL You cannot use a SQL table as the source of a Subtract operation.

During a subtract operation, Database Desktop removes any record that contains a value in its key field that exactly matches the value in the corresponding field of a record in the subtraction table.

To subtract records, choose Utilities | Subtract. Database Desktop opens the Subtract dialog box, shown in Figure 5.6.

Figure 5.6 The Subtract dialog box



To subtract the records in one table from another table,

1. Enter the name of the table that contains the records matching those you want to subtract from the other table in the Subtract Records In text box.
2. Enter the name of the table you want the records subtracted from in the From text box.

3 Choose OK. Database Desktop compares the two tables and subtracts matching records.

Rules for subtracting records

Follow these rules when subtracting records:

- The two tables you use in the Subtract operation must have compatible structures. This means compatible fields in the same field order.
  - For a complete chart of compatible Paradox field types, see Table 3.7 on page 51.
  - For a complete chart of compatible dBASE field types, search online Help for the subject "dBASE field types" and the topic "Compatible dBASE Field Types."
- You cannot subtract from the parent table in a referential integrity relationship.

Subtracting on a network

When you use Subtract, Database Desktop must acquire a read lock on the table that contains the records you're subtracting, and a write lock on the table you're subtracting records from. This means other users can't

- Change the contents or structure of either table
- Perform any operation that requires a write or exclusive lock on either table

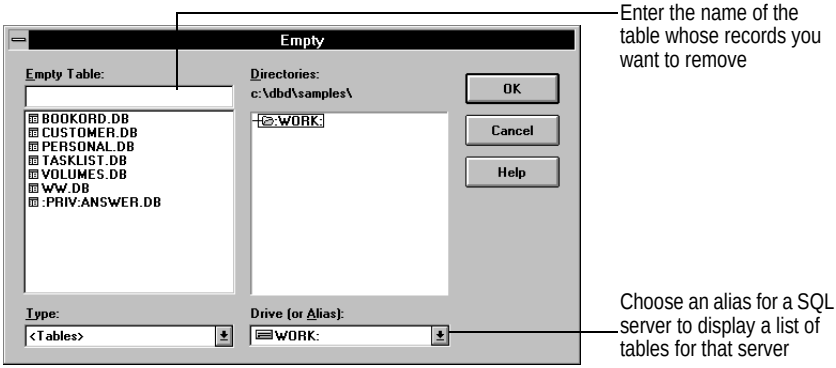
If another user has locked a record or placed a write or exclusive lock on either table, you must wait until the lock is removed before using Subtract.

Emptying tables

Use the Empty utility to remove all records from a table, leaving the table's structure (including all keys, indexes, validity checks, and so on) intact. You can use Empty on Paradox, dBASE, and SQL tables.

To empty a table, choose Utilities | Empty. Database Desktop opens the Empty dialog box, shown in Figure 5.7.

Figure 5.7 The Empty dialog box



All tables in your working and private directories are shown in the list of files on the left. You can use the Directories panel to access files in different directories.

Enter the name of the table you want to empty in the Empty Table text box. When you choose OK, Database Desktop asks you to confirm the Empty operation. Choose Yes to remove all records from the table.

To empty more than one table, hold *Shift* and click the names of the tables you want to empty. When you choose OK, Database Desktop asks you to confirm the Empty operation for each table you selected.

Note You cannot empty a table that is the parent in a referential integrity relationship. You must first either delete the referential integrity (from the child table) or delete the child table.

Note When you perform an Empty operation on a dBASE table, all records in the table are marked as deleted.

Using Empty on a network

When you use Empty, Database Desktop must acquire an exclusive lock on the table. This means

- No user can access the table in any way.
- If any type of lock is open on the table, you must wait until it's released before you can use the Empty utility.

Limiting character sets

By default, Database Desktop (and Windows) uses the ANSI character set to display characters onscreen. Paradox for DOS uses the OEM character set. Although these sets have most characters in common, some characters are in only one set or the other.

If you try to view a table that contains OEM characters that don't exist in the ANSI set, or if you type ANSI characters into a table created with the OEM set, some characters might not appear as expected.

To prevent surprises of this sort, you can choose Table | Strict Translation. This command limits the characters used by Database Desktop to those that are in both the OEM and ANSI character sets.

With Strict Translation on, when you type a character outside the OEM character set, Database Desktop considers it an error, and won't let you leave the field. Additionally, when you start to edit a field, Database Desktop warns you if the field contains OEM characters.

Chapter

6

Running SQL statements

This chapter describes how to use the SQL Editor to enter and execute SQL statements. It describes the SQL Editor and covers the following topics:

- Opening the SQL Editor
- Specifying an alias
- Entering a SQL statement
- Running a SQL statement
- Saving a SQL statement
- Viewing the SQL translation of a QBE query
- Using SQL against local tables

What is the SQL Editor?

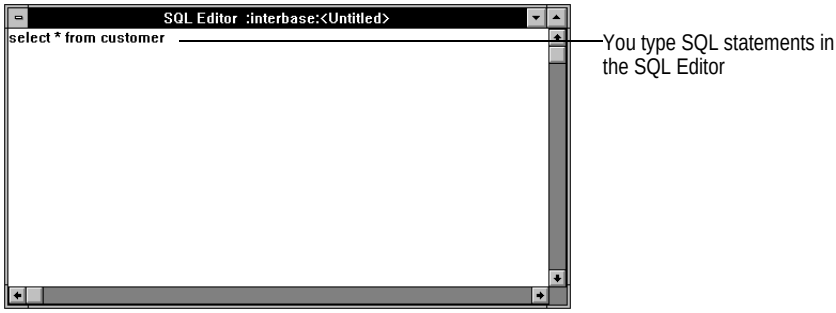
Programmers familiar with SQL can use the SQL Editor window to directly enter, execute, or save a SQL statement. This is sometimes called pass-through SQL. You specify the SQL statement in your server's dialect. The SQL server performs all error or syntax checking and executes the statement without any involvement by Database Desktop.

You can save the SQL statement to a disk file, and then later load, modify, or execute it.

You can also use the SQL Editor to run SQL statements against local tables and to view the SQL equivalent of a query you construct with QBE.

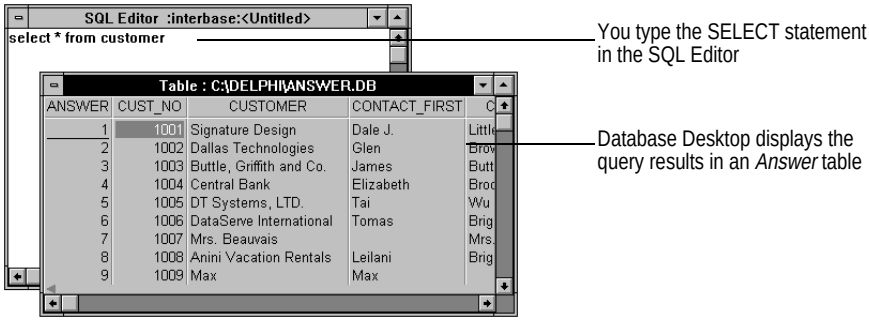
Figure 6.1 shows the SQL Editor.

Figure 6.1 The SQL Editor



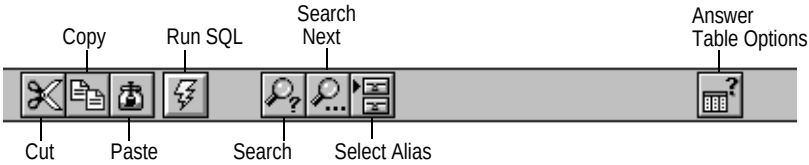
If you execute a SELECT statement in the SQL Editor, Database Desktop displays the resulting data in an *Answer* table, as shown in Figure 6.2.

Figure 6.2 The SQL Editor and an *Answer* table



The SQL Editor has the Toolbar shown in Figure 6.3.

Figure 6.3 SQL Editor Toolbar



Opening the SQL Editor

To open the SQL Editor, do one of the following:

To do this	Do this
Enter (and execute) a new SQL statement	Choose File	New	SQL Statement Or right-click the Open SQL Script Toolbar button and choose New
Open (and edit or execute) an existing .SQL file	Choose File	Open	SQL Statement Or click the Open SQL Script Toolbar button Or right-click the Open SQL Script Toolbar button and choose Open
View the SQL equivalent of an open QBE query	Choose Query	Show SQL Or click the Open SQL Script Toolbar button

Specifying an alias

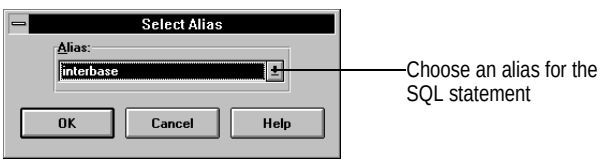
Before running a SQL statement, you must specify the alias that the statement will run against. To specify an alias, do one of the following:

- Choose SQL | Select Alias.

- Click the Select Alias Toolbar button.

Database Desktop opens the Select Alias dialog box, where you can choose one of the aliases you created in the Alias Manager dialog box.

Figure 6.4 The Select Alias dialog box



You cannot include an alias in the text of the SQL statement.

Entering a SQL statement

To enter a SQL statement, type the statement in the SQL Editor. You can enter multiple SQL statements if your server allows it and you include only one SELECT statement.

Use the following commands on the Edit menu to select, locate, and replace text:

Command	Description
Find	Search for strings of text in your code. You can also use the Search button on the Toolbar.
Find Next	Search for the next occurrence of the text you specified using Find. Find Next is dimmed if you have not searched for anything in this session. You can also use the Search Next button on the Toolbar.

Command	Description
Replace	Search for text and replace it with a value you specify.
Replace Next	Replace the next occurrence of the text specified using Replace. Replace Next is dimmed until text has been replaced.
Select All	Select all text in the SQL Editor window.

Running a SQL statement

To run a SQL statement that you have typed in the SQL Editor window, do one of the following:

- Click the Run SQL Toolbar button.
- Choose SQL | Run SQL.

The SQL server performs all error or syntax checking and executes the statement. Figure 6.5 shows a SQL statement in the SQL Editor.

Figure 6.5 SQL statement in the SQL Editor

If your SQL statement is a query, the query results are displayed in an *Answer* table, as shown in Figure 6.2 on page 88.

Saving a SQL statement

To save the SQL statement in the active window, choose File | Save or File | Save As.

When you save a SQL statement to your local hard disk, Database Desktop places it in an unformatted text file with a .SQL extension.

Viewing the SQL translation of a QBE query

Query by example (QBE) provides you with a graphical format of constructing a query. When you use QBE to query a SQL table, SQL Link attempts to translate your query to an equivalent SQL statement and pass it to the SQL server. If successful, the server processes your query, then passes the answer set back to you through SQL Link.

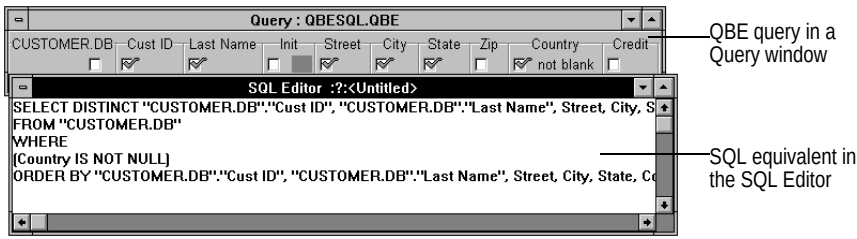
Database Desktop lets you view the equivalent SQL statement for the query at any time during query construction, or after it is processed.

To view the SQL translation for a QBE query, construct the query and do one of the following while the Query window is active:

- Choose Query | Show SQL.
- Click the Show SQL Toolbar button.

Database Desktop opens the SQL Editor and displays the SELECT statement for your query.

Figure 6.6 SQL translation of a QBE query



Note If the SQL database does not support an equivalent SQL statement for a QBE query, a message confirms that the query is processing in the QBE environment. For further information, see your Borland SQL Link *Getting Started* manual.

For detailed information on constructing QBE queries, see Chapter 7.

For more information about queries on SQL tables, search for “SQL queries” in the keyword list in Database Desktop Help, and choose the topic “Using QBE to query SQL data.”

Using SQL against local tables

Local SQL lets SQL users apply SQL statements to local tables through the SQL Editor. Just as it passes SQL statements directly to the server for processing against SQL server data, the SQL Editor passes local SQL statements to Database Desktop for processing against Paradox or dBASE tables.

The SQL Editor transforms SQL Data Definition Language (DDL) statements directly into calls the Borland database engine can interpret. It also transforms SQL Data Manipulation Language (DML) statements into Database Desktop QBE syntax, which then get processed by Database Desktop’s query engine.

For more information about local SQL, search for “local SQL” in the keyword list in Database Desktop Help, and choose the topic “Using Local SQL”.

Chapter

7

Querying table data with QBE

A *query* is a question you ask about information in one or more tables. You can use queries to play "what if?" with your data.

Database Desktop uses a technique called query by example (QBE) to extract and manipulate data in tables. With QBE, you make the query image look like an example of the records you want to search for.

Database Desktop actually *accesses* external database tables so multiple users can edit, add, delete, or change the values in the tables. Queries of multiple relational tables and links to tables can be set up quickly so the data is as current as you need it.

Database Desktop's QBE offers an easy-to-use graphic method for querying databases, especially if you're already familiar with Paradox or dBASE.

This chapter explains how to

- Use the Query window to create and save a query
- Specify which fields you want to see in the answer
- Create exact and inexact matches in a query
- Use operators to perform calculations on field values
- Use example elements to represent values and join multiple tables
- Use reserved words and summary operators to work with tables
- Use set comparison operators to work with groups of data

SQL If you have Borland SQL Links, you can use QBE to view and query tables on SQL servers. For more information, see the Database Desktop Help Contents.

Using the Query window

In Database Desktop you work with queries in the Query window.

- To open a query in the Query window, choose File | Open | QBE Query or right-click the New Query button and choose Open.

- To create a new query in the Query window, choose File | New | QBE Query or click the New Query button in the Toolbar.

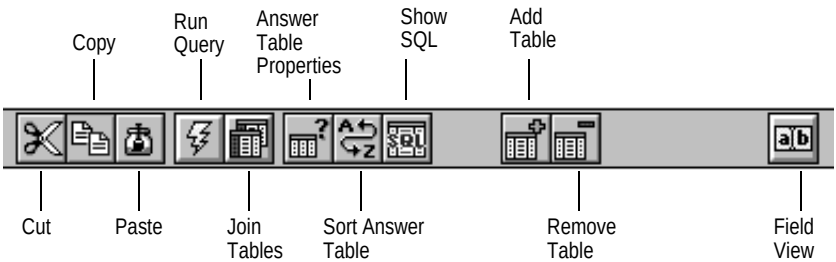
When the Query window opens, the Toolbar changes to display the buttons used for queries.

Because the Query window shares the Database Desktop with the Table window, the Toolbar changes to reflect the operations appropriate for the active window.

Using the Query window Toolbar

The Query window Toolbar buttons are illustrated in the next figure.

Figure 7.1 The Query window Toolbar



Creating a new query

Here are the general steps you'll follow to create a basic query:

1 **Create a new query** by right-clicking the Open Query button or by choosing File | New | QBE Query. The Select File dialog box appears in front of an empty Query window.

2 **Add a table** to the Query window (page 95) by selecting the name of the table you want to query. A table in the Query window appears as a *query image*. A query image contains all the fields in a table, but none of the records.

3 **Define the query** by specifying which fields to include in the answer (page 95) and which records to search for data (page 101). You place a *checkmark* in a field's check box to show values in that field in the answer. You specify selection conditions to tell Database Desktop which records to search.

4 **Run the query** after you define it. The result of the query appears in a *temporary table*. A temporary table is overwritten after each query. To save its data, use the Properties | Answer Table command to save the table with a different name (see page 101).

The following sections describe steps 2 through 4 in detail.

Adding tables

1 In the Select File dialog box, select the name of the table you want to add to the Query window. Drag to select multiple contiguous tables, or hold down *Ctrl* as you click to select several noncontiguous tables.

2 Choose OK.

To add tables to an existing query, activate the Query window, then either click the Add Table button or choose Query | Add Table.

Note If you've rearranged the columns in a Table window by dragging the column titles, your custom view of the table doesn't appear in the Query window.

Arranging tables

You can use the View menu to arrange the tables (query images) in a Query window:

- **Tile Tables** is the default; the tables or query images appear one below the other.
- **Cascade Tables** displays the tables or query images as individual, stacked windows.

Moving between tables

To move between tables in the Query window, click the table you want. You can also use *F3* and *F4* to cycle through all the tables.

Removing tables

1 Click the Remove Table button or choose Query | Remove Table.
2 In the Remove Table dialog box, select the table name to remove.
3 Choose OK.

Including fields in a query

Checkmarks in a query image identify the fields that will appear in the *Answer* table. Each check box in the query image has a check menu with five checkmark types. The different checkmarks produce different results, as described in the next table.

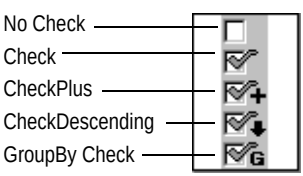


Table 7.1 Checkmark types

Name	Result in the *Answer* table
Check	All *unique values* for the field appear in the *Answer* table, sorted in ascending order (lowest to highest, or A to Z and 0 to 9). No duplicate values are displayed.
CheckPlus	*All values* for the field—including duplicates—appear in the *Answer* table. Because sorting removes duplicate records, the *Answer* table is unsorted.
CheckDescending	All unique values for the field appear in the *Answer* table, sorted in *descending order* (highest to lowest, or Z to A).

Table 7.1 Checkmark types (continued)

Name	Result in the *Answer* table
GroupBy check	Groups records by the values within the field (for use only in a Set query). Unlike other checkmarks, a field with the GroupBy check does *not* appear in the *Answer* table. GroupBy and Set queries are discussed on page 139.
No check	Removes any checkmark type from the field.

Placing checkmarks

1. To place a checkmark in a single field, position the pointer on its check box and hold down the left mouse button.
2. Drag the pointer to select the checkmark you want, then release the mouse button.

As a shortcut, click the check box to place a Check in a field.

To remove a checkmark, click the check box.

Tip To place the same checkmark in every field in a table, use the check box below the table name and select the checkmark you want.

Running the query

To run the active query, click the Run Query button, choose View | Run Query, or press *F8*.

While the query is running, a dialog box displays status information about the query. If there's a problem with the query, an error message appears in another dialog box. When additional information is available, you can use the >> and << buttons to see more information about the error.

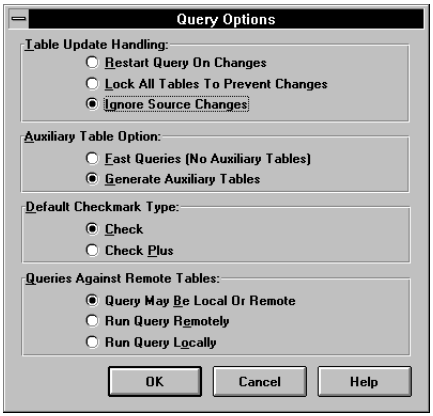
If there are no problems with the query, the *Answer* table appears.

Note While an *Answer* table is open in Database Desktop, you won't be able to run a different query in another Query window until you either close the *Answer* table or rename it (see page 101). This isn't necessary if you are running the same query, or modifications of the same query.

Setting query options

Database Desktop lets you specify query preferences that you can save with the query. Choose Query | Execution Options to open the Query Options dialog box, shown in Figure 7.2.

Figure 7.2 The Query Options dialog box



Handling table updates

When using Database Desktop on a network, multiple users can make changes concurrently to a shared table in a shared data directory. You can choose whether you want your *Answer* table to reflect changes made to the source table(s) of your query *while the query is running*.

- **Restart Query On Changes** makes Database Desktop restart the query when it detects a change to the source table(s).
- **Lock All Tables To Prevent Changes** locks all tables in your query, preventing any changes to them while Database Desktop runs the query. Database Desktop releases the locks when it finishes running the query. (If someone else is already using the table(s) you want to lock and query, Database Desktop can't place your locks. You'll see a message informing you that a table is locked.)
- **Ignore Source Changes** lets other users make changes to the source table(s) while Database Desktop runs your query *and* prevents Database Desktop from restarting the query if they do. (This is the default selection.)

Setting auxiliary table options

Some types of queries generate more than the *Answer* table. For example, DELETE queries create the *Deleted* table and INSERT queries create the *Inserted* table. Creating these extra tables takes a certain amount of time, and you might not be interested in them.

To tell Database Desktop that the only type of generated table you're interested in is the *Answer* table, check Fast Queries from the Auxiliary Table Options panel of the Query Options dialog box.

Tip When you generate only *Answer* tables, your queries will run more quickly.

INSERT queries are discussed on page 124, and DELETE queries are discussed on page 126.

Choosing a default checkmark

Choose whether you want the Check or Check Plus operator to be the default checkmark that Database Desktop places when you either click a field's check box or press *F6*.

Setting SQL options

When creating a query that uses SQL data from a remote database server, you can choose whether you want Database Desktop to process the query locally (on your hard drive) or remotely (on the server). If you choose Query May Be Local Or Remote, Database Desktop decides how the query can be run most efficiently.

Whether you create a query on local (Paradox or dBASE) or remote (SQL) data, Database Desktop can translate your QBE statement into valid SQL syntax. This is done automatically when you query remote data. You can view this SQL syntax by choosing Query | Show SQL or clicking the Show SQL Toolbar button. Database Desktop opens the SQL Editor with the translated SQL syntax in it.

If you prefer writing SQL syntax to creating QBE statements, you can use the SQL Editor to write SQL statements to be run against local (Paradox or dBASE) or remote tables. The only restriction is that QBE must be able to interpret the SQL syntax correctly.

For information on the SQL Editor, see Chapter 6.

For more information on using SQL data with Database Desktop, search online help for the subject "SQL" and the topic "SQL."

Viewing the *Answer* table

The *Answer* table is a temporary table created after a query is run. It is overwritten every time you run a query that creates an *Answer* table.

During installation, a private directory for Database Desktop is assigned to the directory where you installed the client application. The *Answer* table (and any other temporary table) is stored in this directory.

Note By default, the structure of the *Answer* table is based on values in the checked fields in the query (from left to right, and top to bottom). The first field checked in the first query image becomes the first field of the *Answer* table, and so on.

Tip To save the results of a query, you can rename the *Answer* table (see "Modifying the Answer table" on page 99). You can also use the Rename utility to rename ANSWER.DB with a new name so that it isn't overwritten. For information on the Rename utility, see "Renaming objects" on page 64.

Saving the query

To save a query, choose File | Save or File | Save As. By default, the file extension is .QBE. A .QBE file is an ASCII text file that contains all the instructions for running the query.

If you're using Database Desktop in another application, you can run the saved query from there.

Modifying the *Answer* table

You can change an *Answer* table's properties before you run a query. Using the Properties menu in the Query window, you can change

- **Sort order.** Before you run the query, you can choose Properties | Answer Sort to override the default sort order of the *Answer* table. This lets you change the order in which fields are sorted in the *Answer* table, without affecting the display order, or view, of the fields.
- **Name or table type.** Before you run a query, choose Properties | Answer Table to define a new name or data format for the *Answer* table. This lets you save the results of a query in a file that is not overwritten every time you run a query.

Sorting *Answer* tables

The following sections describe the *Answer* table sort order, how to change it, and how to perform the sort.

Sort order of records

The records in the *Answer* table are first sorted on the basis of the *table order* of its fields, then the *values* in its fields from left to right.

The table order is the default order of the fields as they are defined in the *structure* of the table—how the table and its fields are defined. This is distinctly different from the *view* (or onscreen image) of the table, query image, or *Answer* table.

The records are first sorted by the values in the first field (again, first in the table order). Then, any ties are settled by the values in the second field, and so on.

To change the display order of fields in the *Answer* table, rotate the columns in the query image by dragging their titles (just as you do in the Table window). To override the default sort order of field values in the *Answer* table, use Properties | Answer Sort (discussed in the next section).

Using ascending order

When a field in a query image has a Check in it, Database Desktop sorts records in that field in *ascending* order:

- Alphabetically from A to Z
- From lowest to highest number
- From earliest to latest date

Using descending order

To sort records by *descending* order—the opposite of ascending order—place a CheckDescending check in the field.

Note Sorting in either ascending or descending order always removes duplicate records. When you use CheckPlus to retain duplicate records, the records in the *Answer* table are not sorted.

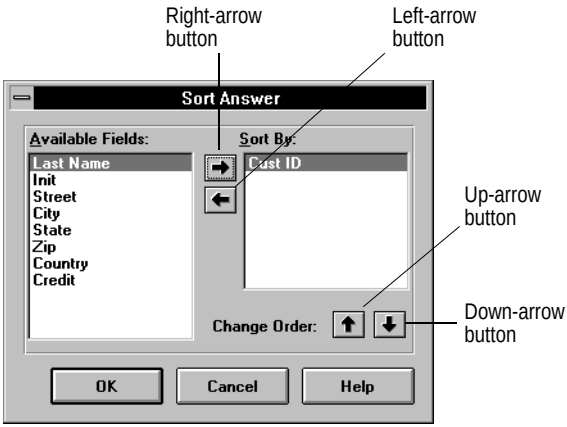
Tip To increase the speed of a query, try replacing Checks with CheckPlus marks. CheckPlus queries are faster because the results aren't examined for duplicates nor are they sorted.

Performing the sort

1 Complete the query definition in the Query window.

2 Choose Properties | Answer Sort. The Sort Answer dialog box appears, as shown in Figure 7.3.

Figure 7.3 The Sort Answer dialog box



3 Select a field to sort the *Answer* table by, from the Available Fields list.

4 Choose the right-arrow button to move the field to the Sort By list.

5 Repeat steps 3 and 4, adding the fields in the order you want the *Answer* table sorted by.

6 To remove a field from the Sort By list, select it, then choose the left-arrow button.

7 To rearrange the order of the fields in the Sort By list, select a field and then use the up- or down-arrow buttons to move it up or down in the list.

8 When you finish, choose OK.

9 Run the query.

Note If a query is undefined or has syntax errors, the Sort Answer dialog box is unavailable.

Changing the view of an *Answer* table

While the *Answer* table is active, you can change column order and size, or row height, using the mouse just as you do in the Table window (see "Customizing a table view" on page 71). Then you can use the Properties menu to save, restore, or delete its properties:

- **Save** writes the column arrangement and width, row height, scroll locks, and position of the table title to the ANSWER.TV or .TVF file.

  Saving the *Answer* table properties lets you run the query several times in a row without setting the properties each time.

- **Restore** resets the *Answer* table properties to the last saved version. This is useful for restoring the *Answer* table after experimenting with changes to its properties.

- **Delete** erases the ANSWER.TV or .TVF file, and then removes the custom property settings from the *Answer* table.

Answer table properties are overwritten when a different query is run.

Changing the name or table type

To save the results of a query, rename the *Answer* table before running the query. At the same time, you can change the table type of the *Answer* table to dBASE or Paradox. These settings take effect the next time the query is run.

Renaming an *Answer* table

1. Make sure the Query window is active, then choose Properties | Answer Table.
2. In the *Answer* Table dialog box, specify a new name for the *Answer* table.

Changing the *Answer* table type

1. Make sure the Query window is active, then choose Properties | Answer Table.
2. In the Answer Table dialog box, specify dBASE or Paradox for the data format of the *Answer* table.

Defining queries in the Query window

To define query statements that narrow a search to specific records, you type *selection conditions* into fields in the query image.

For example, in a query of the *Customer* table, placing a Check in the State field returns *all unique* state values, even if they're blank. Compare this with the *Answer* table for a similar query in which you *also* type the selection condition `CA`. (Only records with the value CA in the State field will be in the *Answer* table.)

To view long selection conditions, you can resize a column in the query image by dragging its right border (just as you can in the Table window). Each field in a query image can hold up to 255 characters.

Selection conditions can contain any combination of reserved symbols or words, values, or operators. For example, query statements can locate records based on

- An exact match
- A range of matching values
- An inexact match
- A pattern

When you create a query, be aware that certain characters and words are reserved for special uses. The following table lists these reserved symbols and words—as well as various kinds of query operators—and provides cross references to sections that describe these items in more detail.

Table 7.2 Reserved symbols and words, and query operators

Category	Item	For more details
Reserved symbols	Check CheckPlus CheckDescending GroupBy check	See page 95
Comparison operators	= > < >= <=	See page 106
Wildcard operators	.. @	See page 109
Special operators	LIKE NOT BLANK TODAY	See page 111
	, (AND) OR AS	See page 111
	! (inclusive)	See page 140
Arithmetic operators	+ – * / ( )	See page 112
Reserved words	CALC INSERT DELETE CHANGETO FIND SET	See page 121

Table 7.2 Reserved symbols and words, and query operators (continued)

Category	Item	For more details
Summary operators	AVERAGE COUNT MIN MAX SUM ALL UNIQUE	See page 129
Set comparison operators	ONLY NO EVERY EXACTLY	See page 136

Including punctuation

To include punctuation marks and other reserved characters in a query statement, enclose the value in double quotation marks (" "). Quotation marks identify characters as *literal* characters instead of as part of a reserved word or operator.

Blank spaces in values don't need to be enclosed in quote marks.

For quotation marks and backslash characters to appear in the *Answer* table, precede them with a backslash (\).

Table 7.3 Using backslash (\) characters

Character	Example of use
"	Literal quotation marks in the value must be preceded by a backslash (\). For example, when you type "`Thomas E. \"Ned\" Lawrence`", Thomas E. "Ned" Lawrence appears in the *Answer* table.
Backslash (\)	Literal backslash (\) characters in the value must be preceded by another backslash (\\).

Typing numbers into queries

When you type a number into a query, you can ignore the format specified for the field unless you're specifying an exact match. If you include a comma for a whole number separator in the numeric value, it can cause an error because the comma is the AND operator (see page 111).

To search for numbers with decimal portions, include the decimal point in the example you type. For example, to find all values that match 4150.5, type `4150.5` in the field.

Editing queries

Editing queries is similar to editing table values, except there is no Edit Data button or corresponding Edit mode.

When you select a field that contains a query condition, any characters you type are added at the end of the existing condition. To revise a query condition, use field view or persistent field view, as described in the next sections.

Tip To cancel the last change you made in a query image, press *Esc* before moving off the field.

Using field view

To enter field view, select the field, then use one of the following methods:

- Click the Field View button in the Toolbar.
- Press *F2*.
- Click the field again (or double-click an unselected field).

In field view, you move the insertion point as you usually do when editing text. Actions you can perform with the keyboard in field view are listed in "Keypad shortcut keys" on page 152.

To stay on the current field and exit field view, use any of the actions you used to enter field view. When you click another field or press *Tab* or *Shift+Tab*, you leave the field and exit field view.

To select another field and remain in field view, use persistent field view, described next. For more information about field view, see page 74.

Using persistent field view

Press *Ctrl+F2* to enter persistent field view. When you press *Tab* or *Shift+Tab* to move to another field, you remain in field view.

Ctrl F2 To exit persistent field view, press *Ctrl+F2*.

See "Keypad shortcut keys" on page 152 for a list of keyboard actions in field view.

Using different checkmark types

The result of a query depends on which checkmarks are used to perform the query. (This is called *precedence*.) Checkmarks with the highest precedence override checkmarks with lower precedence, as shown in the next table.

Table 7.4 Checkmark precedence

Operator	Description	Precedence
CheckPlus	Show duplicate records, and don't sort the records	3 (highest)
Check	Show unique records only, and sort in ascending order	2
CheckDescending	Sort in descending order	1
GroupBy check	Groups records in a Set query	1

Note Checks take precedence over CheckPlus marks only when they both appear in the same field of a query with multiple lines.

Displaying duplicate records

Each row of fields that appears in the *Answer* table is treated as a *single record*. This record can contain one field or several fields.

A *duplicate record* is a record that contains all the same values, in each field, as another record. If a query is displaying only a few fields in the *Answer* table, it's possible that some valid records will not appear when you use Checks to display the fields.

For example, in the sample *Bookord* table, there are 10 records of orders for customer number 1784 (McDougal). If you place Checks only in the Cust and Date fields, only 6 records will appear for this customer in the *Answer* table (because 4 of the records are duplicates). To see all orders for this customer, place Checks in all fields in the query image, or use the CheckPlus marks.

When CheckPlus is in *any* field in a query image, it assures that all records for that query, including duplicates, appear in the *Answer* table.

Changing the sort order

CheckDescending, unlike CheckPlus, does not affect the entire record. Only the field in which you place a CheckDescending check is sorted in descending order.

When multiple CheckDescending marks are placed in a query image, the records are sorted in descending order based on the position of the fields in the query image. The leftmost field in the top line of the query image is the field that is sorted first. After that, duplicate values in the other fields are sorted based on their order from top to bottom and left to right in the query image.

The fields in the query image can be rearranged to place specific fields to the left of others. This lets you view widely separated fields in the query image, and has no affect on the order in which fields appear in the *Answer* table.

Matching exact values

Usually you'll want to restrict the results of a query so you see only certain records in a table.

To search for only those records with a specific value in a field, type the value into the field in the query image. Exact matches are case sensitive; only values that are *exactly the same* as the value you type will appear in the *Answer* table.

For example, to find records for customers located in California, you type `CA` in the State field of the *Customer* query image. To see the customer names in the *Answer* table, place a Check in the Last Name field.

To locate more specific records, you can type values into several fields. Only the records that match all typed values will appear in the *Answer* table.

For example, to find records for customers located in San Francisco, California, type `San Francisco` in the City field (in addition to typing CA in the State field, as mentioned earlier).

Typing values in queries

1. To include a field in the results of the query, place a checkmark in the field's check box. The insertion point appears beside the check box.
2. Type the exact value you want to find.
3. To modify a value using standard text-editing keys, click the Field View button in the Toolbar, choose Query | Field View, or press *F2*. To edit the entry in another field, double-click it.
4. Repeat steps 1 and 2 until you finish the query.

Matching ranges of values

To see records that match a *range* of values—not just one value—you can use *comparison operators.* Comparison operators work with the following field types:

- **Paradox**: Alpha, Number, Money, Short, Long Integer, BCD, Date, Time, Timestamp, Logical, and Autoincrement fields
- **dBASE**: Character, Float, Number, Date, and Logical fields

An error appears if any operators are used to search dBASE logical fields.

Using comparison operators

To use a comparison operator, type it *before* the value you're interested in. These operators are listed in the next table.

Table 7.5 Comparison operators

Operator	Meaning
=	Equal to (optional)
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

For example, to find records in the sample BOOKORD.DB table with ordered quantities of 20 or more, place a Check in the Quant field, then type `>=20`. (A space between the operator and the number is optional.)

Ranges of values with upper and lower limits can be specified by combining comparison operators with other operators, such as the AND (,) operator (see page 111).

Tip When you search for numbers stored in scientific notation, use the greater than (>) or less than (<) operator. When you search for logical values, use only f, F, t, T, false, or true as search conditions.

Matching inexact values

When you don't know the exact value you want to match, type one of the following operators before a value in the query image:

- **LIKE** locates records that are similar to, but not necessarily identical to, a particular *string* value (character values in dBASE, alpha values in Paradox).
- **NOT** locates records that exclude the value you specify.
- **BLANK** locates all records without values in that field. This operator does *not* require a value after it.
- **TODAY** locates all records with today's date, and can be used with other operators to locate records relative to today.

The following sections describe these operators in greater detail.

Using the LIKE operator

If a query isn't producing the results you expect, try using LIKE to see if the problem is a spelling error of the value in the query image.

For example, to find the customer named Aberdeen in the sample table CUSTOMER.DB when you can't remember how it's spelled, you could type `like Aberd` in the Last Name field of the query image. This finds all last names similar to Aberd.

There are two general rules to remember when using LIKE:

- The first character in the value you type must match the value you're looking for (case doesn't matter). For example, `like Kalifornia` does not match *California*.
- When the value you type includes half or more of the characters in a value, in the correct order, you'll probably get a match. For example, `like lon`, `like ldn`, `like lnd`, and `like loo` all match *London*. But `like lo` and `like ln` do not match *London*.

The LIKE operator is only used to find string values.

Using the NOT operator

To select records that do *not* meet a specific condition or contain a specific value in a field, use the NOT operator.

When NOT precedes a selection condition, it reverses the result. NOT can precede other operators, exact values, ranges, or *wildcards* used in matching patterns (discussed on page 109).

To use the NOT operator, type `not` before the example of the value you don't want to see. All values—including blanks—that *do not match* the value appear in the *Answer* table.

For example, you can use the NOT operator to find all orders placed by customers who are located in states other than Maryland. To create this query of the *Bookord* table, check the fields in the query image that you want to see in the *Answer* table, then type `not MD` in the State field.

Using the BLANK operator

To find records without a value in a specified field, use the BLANK operator. Sometimes, the *absence* of a value is a useful piece of information. Or, you might want to find records with a blank field so you can fill in information that wasn't available when the record was originally entered.

To use the BLANK operator, type `blank` in the appropriate field.

Tip You can combine NOT with BLANK to find all records that have *any* value in the specified field.

Using the TODAY operator

To locate records based on a date field, use the TODAY operator. For TODAY to work correctly, make sure that your computer's calendar is set to the correct date.

The arithmetic operators + (addition) and – (subtraction) can be used with the TODAY operator to do the following types of date calculations:

- Add a number (of days) to a date
- Subtract a number (of days) from a date
- Subtract a date from a date (the result is a number of days)

Examples of arithmetic operations on dates are shown in the next table.

Table 7.6 Arithmetic operations on dates

Expression	Meaning
< TODAY	Finds dates earlier than today's date
< TODAY – 90	Finds dates earlier than 90 days ago
TODAY + 30	Finds dates 30 days after today's date

Matching patterns of characters

To match patterns of characters in your query examples with more flexibility than the LIKE operator offers, use *wildcard operators*. The .. wildcard operator and the @ wildcard operator are described in the following sections.

When you use a wildcard to find a date, the pattern you define with the wildcard operator must reflect the current date format (set in the BDE Configuration Utility).

Note Changing the way data displays in the Windows Control Panel changes only the *view* of the data. To query data in tables after changing Windows Control Panel settings, make sure you change your BDE settings to match, using the BDE Configuration Utility. Double-click the BDE Configuration Utility icon in the program group where the Database Desktop icon appears. See online Help in the BDE Configuration Utility for more details.

Matching a series of characters

The .. wildcard operator matches any series of any number of characters, including blank spaces. The next table shows what happens when the .. wildcard operator is used in a variety of ways.

Table 7.7 The .. wildcard operator

Pattern	Matches these field values
G..	*Grant*, *glitch*, *Georgia* (notice that this operator is not case sensitive)
g..t	*Grant*, *gross weight* (notice the space character matches)
..T	*hat, Elm St*
...e..s	*Thomas Edward Willis, roses*
7..5	7485, 70005
6/..71	6/01/71, 6/30/71

Matching single characters

When you know how many characters are in the entry you're looking for, use the @ wildcard operator. The @ wildcard operator matches any single alpha character. Any number of @ characters can be used to specify a pattern.

Table 7.8 shows what happens when the @ wildcard operator is used in a variety of ways.

Table 7.8 The @ wildcard operator

Pattern	Matches these field values
m@@e	*Mike, more, made*
wom@n	*woman, women*
S@@@@	*Smith, Smyth, scent*
19@4	1934, 1954, 1994

Using multiple conditions in queries

When you enter more than one selection condition on the same line of a query image, *all* of the conditions must be met before a record appears in the results. This is called a

logical and because only those records that match the first condition and the second condition (and any other conditions) appear in the results.

There are situations when you might want to select records that meet either the first condition or the second condition (or any other conditions, if there are any). This kind of query statement is called a *logical or*; to create it, you can use either

- **Separate lines.** The next section describes how to enter selection conditions for different fields on separate lines of the query image.
- **The OR operator.** To search for more than one selection condition in a single field, see page 111.

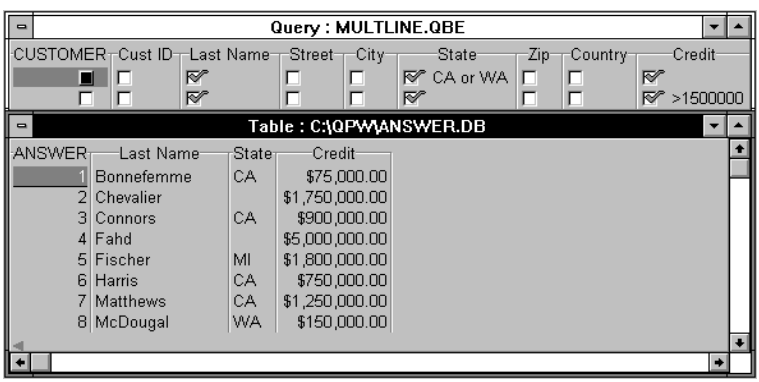
Using multiple lines

To search for records that meet one of multiple selection conditions, you can enter the conditions on separate *lines*, or rows, of the query image.

Each line in a query image defines a search criterion that is independent from any other line. Example elements (see page 116) are a unique situation because the value(s) they represent are usually defined on one line, then used on a separate line of the query image.

For example, the *Answer* table for a two-line query includes any records that match *either* line 1 *or* line 2. The next figure shows a query that uses two lines to search for customers who either live in California or Washington, *or* who have a credit limit over $1,500,000.

Figure 7.4 Multiple lines in a query image



Note In general, all lines must have checkmarks in the same fields for this kind of query to work.

Creating additional lines

To insert a line above the selected line in a query image, press *Ins*. There are two alternate methods for creating additional lines:

- If a query image has a single line, select any field in the query image, then press the down-arrow. If a query image has multiple lines, select any field in the bottom line of the query image, then press the down-arrow. Repeat as needed.
- Select any field in the query, press *End* to go to the last field in the table, then press *Enter* to create an additional line in the query image.

Deleting lines

1 Select the line you want to delete.

Ctrl Del 2 Press *Ctrl+Del* to remove it from the query image.

Note *Ctrl+Del* deletes the line and any query statements in it, and cannot be undone.

Using multiple checkmarks

Checkmarks in queries with multiple lines follow the same precedence as in queries with single lines. When several checkmark types appear in a field that has multiple lines in the query images, the highest precedence checkmark overrides the others.

Using special operators

To ask specialized questions about the data in your tables, use a special operator:

- AND (,)
- OR

To specify a new name for a field in the *Answer* table, use the AS operator. These operators are described in the following sections.

Using the AND (,) operator

To enter multiple conditions in a field and require that they all be met, separate the conditions with commas. Using the comma to separate conditions is called a *logical and*, meaning all conditions must be met for a match to occur.

For example, to find only items in the sample BOOKORD.DB table with ordered quantities between 20 and 40 inclusive, you could place a Check in the Quant field, then type `>=20,<=40`. (Spaces between the operators and the numbers are optional.)

Notice that conditions in separate fields are also treated as *logical and* conditions. Each of the conditions in all of the fields in a query image must be met before a record can appear in the *Answer* table.

Note When a value has a comma (such as Fogg & Peat, Inc), remember to enclose the entire value in quotation marks.

Using the OR operator

To enter multiple conditions in a field and match all values meeting *any* condition, use the OR operator. This is the *logical or* operation.

To use the OR operator in a field of the query image, type `or` between the values you want to find. Make sure to leave a space before and after the OR operator.

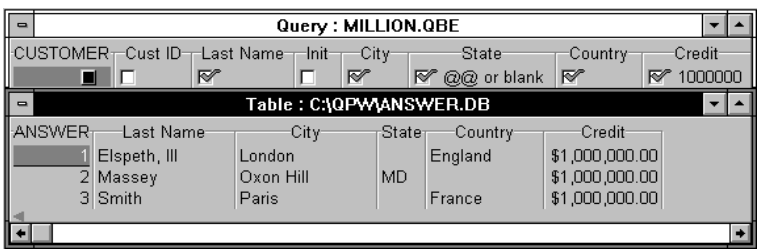
For example, to have customer records from both California and Washington appear in the *Answer* table, type `CA or WA` in the State field of the CUSTOMER.DB query image.

Combining OR with AND conditions

To combine OR operations with AND in a single query, define a query statement such as the one shown in the next figure. One field uses the OR operator, and the other field has an exact value to match.

The values in the *Answer* table meet both the OR requirement in the State field and the AND requirement between the State and Credit fields.

Figure 7.5 Combined OR and AND operations



Using the AS operator

By default, a field you check in a query appears with the same name in the *Answer* table. To change the field name only as it appears in the *Answer* table, use the AS operator, as follows:

1. Type the query statement (if any) in the field.
2. Type `as` followed by a space, then the new field name you want.
3. Run the query. The new field name you specified appears in the *Answer* table.

A field name specified in an AS query statement can contain an expression. It cannot contain an example element.

Tip If you want to include reserved characters or words in a new field name, enclose the field name in double quotation marks.

Using arithmetic expressions

In number, money, date, short, long integer, and BCD fields, you can use arithmetic expressions in a query example.

The result of an expression depends on the order in which the arithmetic operations are performed. (This is called *precedence*.)

Operations with the highest precedence are performed first; operations with equal precedence are calculated from left to right. Use parentheses () to combine and group operations and indicate which calculations should be performed first.

This table lists the arithmetic operators and the precedence assigned to each.

Table 7.9 Arithmetic operators

Operator	Description	Precedence
*	Multiplication	2 (highest)
/	Division	2
+	Addition (or concatenation of strings)	1
–	Subtraction	1
( )	Groups expressions	(Overrides)

Arithmetic operators are especially useful with the following:

- TODAY operator (+ and – only) (see page 108)
- CALC operator (see page 121)
- Example elements (see the following section)

Using example elements

An *example element* is *not* a literal value you type, it's a variable (or placeholder) that represents all values or a subset of values in a field.

Example elements give you great flexibility because they can represent

- All values in a field
- A subset of values in a field
- One specific value in a field
- Joined fields in multiple tables

The following figure shows how you can use example elements in single-table queries. For samples of example elements in multi-table queries, see page 118.

Figure 7.6 Example elements in single-table queries

Note The *name*, or label, of an example element has no relation to the value it represents. To remind yourself of the values an example element represents, use a meaningful name. In the above figure, **EGall** represents all values in the field, **EGsubset** represents a subset of values, and **ConCredit** represents the amount of credit for the customer named Connors.

Creating an example element

F5 To toggle on the placement mode for example elements, click the Join Tables button in the Toolbar. Or, to create a single example element, press *F5* or type an underscore (_) before you type its name. If you type the name of an example element, or edit it, remember to use

- **A unique name.** The name that identifies an example element need only be unique and use valid characters. (The example element *name* doesn't appear in the *Answer* table.)
- **Valid characters.** Any alphabetical character (A to Z) or number (0 to 9). An example element *cannot* have a space in it, nor any reserved characters (such as a comma, +, –, *, !, or /).

Example elements appear highlighted or in a color different from other text you type in a query statement. If you have a color monitor, example elements appear as red text.

Using the Toolbar

To create example elements with the Toolbar, use the Join Tables button. The Join Tables button toggles placement mode on and off. Because placement mode creates *two* example elements, it's typically used to link, or join, two tables in a query (see page 118).

Every time you click the Join Tables button, the number part of the current example element name increases by one.

When you click the Join Tables button, you enter the example element placement mode and the Joining indicator appears in the status line. Your next *two* clicks each create an example element with the same name. After the second example element is created, the pointer returns to normal mode.

If you click twice to create two example elements in the same field, the AND (,) operator is inserted between the example elements. This is useful when you want to use an example element in a calculation (see page 117).

Note If you use the Join Tables button to create a single example element, be sure to click the Join Tables button again after you create the example element. This toggles off placement mode and the status line indicates placement mode is canceled.

Using the keyboard

1. Select a field you want to create an example element in.
2. F5 Press *F5* or type an underscore (_), then type the characters you want to use. The underscore causes the next group of characters to be treated as an example element; it doesn't appear in the field.
3. To add more text in the same field, press *Spacebar* or type a comma, a dash, or an underscore when you finish typing the example element. Subsequent characters you type appear as regular text.
4. Repeat steps 1 through 3 until all example elements are created.

Editing example elements

To edit an example element, make sure the pointer is in normal mode. To use standard text-editing keys to modify the example element, use field view.

To cut, copy, or paste example elements, use the buttons in the Toolbar or the Edit menu commands. To select an example element, first select the field it's in, then double-click the example element.

Note If you delete the leftmost character in an example element's name, the text reverts to normal text (the color changes). To restore the example element formatting, position the pointer in front of the first character, then press *F5* or type an underscore (_).

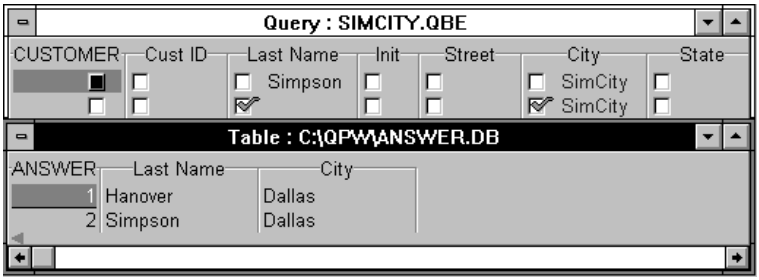
Defining the value of example elements

By default, the value of an example element is all values in a field. Typically, when you use example elements to simply join tables in a query, you want the example elements to represent all values in the common field.

To narrow a search using example elements, you can refine the values the example element represents by defining its value. To define the value of an example element, you create selection conditions in other fields on the same line of the query image. Once an example element is defined, you can use it to search for the same values in those fields.

For example, to list all customers who live in the same city as the customer named Simpson, you can find the city *and* all matching cities in one step, as shown in the next figure.

Figure 7.7 Using an example element to match a specific value



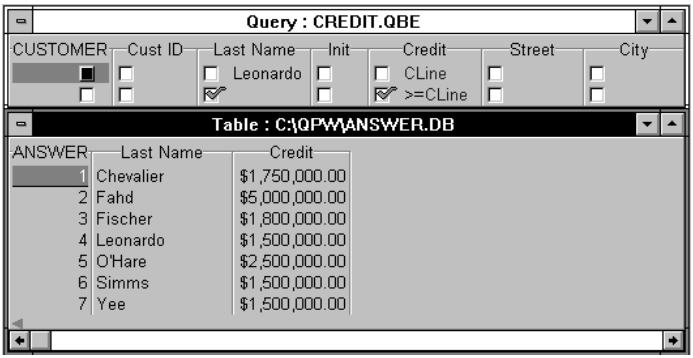
In the City field is an example element named **SimCity**, which could represent all cities in that field if no other selection conditions were in the query image. However, the customer name Simpson in the Last Name field defines **SimCity** as the city where that particular customer is based. Below the line that defines **SimCity** is another line that has Checks in the fields that will appear in the *Answer* table.

Note Fields with example elements don't have to be checkmarked if you don't want to see them in the *Answer* table. In the previous figure, any of the fields in the second line could have been checkmarked.

Defining the value as a range

To define an example element as a range of values, you create a query with multiple lines (see page 110). For example, look at the next figure. The first line in the query image retrieves the amount of credit ($1,500,000) for the customer named Leonardo and assigns this value to the example element **CLine**. The second line in the query image uses the value of **CLine** to retrieve all records for customers with credit amounts equal to or greater than $1,500,000.

Figure 7.8 An example element for a range of values



Calculations using example elements

To perform a calculation in a field, you can create an example element (which defines the value to use) so you can use a copy of the example element (which describes what to do with the value).

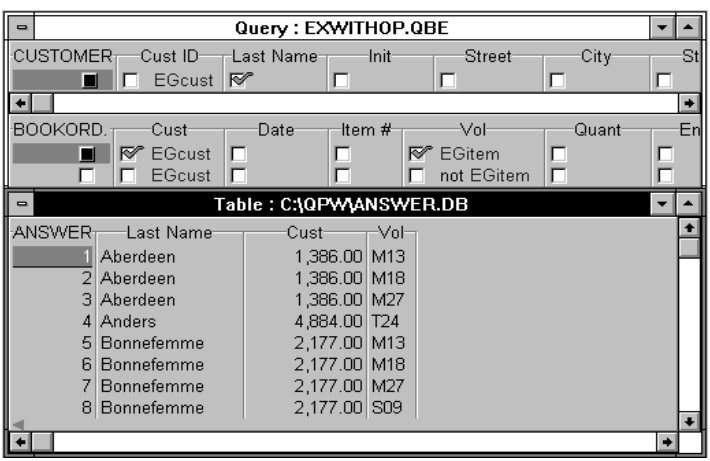
You can use example elements with arithmetic and other operators, such as AND, NOT, LIKE, and + (concatenation). You cannot use example elements with the OR operator.

Using NOT or LIKE with an example element

To list all the customers who have ordered two or more different items, you can create a query like the one shown in the next figure. This query uses two example elements: one example element (**EGitem**) represents the specific item ordered, and the other (**EGcust**) represents the customer who ordered it.

The second line in the *Bookord* query image assures the order is from the same customer (**EGcust**), and the expression `NOT EGitem` assures the order is for a different item.

Figure 7.9 Combining example elements with operators



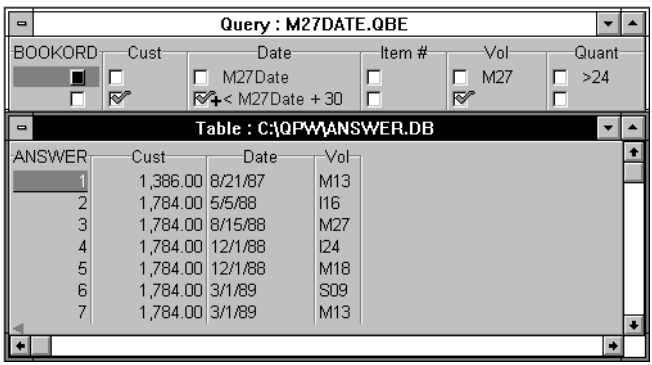
Tip To only see the customer names and their identification numbers in the *Answer* table, omit the Check in *Bookord's* Vol field.

Using example elements with dates

To use an example element in a date expression, you construct a query to list those records with a specific time relationship to a date value.

For example, to list all orders made less than 30 days after the order of item M27, create a query as shown in the next figure.

Figure 7.10 Example elements and dates



This query uses an example element to represent the date the item M27 was ordered. The arithmetic expression calculates the date 30 days after the order, and the < (less than) operator selects the records prior to that calculated date.

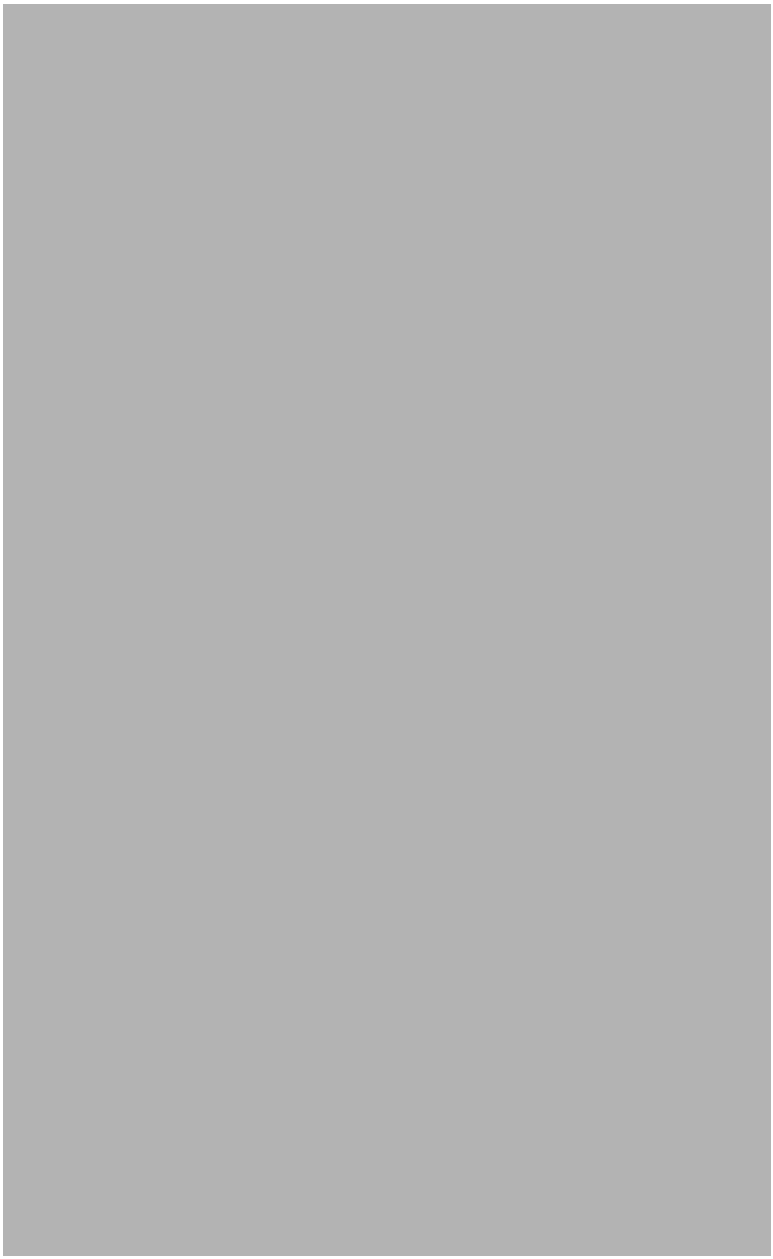
Using multiple tables in a query

Up to 24 separate tables in a query can be joined through their *common fields*, fields in each table that contain the same kind of information. To join any tables with common fields, you place the same example element in the common field shared by each of the tables.

For example, the sample files *Customer*, *Bookord*, and *Personal* each have a field containing customer ID numbers (called Cust ID in *Customer* and *Personal*, and Cust in *Bookord*). Also, the sample files *Bookord* and *Volumes* both have a field containing volume ID numbers (called Vol). Examples of joins that use these common fields are shown in the next figure.

Notice that multiple example elements placed in the same common field must be separated with a comma (the AND operator). The Join Tables button creates the comma as well as an example element when you click a field that already has an example element in it.

Figure 7.11 Example elements used to join tables

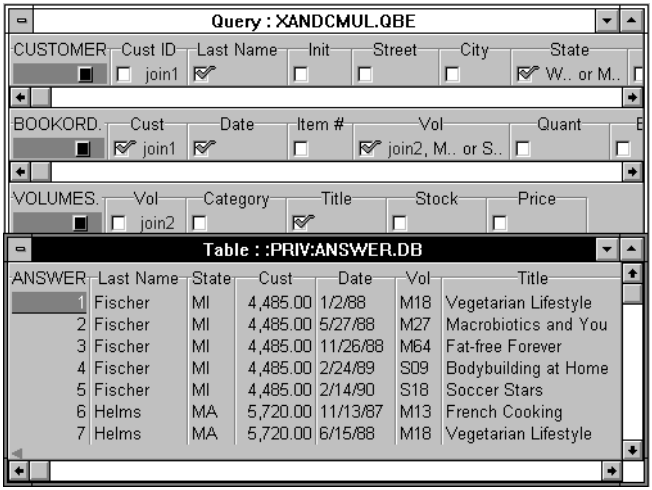


Using selection conditions

When you use example elements to join tables, you can also add multiple selection conditions to any query image.

The following figure shows the use of example elements to join three tables with multiple conditions.

Figure 7.12 A multi-line, multi-table query



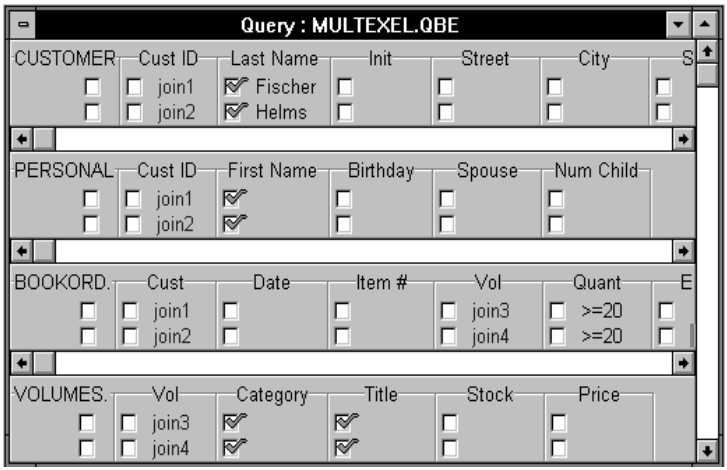
When you enter example elements in multi-line, multi-table queries, be sure to join the corresponding lines in the query images. For example, note the Checks on the lines in the previous figure.

Asking questions about multiple tables is similar to asking questions about a single table. You can use the OR operator when you want to set alternative conditions in the same field. Do this as if you were working with a single table.

As with single tables, when you use the OR operator in different fields of a multi-table query, or define more than one set of OR conditions, you must enter them on separate lines of the query image.

Note When a query has multiple lines and multiple tables, unique example elements must be used for an OR operation across multiple lines. These example elements must also be placed on separate lines in the other tables, as shown in the following figure.

Figure 7.13 Example elements on multiple lines



Calculating with queries

With QBE, you can do more than ask questions about the values in tables. You can use the CALC reserved word to perform the following calculations on field values:

- Construct and evaluate mathematical expressions
- Combine values from two or more fields
- Combine field values with constants

The *Answer* table generated by a CALC query contains an additional field for the calculated result. Because of this, it doesn't matter in which field of the query image you type the CALC expression, and it's unnecessary to place a checkmark in the field.

Calculating math expressions

When using CALC with the arithmetic operators +, –, *, /, and ( ), you can also use

- Constants (like *154* or *7/12/91*)
- Example elements

Combining calculations

Because example elements represent values, you can use them to perform calculations on the values they represent. This means instead of separately locating values and performing calculations on the results, you can do it in a single query with example elements.

For example, look at the following figure. To increase the amount of credit for each customer in the *Customer* table, you can use an example element and the multiplication operator (*). You could phrase this in English as "Multiply all current credit amounts by

1.15." In this sentence, "all current credit amounts" acts as an example of the values you want to change. It is, in effect, an example element.

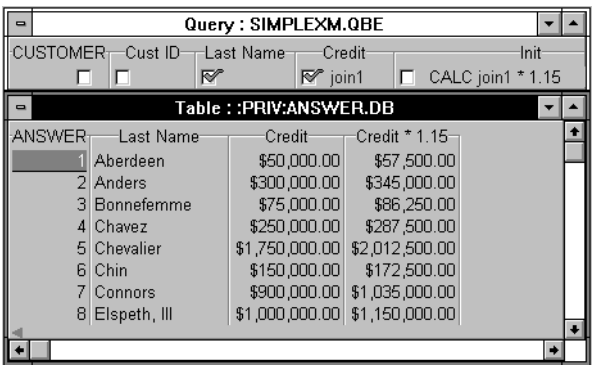
To phrase the increase as a query statement, you enter

```
CALC CLine * 1.15
```

in any unused field in the *Customer* query image. The first instance of **CLine** in the Credit field defines the example element as representing all values in the Credit field. The second instance of **CLine** uses the example element in the mathematical expression to multiply all values in the field by 1.15.

Note To perform a calculation on an example element, the example element must be defined so it can be used.

Figure 7.14 A simple example element



Tip To show the calculated increase in the credit amount *without* the existing credit amount, omit the Check in the Credit field.

Using CALC with alphanumeric values

You can combine (*concatenate*) alphanumeric values and constants using CALC and the + operator. Constant values that are also Database Desktop reserved characters, such as commas and spaces, must be enclosed in double quotation marks to be treated as literal characters (see page 102).

For example, you can combine values from several fields into a single field and add the appropriate punctuation, as shown in the next figure. Notice that **A**, **C**, **S**, and **Z** are example elements.

Figure 7.15 Using CALC to concatenate values

Query : USA-ADDR.QBE

CUSTOMER	Last Name	Street	City	State	Zip	Country
☐	☑ CALC A+","+C+","+S+","+Z AS Address	☐ A	☐ C	☐ S	☐ Z	☐ blank

Table : C:\QPW\ANSWER.DB

ANSWER	Last Name	Address
1	Aberdeen	45 Utah Street,Washington,DC,20032
2	Bonnefemme	128 University Drive,Stanford,CA,94323
3	Chavez	Cypress Drive,Palm Springs,FL,32938
4	Connors	27 Portfolio Drive,Belair,CA,90026
5	Fischer	14 Willow Lane,Birmingham,MI,48011
6	Hanover	15 State Street,Dallas,TX,75043
7	Harris	Old Country Road,Atherton,CA,94322
8	Hawes-Anderson	Waves Cottage,Palm Springs,FL,32382
9	Helms	52 Brattle Street,Cambridge,MA,02138

Performing table operations

The results of all queries described so far in this chapter use

- Checkmarks to define the fields that appear in the *Answer* table
- The queried table(s) to locate records, without affecting the tables

The following table describes reserved words, which perform operations that do *not* use checkmarks and *do* affect tables in the query.

Table 7.10 Reserved words for table operations

Name	Description of result
INSERT	Inserts records from one or more tables (called the *source tables(s)*) into a *target table* (the single table that receives the records from the source table). Source tables are unaffected by the INSERT query. The target table must already exist before it can be added to the Query window. After the query is done, a list of the records that have been inserted into the target table appears in the Inserted table.
DELETE	Removes whole records (*not* specific values within records) from a table. This query type is appropriate when the records to delete have something in common that can be specified in selection conditions. After the query is run, a list of the records deleted from the table appears in the Deleted table.
CHANGETO	Alters values in a table based on conditions you specify in a query. CHANGETO offers a kind of global search and replace capability, and is useful when you want to change many values in a similar way. After the query is run, a list of the records changed by the query appears in the Changed table.

Note You can use INSERT, DELETE, and CHANGETO only in these field types:

- **Paradox**: Alpha, Number, Money, Short, Long Integer, BCD, Date, Time, Timestamp, and Logical
- **dBASE**: Character, Float, Number, Date, and Logical

If a problem occurs while a query is inserting, deleting, or changing records, the data is saved in temporary tables called ERRORINS.DB, ERRORDEL.DB, or ERRORCHG.DB, respectively. If you're querying a dBASE file, these files are saved with the .DBF extension instead.

The sections that follow explain how to use these reserved words.

To define a query as an INSERT or DELETE query, click under the table name in the Query window and choose the reserved word you want from the menu. Or, use the keyboard to move to the leftmost column, then type the first letter of the operation you want (I or D). For more information, see pages 125 and 126.

To remove one of these reserved words from the leftmost field, choose the blank option at the top of the menu.

To define a query as a CHANGETO query, use the reserved word CHANGETO in the field you want to change, between the old and the new values. For more information, see page 127.

In a query image, the table name is located above the leftmost field (level with the field names). If the table name isn't visible, scroll left to display it.

If you're running a query on a network, Database Desktop places a full lock on tables involved in an INSERT, DELETE, or CHANGETO query because they change the values in the tables. You won't be able to process the query until all other users have released their locks on the table(s).

After an INSERT, DELETE, or CHANGETO query is run, a temporary table appears with the results. As with the *Answer* table, the Inserted, Deleted, and Changed tables are overwritten by the next query of the same type.

Note To save a copy of any temporary table, first create a query for that table with CheckPlus marks in all fields. (Use the check menu in the check box under the table name.) Before running the query, choose Properties | Answer Table to specify a new name for the *Answer* table (see page 101) so it won't be overwritten by subsequent queries.

Caution Since these queries change data in tables, it's a good idea to try them on *copies* of the sample tables, to keep the originals intact.

Inserting records

Unlike other kinds of queries, the values you type into query statements in an INSERT query are expressions that create new values; they do *not* select records.

In an INSERT query, if you don't want a field to appear in the target table, you omit a query statement in that field. After the query is run, fields in the target table remain blank if fields in the source table are blank (don't have query statements).

When you add data from a source table to a target table, the entire table structure doesn't have to match, just the fields with query conditions.

If the target table is not keyed, records from the source table are inserted at the end of the target table. If the target table is keyed, the records are inserted in key sort order. If any records in the source table have the same key value as existing records in the target table, the records are not inserted. Instead, they appear in a *Keyviol* table, another type of temporary table. (Multiple key violation tables are named *Keyviol1*, *Keyviol2*, and so on.)

Note Do *not* place checkmarks in any fields in an INSERT query; checkmarks cause an error and the query won't run.

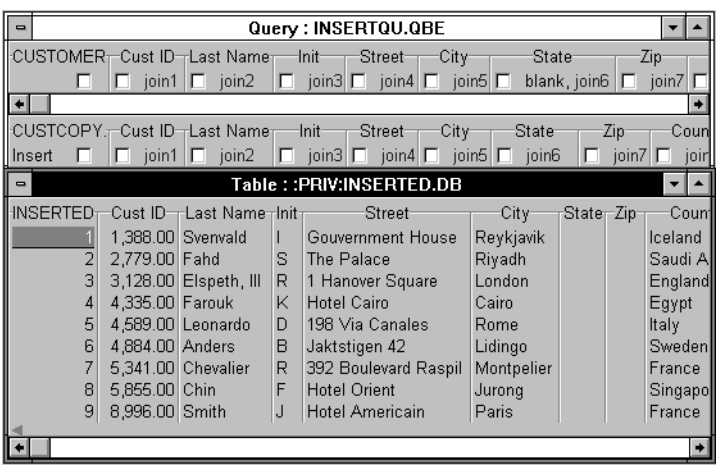
Creating an INSERT query

1. Add the source tables and target table to the Query window.
2. In the source table(s), place a unique example element in each field you want to insert in the target table.
3. For each field in the source table(s), enter any selection conditions for the field values.
4. In the target table, position the pointer under the table name, hold down the mouse button, then choose Insert from the list that appears.
5. In the target table, use example elements that match example elements in the source table(s) to specify which fields you want to insert.
6. To add any constants to the values, add them to the query expressions in the target table.
7. Run the query.

Tip To use an INSERT query to copy records to a separate table, you can use DOS or the File Manager to create a copy of the source table, then delete all the records in the copy, and use it as the target table. To also save a copy of the index, copy the table's .PX file (for Paradox) or its .MDX file (for dBASE).

For example, to copy all of the international customers to a separate table, copy CUSTOMER.DB, delete all records from the new copy in the Table window (in Edit mode), then run a query as shown in the next figure. (This copy of the table won't be indexed unless you also copy its associated index (.PX) file.)

Figure 7.16 Inserting records into a target table



Undoing an INSERT query

After you run a query, the *Inserted* table lists the records that have been added to the target table. To remove the records shown in the *Inserted* table from the target table, create a new query as follows:

1. Choose File | New | QBE Query.
2. Add Inserted as the source table to the Query window.
3. As the target table, add the table containing the inserted records you want to delete.
4. In the target table, position the pointer under the table title, hold down the mouse button, then choose Delete.
5. Use example elements to join the source table and the target table.
6. Run the query.

Caution If any records in the *Inserted* table duplicated any existing records, this procedure will delete the original as well as the duplicate records.

Deleting records

Although in Edit mode you can choose Record | Delete (or press *Ctrl+Del*) to remove the selected record from a table, a DELETE query is faster when there are multiple records that are similar enough to meet a set of selection conditions.

Caution If you don't enter any selection conditions in a DELETE query, *all records* will be deleted from the table.

Creating a DELETE query

1. In the Query window, add the table you want to delete records from.
2. Position the pointer under the table name, hold down the mouse button, then choose Delete.
3. Enter any selection conditions that identify the records to be deleted.
4. Run the query.

Records deleted from the source table appear in *Deleted*, a temporary table.

SQL The *Deleted* table is not created when you run a DELETE query on a SQL table.

Tip To undo deletion from a table, define *Deleted* as the source table in an INSERT query, and define the original table as the target table.

A sample DELETE query

To remove all 1987 and 1988 orders from the *Bookord* table, you can create and run the query shown in the next figure. *Deleted*, the name of the temporary table created by this query, can be saved by renaming it in DOS or by using it as a source table in an INSERT query.

Figure 7.17 Deleting records with a DELETE query

Query : DELETED.QBE

BOOKORD	Cust	Date	Item #	Vol	Quant	Emp #
Delete ☐	☐	☐ ../87 or ../88	☐	☐	☐	☐

Table : :PRIV:DELETED.DB

DELETED	Cust	Date	Item #	Vol	Quant	Emp #
1	1,386.00	8/21/87	1	M13	22.00	146.00
2	1,784.00	5/5/88	1	I16	23.00	517.00
3	1,784.00	8/15/88	1	M27	23.00	775.00
4	1,784.00	12/1/88	1	I24	21.00	517.00
5	1,784.00	12/1/88	2	M18	14.00	517.00
6	2,177.00	12/6/88	1	T24	31.00	517.00
7	2,579.00	2/29/88	1	I16	21.00	537.00
8	2,579.00	7/13/88	1	E11	1.00	537.00
9	2,579.00	7/13/88	2	I24	12.00	537.00

Modifying records

To modify records in a table, you can create and run a CHANGETO query. A single CHANGETO query can define a calculation *and* write the new values resulting from the calculation. After the query is run, a temporary table named *Changed* appears. *Changed* contains a copy of the original records (before the query was run).

SQL The *Changed* table is not created when you run a CHANGETO query on a SQL table.

Creating a CHANGETO query

1. In any field except the leftmost field in the query image, type the old entry you want to replace, followed by a comma.
2. Press *Spacebar*, then type `changeto` followed by a space.
3. Type the new value you want to replace the old one.
4. Run the query.

Note Checkmarks are not used with a CHANGETO query.

Undoing changes made by CHANGETO

To restore the original contents to records changed by a CHANGETO query, you first delete the new records, then reinsert the original records. Because Changed is a temporary table, you must perform these steps *before* running another CHANGETO query:

1. Create a DELETE query (see the previous section) for the table.
2. Define selection conditions to remove the changed records.
3. Run the DELETE query to remove the changed records.
4. Create an INSERT query that defines the *Changed* table as the source table, and the original table as the target table.
5. Run the INSERT query to reinsert the original records into the table.

Changing values with example elements

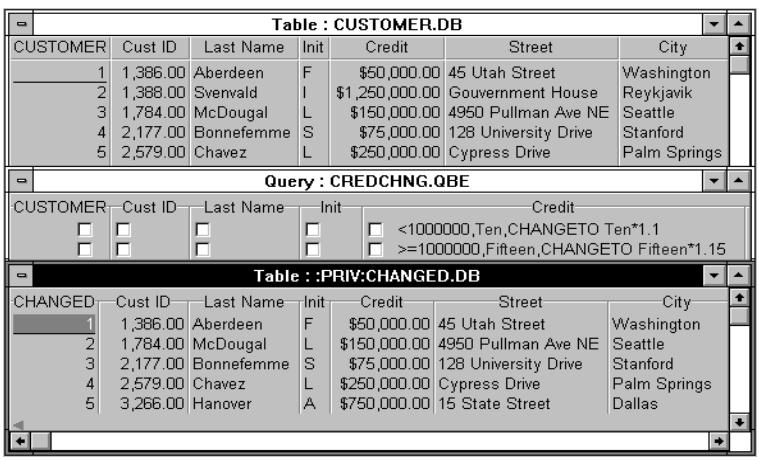
1. In the field with values you want to change, type the selection condition followed by a comma (the AND operator).
2. Place a unique example element name after the comma.
3. Type a comma, and then `changeto` followed by a space, the example element, and the rest of the mathematical expression you want.
4. Run the query.

Changing multiple values with CHANGETO

Separate lines in the query image can be used to define multiple example elements in a single query. For example, you can use example elements with CHANGETO to increase the credit amounts for customers in *Customer* by different amounts.

Notice in the next figure, an example element named **Ten** defines customers with a credit amount of less than $1,000,000; an example element named **Fifteen** defines customers with a credit amount equal to or greater than $1,000,000. The example elements **Ten** and **Fifteen** are then used with CHANGETO and the multiplication operator to write the new credit amounts to the table, as shown in the following figure.

Figure 7.18 CHANGETO with multiple example elements



Notice that the records in the *Changed* table are sorted *in the order they were processed*. For example, the first line of the query changed the records of all customers whose credit rating was under $1,000,000, so these records appear first.

Performing queries on groups of records

You can define queries about groups of records in a table to

- Select records based on characteristics of a group (such as items that appear on two or more orders)

- Calculate statistics on groups of records (such as the average number of orders placed by each city)
- Compare characteristics of a group with other records (such as which customers have placed more orders than any California customer)

Note As with other queries, you check a field to include it in the *Answer* table. However, when a checkmark appears on the same line as a *summary operator* (see the next section), the records are also divided into groups based on the values in the checked field.

Using summary operators

To specify conditions in a query for groups of data, you use summary operators. The following table describes the types of summary operators.

Table 7.11 Types of summary operators

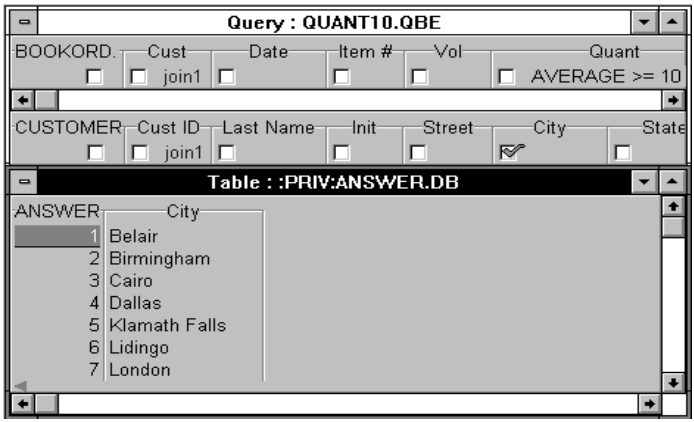
Name	Description
AVERAGE	Averages the values in the group. Valid for operations on Paradox fields (number, short, money, date, long integer, and BCD) and on dBASE fields (number, float, and date).
COUNT	Counts the number of values in the group. Valid for operations on all Paradox or dBASE field types.
MAX	Identifies a maximum value for the group. Valid for operations on Paradox fields (alpha, number, short, money, date, long integer, and BCD) and on dBASE fields (character, number, float, and date). The current language driver, typically ASCII sort order in the U.S., determines the maximum value in alpha and character fields. For example, the value AAC is the maximum for the group AAA, AAB, and AAC because the decimal code number for C in the ASCII sort order is higher (67) than it is for A (65) or B (66).
MIN	Identifies a minimum value for the group. Valid for operations on Paradox fields (alpha, number, short, money, date, long integer, and BCD) and on dBASE fields (character, number, float, and date). The current language driver, typically ASCII sort order in the U.S., determines the minimum value in alpha and character fields. For example, the value ZZA is the minimum for the group ZZC, ZZB, and ZZA because the decimal code number for A in the ASCII sort order is lower (65) than it is for B (66) or C (67).
SUM	Totals the values in the group. Valid for operations on Paradox fields (number, short, money, long integer, and BCD) and on dBASE fields (number and float).
ALL	Includes all values in a group (including duplicates).
UNIQUE	Discards duplicate values when performing a group operation (used in combination with other summary operators).

Tip To override default grouping in a CALC operation, you can add either the reserved word ALL or UNIQUE to the query statement.

Averaging groups of records

To find cities in which the average number of orders is greater than or equal to ten, perform an AVERAGE query as shown in the next figure.

Figure 7.19 A sample AVERAGE query



The *Customer* and *Bookord* tables are joined using example elements in a common field. The Check in the City field of *Customer* groups the records by city, and displays the city names (which match the selection condition) in the *Answer* table. The selection condition `average >=10` averages the number of orders for each city and selects those cities with ten or more orders.

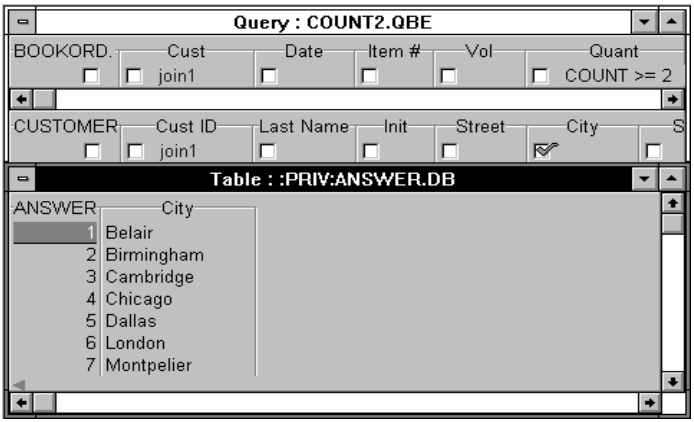
Tip To make each Quant value in the group appear in the *Answer* table, place a Check in the Quant field of this query. (This also causes the records to be grouped by Quant, in addition to City.)

Note In a Set query (discussed on page 135), to calculate the average of all values in a field, type `calc average` in the field. This expression creates a new field in the *Answer* table, called Average of FieldName, where FieldName is the name of the field in which you type the expression.

Counting groups of records

To view cities that have placed two or more orders, you can perform a COUNT query as shown in the next figure.

Figure 7.20 A sample COUNT query

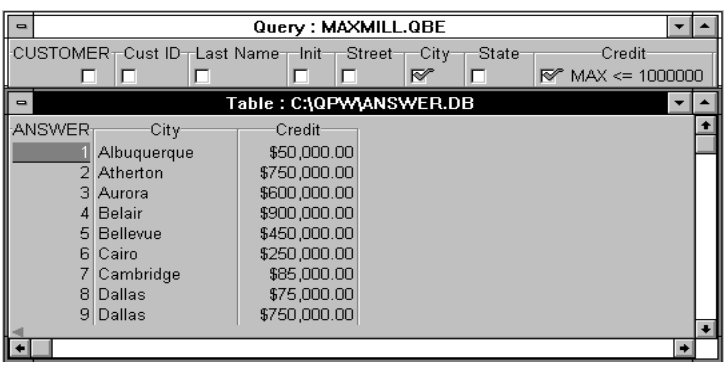


The Check in the City field of *Customer* groups the records by city and makes the cities that match the selection condition appear in the *Answer* table. The selection condition `count >=2` counts all orders in each city, then selects the groups that have a count of two or more.

Grouping records by a maximum or minimum value

A MAX query defines the maximum value for a selection condition, and a MIN query defines the minimum value. The sample MAX query in the next figure locates cities in which the highest credit limit is $1,000,000 or less.

Figure 7.21 A sample MAX query

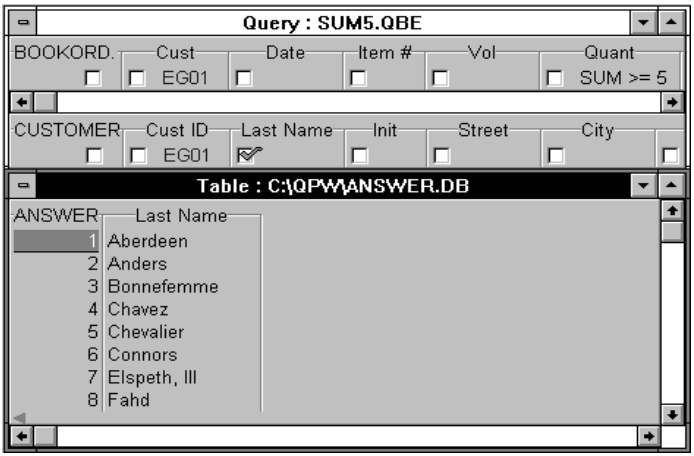


The Check in the City field of *Customer* groups the records by city and displays in the *Answer* table the cities that match the selection condition. The selection condition `max <=1000000` finds the credit limit for each city and selects those with $1,000,000 or less.

Summing groups of records

To find customers who have placed five or more orders, perform a SUM query like the one shown in the next figure.

Figure 7.22 A sample SUM query



The Check in the Last Name field of *Customer* groups the records by customer name and displays in the *Answer* table those names that match the selection condition `sum>=5`. This expression totals the quantities ordered by each customer, then selects those with quantities of five or more.

Tip To use this query to display duplicate customer names, place a Check in the Quant field. The quantities per order as well as the customer name will appear in the *Answer* table.

Calculating group statistics

The CALC operator that calculates new fields for the *Answer* table (discussed on page 121) can also be used to calculate statistics for groups of records. For example, it can be used with

- AVERAGE
- COUNT
- MIN
- MAX
- SUM

All CALC queries create a new field in the *Answer* table. The new field is named SummaryOperator of FieldName (where SummaryOperator is the name of the operation performed on the value, such as SUM; and FieldName is the name of the original field).

Tip To rename the field before you run the query, use the AS operator (see page 112).

Forming groups to calculate

When CALC is used with a summary operator, calculations are performed on groups of records. When you place checkmarks to display fields in the *Answer* table, you are also

forming groups on which to perform the calculation. To perform a calculation on all records in the table, don't check any fields; the entire table is the group.

The following table shows some examples of combining CALC with summary operators.

Table 7.12 Combining summary operators with CALC

Combination	Description of result
CALC COUNT	A checkmark in the field you want to group the records by, and the expression `calc count` in the field with quantities, returns the number of *unique group values* in the table. To group by more than one field, place checkmarks in each field you want to group by. To include duplicates in a COUNT operation, type `all` after the CALC COUNT operator.
CALC MIN	A checkmark in the field you want to group the records by, and the expression `calc min` in a date field, returns the values in the table and the earliest dates associated with the values.
CALC SUM	A checkmark in the field you want to group the records by, and the expression `calc sum` in the field with quantities, returns the sum of the quantities *for each group* in the table. To group by more than one field, place checkmarks in each field you want to group by.

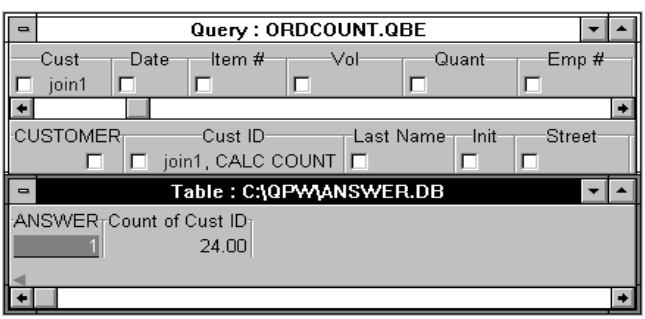
The following sections describe how to create these CALC queries.

Counting unique values

By default, the COUNT operator counts only unique values.

For example, to find out how many customers have placed orders, you can create a query as shown in the next figure.

Figure 7.23 An example query using CALC COUNT



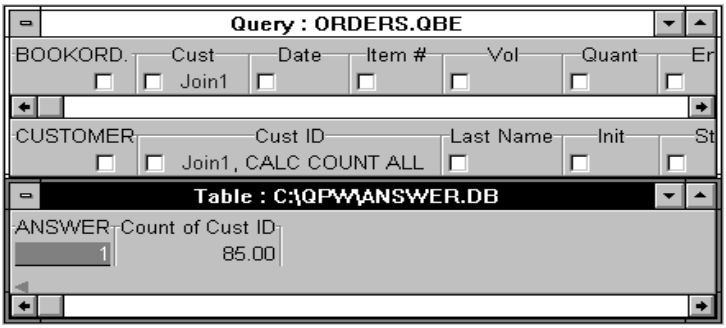
Although *Bookord* has 128 records with order details, the CALC COUNT operation returns a count of 22 because many customers have made multiple purchases (these duplicate values are ignored).

Counting all values

To include duplicate values in a count of records, type `all` after CALC COUNT.

For example, to find the total number of orders placed by customers (regardless of multiple orders), you can create a query as shown in the next figure.

Figure 7.24 An example query using CALC COUNT ALL

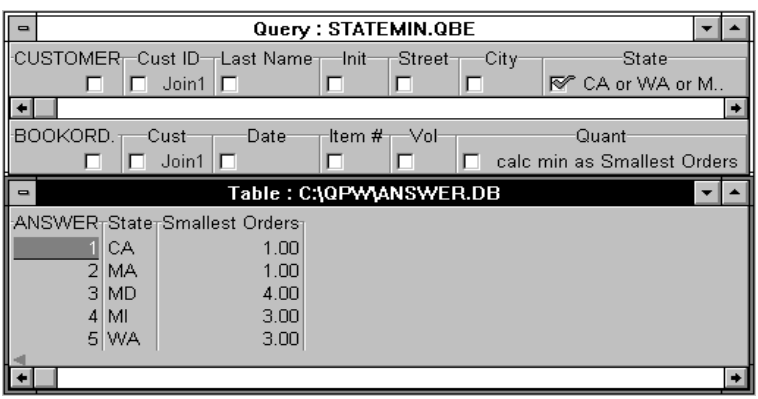


Summarizing group values

To display the minimum or maximum values in the group, you can use CALC MIN or CALC MAX. Because placing a checkmark in a field groups records on that field, only the CALC MIN and CALC MAX expressions are used to display the records in the *Answer* table.

For example, to find the minimum orders placed in California, Washington, Maine, Maryland, and Michigan, you can create a query as shown in the next figure.

Figure 7.25 A sample query using CALC MIN



The Check in the State field creates five groups of records, one for each state value that matches the selection condition `CA or WA or M.. `. The values that match the expression `calc min as Smallest Orders` appear in the *Answer* table in a new field named Smallest Orders. Note that checkmarks are *not* used to display the minimum values.

Grouping multiple fields

When a query performs calculations on a group of records, the number of fields checked in the query image is significant:

- **Single field.** A single, calculated value is returned for each value in the checked field.
- **Multiple fields.** A value is calculated for each *combination of values* in the checked fields. In other words, multiple checkmarks create groups based on multiple fields.

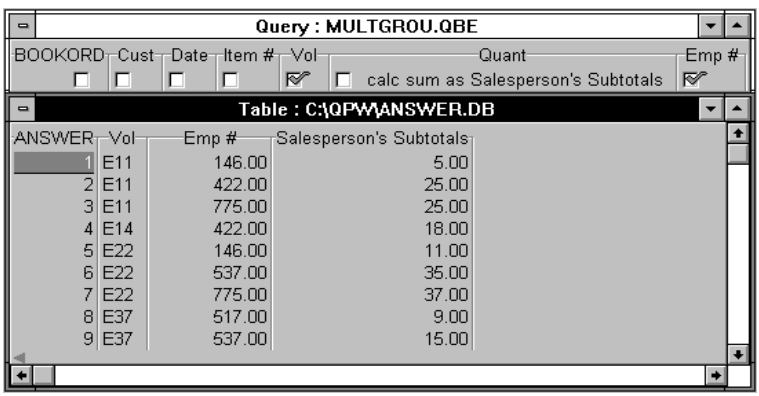
For example, to calculate the sum of all orders for the Vol field in the *Bookord* table, you can check the Vol field, then enter the query statement `calc sum` in the Quant field.

When this query is run, it calculates all the orders for each value in the Vol field. Its *Answer* table displays the values in the Vol field and the number of orders for each item.

The results of this query change dramatically when multiple fields are checked. For example, to find the total number of each volume that has been sold by each salesperson, you can group by both the Vol and Emp # fields, as shown in the following figure.

After this kind of query is run, the *Answer* table displays the relationships between the checked fields.

Figure 7.26 Multiple fields in a group query



ANSWER	Vol	Emp #	Salesperson's Subtotals
1	E11	146.00	5.00
2	E11	422.00	25.00
3	E11	775.00	25.00
4	E14	422.00	18.00
5	E22	146.00	11.00
6	E22	537.00	35.00
7	E22	775.00	37.00
8	E37	517.00	9.00
9	E37	537.00	15.00

The Checks in Vol and Emp # group the orders based on both of these fields. The AS operator clarifies the contents of the calculated field by assigning it a name in the *Answer* table. In the *Answer* table, each volume displays the number of orders taken by a particular salesperson.

Using sets

Set operations are useful for revealing trends and patterns in data with a single query.

A *set* is a type of group—a specific group of records about which you intend to ask further questions. Once you've defined a set in a query, there are two kinds of comparisons you can make:

- **To other groups of records.** The *set comparison operators* (ONLY, NO, EVERY, and EXACTLY) are used to compare other groups of records to the set. For example, you can ask

- Which products have been ordered only by customers from the western United States?
- Which customers haven't ordered any books?

- **Summary comparisons with other groups.** Because a set is a kind of group, you can use summary operators to compute its values, then compare the result to values in other records. For example, you can ask
  - Which customers' credit limits are more than twice the average credit limit?
  - Have customers from any other state ordered more books than customers from Texas?

Creating a Set query

Defining a Set query is very similar to creating selection conditions. Every Set query consists of a

- **Set definition.** One or more lines in a query image can define a set. To define a set, you click below the table name, then choose Set. Also, you create example elements and selection conditions on the same line in the query image. Lines that are part of the set definition *cannot* contain checkmarks or summary operators.
- **Set comparison.** To compare a defined set to other records, you use set comparison operators. The set operators ONLY, NO, EVERY, and EXACTLY determine which records meet specific comparisons to the set. You can also use a summary operator (see Table 7.11 on page 129) instead of a set comparison operator. To form groups of records to compare to the defined set, you use checkmarks.

Optionally, you can display related information about the records by checking other fields or adding joins to additional tables.

Comparing records to a set

To ask questions about other records or groups of records, you use set comparison operators, as described in the following table.

Table 7.13 Set operators

Name	Description
ONLY	Displays groups that only contain values in the set (it doesn't display the members of the set itself).
NO	Displays groups that don't contain any of the values in the set. (To find individual records that don't match the selection conditions of a "set," use the NOT operator instead.)
EVERY	Displays groups that contain every value in the set (and possibly others).
EXACTLY	Displays groups that contain only values in the set and no others.

To use any of these set comparison operators, first define the set. On another line of the query image, type the name of the operators followed by the set name, as shown in the following two examples:

- **ONLY operator.** In Figure 7.27 on page 138, the ONLY operator is used to select groups of records containing the same value as the **Join1** set. This operator selects groups that contain only members of the defined set.
- **EVERY operator.** In Figure 7.30 on page 140, the EVERY operator is used to locate groups of records containing all values in the **JoinVol** set. This operator locates customers who have ordered all titles in the set regardless of their other purchases.

Note The ONLY operator cannot be used to perform calculations.

Defining a set

1. Click the New Query button in the Toolbar.
2. Add the table(s) you want to query.
3. To create a Set query, click the menu under the table name in the Query window, then choose Set.
4. To define the set, place a unique example element in each field you want to select.
5. To refine the set further, enter any selection condition(s) that specify the records to be included in the set.
6. To join fields in multiple tables, use matching example elements.

Comparing a set

1. To compare the defined set to records in another table, create selection conditions in the other table.
2. To list the query results of any field, place a checkmark in the field.
3. Run the query.

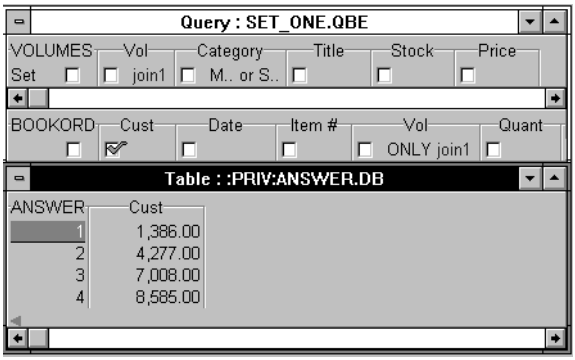
Tip To group the records by a field, without displaying that field in the *Answer* table, use the GroupBy check instead of a checkmark (see page 139).

A sample Set query

To plan a direct mail campaign to promote a book that features diets of sports celebrities, you could use a Set query to locate specific customers.

For example, the most likely customers are those who only order books in the Meals or Sports categories. To create a query to do this, you first need to define this set, then compare it to the orders placed by all customers, as shown in the next figure.

Figure 7.27 A sample Set query



The pair of **Join1** example elements links the lines in the *Volumes* and *Bookord* tables. In the *Volumes* table, the Set reserved word in the leftmost field and the selection condition in the Category field defines a set of books that contains either Meals or Sports titles.

In *Bookord*, the Check in the Cust field groups the orders by customer. The ONLY operator in the Vol field matches those records of customers who have ordered only books included in the set. Customers who have ordered books from any other series aren't listed in the *Answer* table.

Defining a two-line set

A set definition can occupy more than one line. However, the way you define the set is exactly the same. For example, to find which books cost more than the average price of books in the Meals series, you can create a query as shown in the next figure.

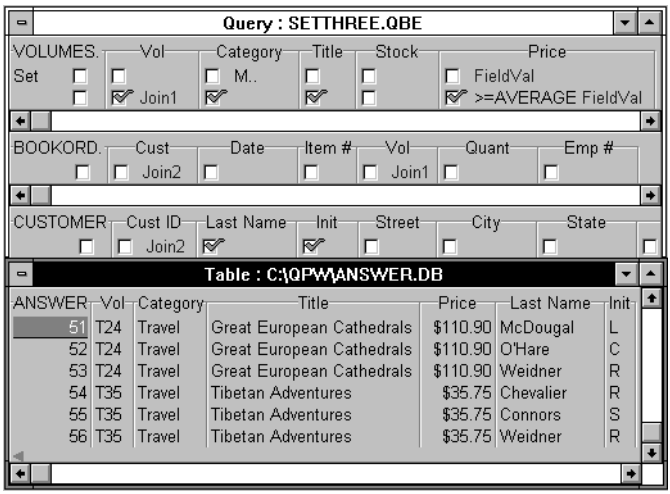
Figure 7.28 A two-line Set query definition

The first line in the query image defines the set of Meals books and their prices. Notice that the Set reserved word appears in the leftmost field of the query image.

The second line of the query image for the *Volumes* table selects records of books with prices greater than the average of the set.

To list the names of the customers who placed the orders shown in the previous figure, you can use example elements. As shown in the next figure, you can add the *Customer* table to the query, then join the *Volumes*, *Bookord*, and *Customer* tables.

Figure 7.29 Joining another table to a set query

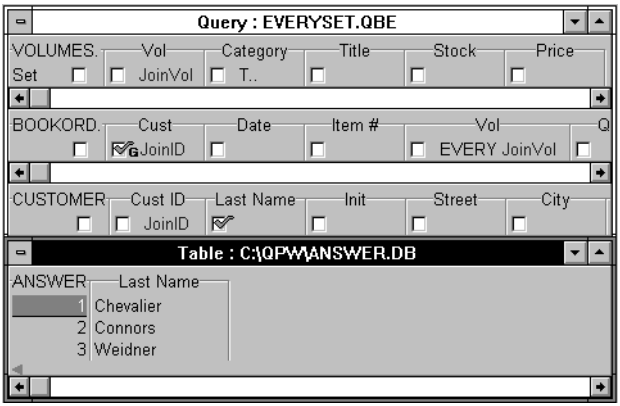


Using the GroupBy check

To group records by the values in a specified field *without* displaying those values in the *Answer* table, use the GroupBy check. The GroupBy check is valid *only* with Set queries.

For example, to locate records for all customers who have ordered all titles in the Travel category and display only the customer names, you can create a query as shown in the next figure.

Figure 7.30 A Set query using the GroupBy check



After the set of titles is defined with the *Volumes* query image, the EVERY operator is used to compare orders with the set. This operator selects groups that include orders for every member of the set. Customers who have also ordered books from other categories might also appear in the *Answer* table.

The GroupBy check groups the records by customer, but does not show the Cust ID field in the *Answer* table.

Using inclusive links

So far you've seen several queries that use example elements to join tables together. These queries, called *exclusive links*, are useful for retrieving all the records in one table that match records in another table. However, exclusive links also exclude any records that don't match records in another table.

To retrieve all records in a table, regardless of whether they match records in another linked table, you can use the inclusion operator ! (an exclamation point) to create an *inclusive link*.

When a query with an inclusive link is run, the complete set of records is retrieved from the table that has the ! operator in its query image. The corresponding records that match the selection conditions are then retrieved from the other table(s). If there isn't a matching record in the other table, the corresponding fields in the *Answer* table will be blank.

In other words, the ! operator overrides Database Desktop's default for linked tables. This process is demonstrated with an example in the next section.

In the following sections you'll see how to

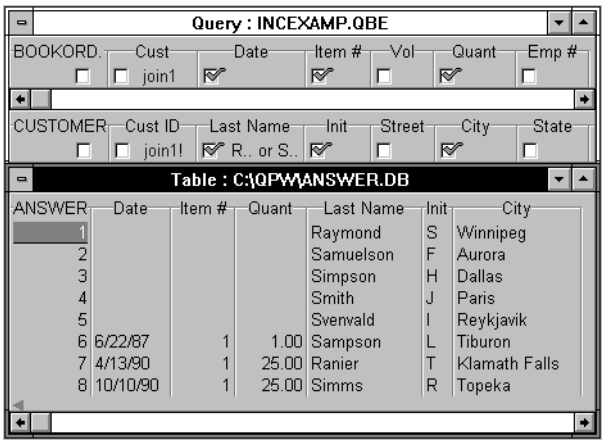
- Use multiple !s to retrieve all the records from multiple tables
- Use ! in an arithmetic expression
- Use both inclusive and exclusive links in the same query

Note As in all other queries, to see duplicate records in the *Answer* table you must use the CheckPlus mark in the query image.

An example of an inclusive link

To find out if there are customers in the *Customer* table who have never placed an order, you can use an inclusive link as shown in the next figure. Notice that the *Answer* table contains customer records for customers whose names begin with R or S, including those customers who haven't placed any orders.

Figure 7.31 Using the inclusion operator !



The inclusion operator in the *Customer* table identifies it as the *master table*. All customer records in the master table that meet the selection condition `R.. or S..` appear in the *Answer* table. The *Bookord* table is the *lookup table*; it supplies any corresponding values for records in *Customer*.

Because only three customers have placed orders, the order records for Raymond, Samuelson, Simpson, Smith, and Svenvald are blank.

Tip To see the difference in the results, try creating and running this query, then delete the ! operator and run the query again. The second *Answer* table will display only those customer records that match a record in *Bookord* (there won't be any blank order values).

Processing order for links

The order in which inclusive links are processed is significant. Selection conditions in the *master table* (the table with the ! operator) are always processed first. Because the records in a master table are always included in the *Answer* table, selection conditions in the linked table(s) might not produce the results you want.

For example, the only difference in the next two figures is the placement of the ! operator, but it significantly changes the result.

In each query, the selection condition in the master table retrieves the appropriate records. Then, the records in the lookup table that match the selected master records*and* meet the selection condition in the lookup table are retrieved. Notice that in each case, multiple records appear for customers who placed multiple orders meeting the selection condition.

Figure 7.32 The ! operator in *Customer*

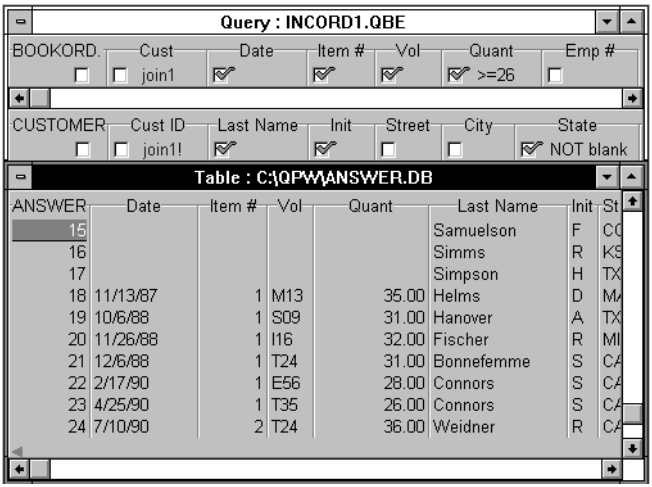
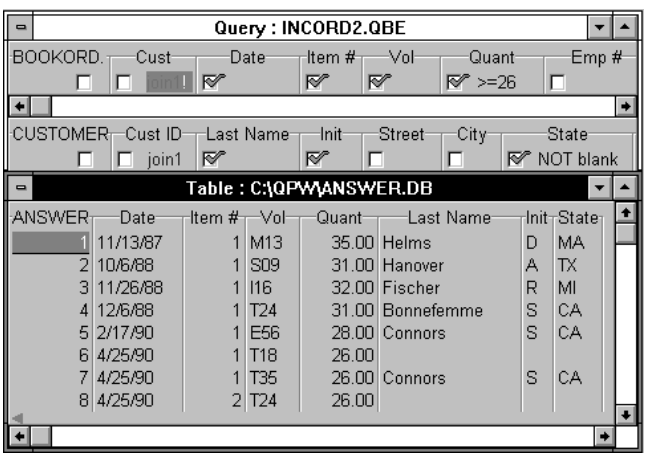


Figure 7.33 The ! operator in *Bookord*



Using multiple ! operators

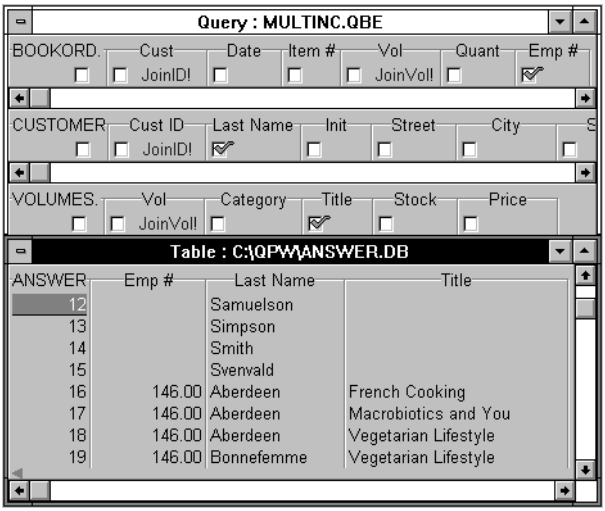
A query that uses the ! operator is sometimes called an *outer join*. A query that uses a single ! operator is also called an *asymmetrical outer join* because an inclusive link has been specified for only one of the tables involved in the query.

A *symmetrical outer join* is a query using multiple ! operators; linking tables this way can reveal information that might get lost in other queries.

To clearly see the relationships between two tables, you can create a symmetrical outer join that is all-inclusive; it retrieves *all* values in the common fields of the tables (without requiring that the values match each other).

For example, the next figure shows a query that creates an *Answer* table that is more useful than viewing the *Bookord* and *Volumes* tables as separate tables. The *Answer* table makes it clear which products haven't been ordered and which customers haven't placed any orders.

Figure 7.34 Multiple ! operators in a query

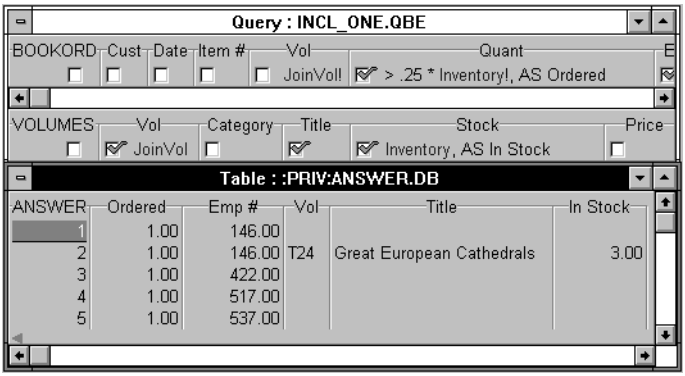


The multiple ! operators in this query make all book titles appear in the *Answer* table—even if an order hasn't been placed by a sales representative.

Using ! in arithmetic expressions

Another use for the inclusion operator is in expressions that perform calculations. For example, to keep track of current inventory levels, the shipping department can use the ! operator as shown in the next figure. Orders that exceed one-quarter of inventory appear in the *Answer* table within the context of all orders.

Figure 7.35 A query using the ! operator in an expression



The ! operators in the *Bookord* query image ensure that the *Answer* table contains all values in *both* the Vol and Quant fields (volumes as well as all orders).

The two tables are linked by two pairs of example elements, **JoinVol** and **Inventory**. Because there is no ! operator in the *Volumes* table, the volume ID, title, and number of items in stock are retrieved only for records that exceed the selection condition `>.25*Inventory`.

Avoiding ! operator conflicts

In the previous example, it's critical that the ! operator in the *Bookord* query image follow *both* example elements. A single ! operator in the first line makes the query ambiguous and will cause an error message to appear when the query is run.

In this example, when the ! operator follows the JoinVol example element, it assures that all volume ID's appear in the *Answer* table. When the ! operator does *not* follow the Inventory example element, it is a conventional link and it retrieves only those products whose ordered quantity exceeds one-quarter of the existing inventory.

The following section describes the rules for using inclusive and exclusive links.

Rules for linking tables

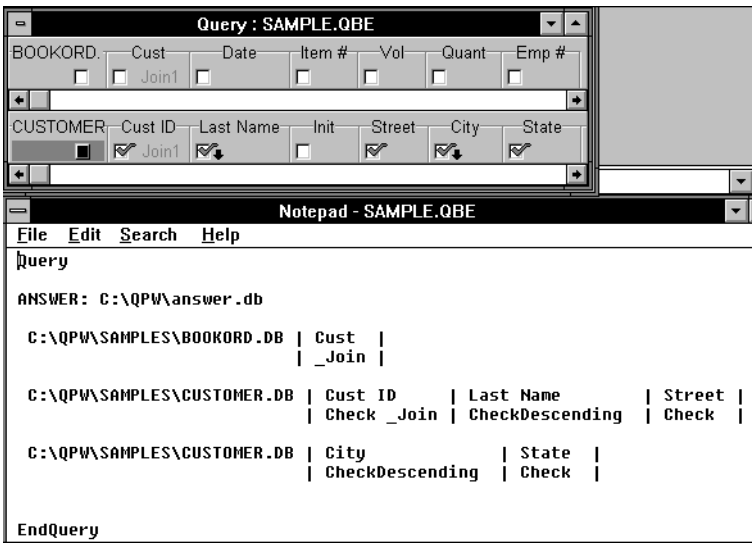
In some ways both asymmetrical and symmetrical outer joins are different from other types of queries. The main differences concern the order in which the elements of a query are processed, and the ways you can link the different lines of a query.

- Any two lines in a query statement can use *either* an inclusive link (!) or an exclusive link to associate them (but not both).
- Both inclusive and exclusive links can be used in the same query statement, if they don't both involve the *same pair* of lines. When both link types exist in one query, they are processed in order from least to most inclusive.
- You can use ! with any given example element only once per line and twice per query.

QBE file syntax

Text in a .QBE (query) file defines the query, which is always enclosed by the reserved words Query and EndQuery. The following figure shows a sample query, and how it looks when the contents of its file is viewed in Notepad, for example.

Figure 7.36 A query and its .QBE file



The best way to learn the file syntax is to save your queries, then open them in a text editor and experiment by revising them. For example, to change the sort order of a checked field from ascending to descending, replace the word Check with CheckDescending. To see the results, run the modified query, then study the difference.

Once you learn the .QBE file syntax, you can create queries directly in a text editor. The following sections describe the syntax of .QBE files.

Tip Notice how easy it is to read the query text in the above figure when the columns line up. To duplicate this method, set the font property for the block to Courier, a nonproportional font.

Defining the *Answer* table name

Below the reserved word Query in a .QBE file, the line that begins with `ANSWER:` defines the location and file name for the *Answer* table created by the query. This line is optional because the *Answer* table is created by default in your program directory.

Tip Temporary tables, such as the *Answer* table, are overwritten each time a query is run. To keep a copy of the *Answer* table, replace the `answer.db` text with another valid file name: open the .QBE file in Notepad, edit the *Answer* table name, save the file, then run the query again.

Caution If you use this technique, make sure to use a unique file name to keep a copy of the *Answer* table. If there is already an existing file with the name you specify, Database Desktop will overwrite it.

Overriding the sort order

Below the *Answer* table definition in Figure 7.37, the line that begins with SORT: defines which fields are sorted first, instead of the default sort order. This definition is optional, and only appears in a file if you have set *Answer* table sort order options in the Sort Answer dialog box (see page 99).

To type the names of fields you want to be first in the sort order, follow the sample format in Figure 7.37.

Defining tables and fields

Below the sort order definition (if there is one), and slightly indented from the left margin, is the query image information. Indentation of this information is optional, as is the use of spaces before or after delimiters.

On the first line, the location and name of each table in the query is followed by each field name that contains checkmarks or conditions. The pipe character (|) delimits, or separates, table names and field names. In a .QBE file, one space appears before and after the pipe character, but these space characters are optional. Similarly, it's optional for the delimiters to be aligned.

The line directly below the table and column name in the .QBE text file contains the information you placed in the query image. When there are multiple lines in a query image (page 110), the number of lines and information in the lines of the .QBE file match the query image.

Checked fields in the query image are indicated by the keyword Check, CheckPlus, CheckDescending, or GroupBy. Example elements (see page 113) are preceded by an underscore character (_).

Note Blank rows are required between each line or group of lines that contain a reserved word, a query image definition, or a wrapped piece of a query image definition.

Wrapping long lines

Lines of text in a .QBE file are never broken in the middle of a column name, selection condition, or sort field definition. When a line of text exceeds 80 characters, it appears on another line as follows:

- **SORT section** text (optional) wraps to the next line. Field definitions are broken only after the comma that separates them.
- **Query section** text skips one line, then wraps to the following line. The table location and name are repeated for clarity, before the column names are defined.

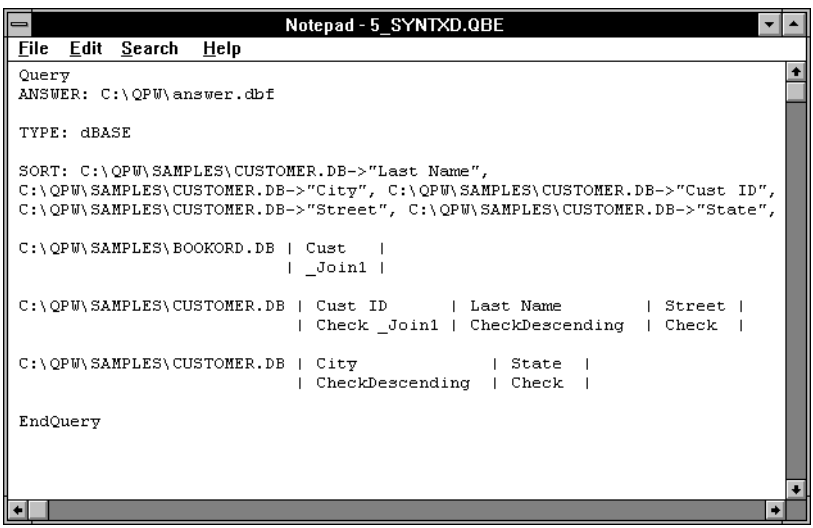
Creating a dBASE *Answer* table

By default, when a query is run, the *Answer* table is written in the Paradox file format. To create an *Answer* table in dBASE format,

- **Change the extension** of the *Answer* table from ANSWER.DB to ANSWER.DBF.
- **Add the TYPE clause** to a new row below the *Answer* table definition. To do this, type `TYPE:dBASE` into the query file.
- **Choose dBASE** for Answer Table Type in the Answer Table Properties dialog box.

The next figure shows how to change SAMPLE.QBE (shown in Figure 7.36) so it produces an *Answer* table in dBASE format.

Figure 7.37 A query for an *Answer* table in dBASE format



Translating field types

When you query a Paradox table to create an *Answer* table in dBASE format, Database Desktop translates Paradox field types to dBASE field types. Table 7.14 shows what to expect when you query a Paradox table to a dBASE *Answer* table.

Table 7.14 Querying a Paradox table to a dBASE *Answer* table

From Paradox table	To dBASE *Answer* table	Side effects
Alpha	Character	
Number	Number	Assigns size (20) and dec. (4)
Money	Number	Assigns size (20) and dec. (4)
Short	Number	Assigns size (6) and dec. (0)
Long Integer	Number	Assigns size (11) and dec. (0)
BCD	Number	Assigns size (20) and dec. (4)
Date	Date	

Table 7.14 Querying a Paradox table to a dBASE *Answer* table (continued)

From Paradox table	To dBASE *Answer* table	Side effects
Time	Character	Assigns size (8)
Timestamp	Character	Assigns size (30)
Memo	Memo	
Formatted memo	Memo	Formatting is lost
Graphic	Binary	
OLE	OLE	
Logical	Logical	
Autoincrement	Number	Assigns size (11) and dec. (0)
Binary	Memo	Data cannot be displayed
Bytes	None	Cannot be checked when querying Paradox to dBASE *Answer*

When you query a dBASE table to create an *Answer* table in Paradox format, Database Desktop translates dBASE field types to Paradox field types. Table 7.15 shows what to expect when you query a dBASE table to a Paradox *Answer* table.

Table 7.15 Querying a dBASE table to a Paradox *Answer* table

From dBASE table	To Paradox *Answer* table	Side effects
Character	Alpha	
Float	Number	Removes size
Number	Number	Removes size
Date	Date	
Logical	Logical	
Memo	Memo	Adds size (1)[1]
OLE	OLE	
Binary	Graphic	

1. If the data in the dBASE memo is in text form.

Appendix

A

Using the keyboard

This appendix offers keyboard techniques and tables that summarize keyboard operations.

Keyboard techniques

The following techniques can be used as alternatives to the mouse operations described earlier in the manual:

- To select multiple fields across rows and columns in the Table window, hold down the *Shift* key while you press arrow keys. Fields selected this way must be contiguous.
- To change the order of columns, select a column, then press *Ctrl+R*. The selected column moves to the last position in the table, and all columns to the right of the selected column shift left by one position.

Data entry shortcuts

You can use the following keys when a Table window is in Edit mode. Asterisks indicate these keys also work when viewing records in the Table window.

Table A.1 Data entry shortcut keys

Key(s)	Description of action
←	*Selects the field to the left of the selected field.
→	*Selects the field to the right of the selected field.
↓	*Selects the same field in the record below the current one.
↑	*Selects the same field in the record above the current one.
Home	*Moves to the leftmost field of the table, remaining on the selected record.
Ctrl←	*Same as *Home*.
Ctrl+Home	*Moves to the leftmost field of the first record of the table.

Table A.1 Data entry shortcut keys (continued)

Key(s)	Description of action
End	*Moves to the rightmost field of the table, remaining on the selected record.
Ctrl→	*Same as *End*.
Ctrl+End	*Moves to the rightmost field of the last record of the table.
Enter	Commits value and moves to next field.
Tab	Commits value and moves to next field.
Shift+Tab	Commits value and moves to previous field.
Ins	Inserts record, in Edit mode.
Shift+Del	Cuts, in Edit mode.
Ctrl+Del	Deletes current record, in Edit mode.
Ctrl+Backspace	Deletes selected characters. If no characters are selected, deletes the whole word to the left of the cursor.
Esc	Undoes an edit to an entry (if you press *Esc* before you move off the field).
Alt+Backspace	Undoes changes to a record (before you move off the record).
Ctrl+D	In a blank field, duplicates the information from the record above the current field and inserts it into the current field.
Spacebar	Enters the current date in a date field. Database Desktop enters the components of a date field separately, so if your date format has more than one component, press *Spacebar* more than once.
Ctrl+Spacebar	Displays lookup help, if available, for the table you're editing.
Ctrl+Shift+Spacebar	Opens the Move Help dialog box, where you can move a dependent record from one master to a different master.

Function keys

The following table lists all actions that can be performed with the function keys, alone and in combination with other keys.

Table A.2 Function key actions

Key(s)	Action in table	Action in query
F1	Help	Help
F2	Field view	Field view
Ctrl+F2	Persistent field view	Persistent field view
F3	Super back tab	Up Image (if more than one query image exists in an active Query window)
F4	Super tab	Down Image (if more than one query image exists in an active Query window)
F5	Lock record	Create an example element
Shift+F5	Commit record and unlock	
Ctrl+F5	Post/Keep Locked	
F6	Lookup help	Place a Check or clear any type of checkmark (toggle on/off)
Shift+F6		Select option in check menu (repeatedly pressing this key combination cycles through the different options)
F8		Run the current query
F9	Enter/exit Edit mode	

Table A.2 Function key actions (continued)

Key(s)	Action in table	Action in query
F10	Menu	Menu
F11	Previous record	Move up one line in the current query image; if in last line, and last line is empty, remove line
Shift+F11	Previous set	
Ctrl+F11	First record	
F12	Next record	Move down one line in the current query image; if in last line, add new line
Shift+F12	Next set	
Ctrl+F12	Last record	

Super tab

Super tabs let you jump from one multi-region area to another in query images.

Table A.3 Using Super tab

Key	Action	Description
F4	Super tab	In a query with multiple tables, moves downward to the next table in the Query window. When the active query image is at the bottom of the window, the next one selected is the query image at the top of the window.
F3	Super back tab	Moves in a reverse direction to Super tab, otherwise works identically.

Keyboard shortcuts for table operations

The next two sections list operations you can perform in a Table window.

Ctrl+letter shortcuts

The next table shows the operations you can perform in a Table window with *Ctrl+letter* and *Ctrl+Shift+letter* combinations.

Table A.4 *Ctrl+letter* shortcut combinations

Keys	Action in Table window
Ctrl+C	Copy
Ctrl+D	Ditto (repeat last entry)
Ctrl+F	Field view
Ctrl+L	Lock record
Ctrl+Shift+L	Commit record
Ctrl+R	Rotate columns
Ctrl+V	Paste in Edit mode
Ctrl+X	Delete selected text in Edit mode
Ctrl+Spacebar	Lookup value
Ctrl+Shift+Spacebar	Multi-field lookup

Keypad shortcut keys

To use the shortcuts in this table, be sure *Num Lock* is *off* before you press the navigation keys on either the numeric keypad or the navigation keypad.

Table A.5 Keypad combinations

Key(s)	Non-field view	Field view
PgUp	Up one set of records	Up one set of records
Ctrl+PgUp	Left one screen	Left one screen
PgDn	Down one set of records	Down one set of records
Ctrl+PgDn	Right one screen	Right one screen
Home	First field of current record	Beginning of field
Shift+Home	Select to first field of record	Select to beginning of field
Ctrl+Home	First field of first record	First field of first record
Alt+Home	First field of current record	First field of current record
End	Last field of current record	End of field
Shift+End	Select to last field of record	Select to end of field
Ctrl+End	Last field of last record	Last field of last record
Alt+End	Last field of current record	Last field of current record
←	Left one field	Left one character
Shift←	Select left one field	Select left one character within field
Ctrl←	No action	Left one word
Alt←	Left one field	Left one field
Shift+Ctrl←	Select to leftmost field	Select left one word
→	Right one field	Right one character
Shift→	Select right one field	Select right one character within field
Ctrl→	No action	Right one word
Alt→	Right one field	Right one field
Shift+Ctrl→	Select to rightmost field	Select right one word
↑	Up one line within field	Up one line within field
Shift↑	Select up one line within field	Select up one line within field
Ctrl↑	Same field in first record	Same field in first record
↓	Down one line within field	Down one line within field
Shift↓	Select down one line within field	Select down one line within field
Ctrl↓	Same field in last record	Same field in last record
Ins	Insert record in Edit mode	Insert record in Edit mode
Shift+Del	Cut in Edit mode	Cut in Edit mode
Ctrl+Del	Delete current record in Edit mode	Delete current record in Edit mode
Backspace	Delete selected text in Edit mode	Delete character to left in Edit mode

Index

Symbols

! (inclusion) operator 140–144
  multiple 142
"Fast" queries
  Queries 97
* (asterisk), key field indicator 26
+ operator
  alphanumeric values 122
  date values 108
, (AND) operator 111
.. (wildcard) operator 109
= operator 106
@ (wildcard) operator 109
\ (backslash), query statements 103
| (pipe) in .QBE file 146

A

access rights 39
  assigning 41
  combining 42
  field 41–42
  table 41
activating Database Desktop 9
Add command, Utilities menu 79
Add dialog box 79
adding
  fields 20, 50
  key fields 27, 49, 51
  records 79–82
aliases
  defined 14
Aliases command, File menu 14
All Corresponding Fields lookup 32
ALL operator (group summary) 133
All option (field rights) 42
All option (table rights) 41
alpha fields 23, 50
  changing to date fields 52
  converting data to 52
alphanumeric values, combining 122
alternative sort orders 33
AND operator (,) 111
ANSI character set 33
Answer Sort command
  Properties menu 99, 100
Answer Table command
  Properties menu 99, 101
Answer tables 99–101
  changing data formats 99, 101
  checkmarks 116
  closing for different queries 96
  dBASE 147
  defined 7
  duplicate records 105
  example elements 114
  group statistics, calculating 132–135
  properties 100
  .QBE file, definition in 145
  renaming 99, 101
    fields 112, 121
    table in query block 145
  saving under another name 145
  single records 105
  sorting 100
    defaults 99
ANSWER section, .QBE files 145
ANSWER.TV files 100
ANSWER.TVF files 100
application window 9–11
  Toolbars 10
arithmetic operations
  ! (inclusion) operator and 143
  See also calculations
  date values and 108
arithmetic operators 112
Arrange Icons command, Window menu 10
arranging tables 95
AS operator 112
ascending sort order (default) 95, 99
ASCII files (.QBE) 98
ASCII sort order 129
Assist button (pictures) 29
asterisk (*), key field indicator 26
asymmetrical outer joins 142
at-sign (@), wildcard operator 109
auxiliary passwords 40, 41
  creating 42
Auxiliary Passwords dialog box 40
  Passwords list 42
Auxiliary Passwords option 40
Auxiliary tables
  queries 97
AVERAGE operator (group summary) 129
averaging groups 129

B

backslash characters, query statements 103
beeps, warning 85
binary fields 24
binary large objects 25
blank fields 19
BLANK operator 108
blanks
  (spaces) in values 103
  values, finding 107, 108
BOOKORD.DB (sample table) 4
Borrow button 47
Borrow Table Structure dialog box 47
borrowing a table's structure 46
building tables 18, 19, 22, 46
bytes fields 24

C

CALC operator
  calculations and example elements 121
  group statistics, calculating 132–135
calculations
  See also arithmetic operations
  alphanumeric values and 122
  CALC and example elements, combining 121
  date values and 108
  fields, with example elements and 117
  groups, performing on 132
  queries 121
calendar, computer 107, 108
Cancel Changes command, Record menu 76
Cascade command, Window menu 10
Cascade option (Referential Integrity) 38
Cascade Tables command

View menu 95
cascading changes 38
cascading tables 95
case sensitivity
indexes 35, 37
LIKE operator (case insensitive) 107
passwords 40
queries (case sensitive) 105
.CFG files 11
Changed table (temporary) 127
changes
cascading 38
CHANGETO query (reserved word) 123–124, 127
changing
field names 20, 21, 50
field types 49, 51–52
passwords 41, 42
picture strings 30
referential integrity 39
tables 47
validity checks 50
character sets 43
character strings 28
characters
ANSI 33
backslash 103
blanks (spaces) in values 103
delimiters (|) in .QBE files 146
example elements
leftmost 115
names 114
literal 102
matching patterns of 108
maximum
.QBE file lines 146
query image fields 101
OEM, translating in Database Desktop 33
parentheses (for precedence) 112
picture strings 29
pipe (|), .QBE file and 146
quotation marks 103
space, in query text 146
underscore
example elements 114
.QBE file 146
Check 95
reserved word 146
check menu 95
CheckDescending 96, 105
reserved word 146
checkmarks
CALC fields 121
defined 95
error messages 123
example elements and fields 116
group queries and calculations 132–135
GroupBy check, set queries and 137, 139
multiple in a query 104, 135
placing 96
placing in all fields 96
precedence of 104
.QBE file and 146
removing from a field 96
set query definitions and 136
table operations and reserved words 123
types (table of) 95
CheckPlus 95, 105
reserved word 146
child tables 36
circular references 39
client (DDE) 8
Clipboard
bypassing during editing 76
pasting records to 76
Close All command, Window menu 11
column headings
resizing 72
columns
locking 71
rearranging 72
keys for 149
resizing 72
scrolling 70
selecting entire 76
combining
alphanumeric values 122
comma
(AND) operator 111
query statements 103
common fields, example elements and 118
comparison operators 106
compatibility
field types 52
referential integrity 31, 38
composite indexes
creating 34, 35
composite keys 5, 27, 38
concatenating
alphanumeric values 122
constant values 121
constants 29
conversions, field types and 63, 147
Copy command
Edit menu 76
Utilities menu 60
Copy dialog box 61
copying
data 76
objects 60–64
tables 60–64
COUNT operator (group summary) 130
counting groups 130
Courier font (nonproportional) 145
Create Table dialog box
Field Roster 20, 22
moving through 20
SQL tables 44
creating
indexes 34, 35, 39
lookup tables 31–33
new fields 20
passwords 40, 42
picture strings 29
SQL tables 44
tables 17–56
current field 19
CUSTOMER.DB (sample table) 4
customizing
table view 71–73
undoing 73
Cut command, Edit menu 76
cutting data 76
See also deleting

D

data
converting 49, 50–52
duplicating 18, 26
editing 73–77
hiding 42
losing 50, 51
organizing 17–18
trimming 49
data entry 22, 73–77
automatic 28, 29, 31
facilitating 31
required 28
shortcut keys for 149
templates 28
validity checks 28
values, looking up 31–33, 78–79
Data Entry option (table rights) 41
data integrity 26

databases 17
date fields 23
converting data to 52
dates
arithmetic operations on 108
computer calendar and 107
example elements and 118
.DB files (Paradox tables) 11
dBASE
Answer table 147
field types 25
indexes 11, 27
tables 21
dBASE tables
copying to Paradox tables 147
viewing 26
.DBF files (dBASE tables) 11
.DBT files (dBASE tables) 11
DDE
client 8
defined 8
links 8
pasting 76
server 8
decimal code, ASCII 129
decimal places 23
decimal point, in query statements 103
default field values 28
defaults
exclusive links (defined) 140
sort order 95, 99
Define Index dialog box 45
Define Secondary Index dialog box 34
Clear All button 35
Fields list 35
Indexed Fields list 35
Maintained option 34
Delete command
Edit menu 76, 77
Properties menu 73
Record menu 77
Delete command, Utilities menu 66
Delete dialog box 66
Delete option (table rights) 41
DELETE query (reserved word) 123–124, 125, 126
Deleted tables (temporary) 126
deleting
data 19
fields 50
indexes 36
key fields 27
objects 66–67
passwords 42
queries 125
records 77, 123–124, 126
tables 66–67
values 76
Dependent Tables option 56
descending sort order 96, 99
Desktop *See* application window
dialog boxes
moving through 20
directories
aliases 14
private
temporary tables 98
private (:PRIV:) 13
working (:WORK:) 13
Display Table option 44
displaying record numbers 70
double-clicking, column selection and 76
drivers 56
sort order 129
duplicate records 99, 105, 126
duplicating data 18, 26
Dynamic Data Exchange *See* DDE

E

Edit Data
button 73
command, View menu 73
Edit menu
Copy 76
Cut 76
Delete 76, 77
Paste 76
Paste Link 76
Select All 74
Undo 76
Edit mode 73–77
typing to replace entries 73
editing 73–77
cutting, copying, and pasting 76
deleting vs. cutting 76
example elements 115
field view and 74, 104
function keys for (table of) 150
persistent field view and 75
problems 85
query statements 103
Table window
keys for (table of) 151
typing to replace entries 73
undoing 76
Empty command, Utilities menu 84
Empty dialog box 84
emptying tables 84
EndQuery, reserved word 145
Enter Password(s) dialog box 12, 42
entries *See* values
equal sign (operator) 106
error message
checkmarks and table operations 123
Errorins, Errordel, Errorchg tables (temporary) 123
EVERY (set comparison operator)
example in query 140
example elements 113
calculations
CALC operator and 121
defining first 117
common fields and (for joining tables) 118
creating 114
date calculations and 118
defining value of 116
INSERT queries and 125
joining tables with 115, 118
LIKE operator and 117
names of, valid characters for 114
NOT operator and 117
operators, valid with 117
.QBE file and 146
values, changing with CHANGETO 128
exclamation point, inclusion operator 140–144
multiple 142
exclusive links (defined) 140

F

F9 (Edit mode) 73
facilitating data entry 31
.FAM files 11
Fast Queries option
queries 97
Field Name text box 32
field names 20, 21
changing 21, 50
field rights 41–42
Field Roster (Create Table)
changing 22
filling in 20
field types 22, 77–78
changing 49, 51–52

compatible 52
described 22–26
logical, operators and 107
specifying 22
SQL 26
viewing 20
field values 36
default 28
hiding 42
maximum 28
minimum 28
validity checks 28
field view 74
entering 75, 104
persistent 75
shortcut keys for (table of) 152
Field View command, View menu 75
fields 19
adding 20, 50
ANSWER table, created by CALC 121
calculations in, and example elements 117
checkmarks *See* checkmarks
common (for joining tables) 118
constraints, size 23
creating new 20
DDE link to 8
defined 3
deleting 50
editing data 75
problems 85
including in a query 95
moving 22
names, in .QBE files 146
naming 20
problems editing 85
protecting 41
renaming 50
renaming in queries with AS 112
repeating 17
required 28
resizing 49, 50
selecting 74
types of
logical, operators and 107
translating with .QBE files 147
validity checks for 78
variable-length 24
view *See* field view
Fields list (Define Secondary Index) 35
file extensions 11
File menu
Aliases 14
New 13
Open 11
Table 69
Private Directory 13
Save 13
Save As 13
Working Directory 13
files
ANSWER.TV 100
ANSWER.TVF 100
ASCII 98
.CFG 11
.DB (Paradox tables) 11
.DBF (dBASE tables) 11
.DBT (dBASE tables) 11
extensions (table of) 11
.INI 11
.MB 11
.MDX 11
.NDX 11
opening 11
.PX (primary index) 11
.QBE 13
defined 98
query text 11
sort order section 146
syntax 145–148
saving 13, 98
.SQL 11
.TV (Paradox table view properties) 11, 72, 73
.TVF (dBASE table view properties) 11, 72, 73
types (table of) 11
.VAL 11
.X*nn* (secondary index) 11
.Y*nn* (secondary index) 11
Fill No Help lookup option 32
finding records 123–124
First command, Record menu 70
font, nonproportional 145
formats
changing Answer table 99
formatted memo fields 24
constraints, size 25
formatting
data 22
text 24
function key operations (table of) 150

G

graphic fields 24
graphics 24
GroupBy check 96, 139
reserved word 146
groups
averaging 129
calculations
overriding default 129
calculations on 132
counting 130
maximum and minimum, finding 131
multiple, using checkmarks 132
querying 128–135
summary operators and 129
summing 131

H

height, changing row 72
Help and Fill lookup option 33
hiding
data 42
hot zones 71
hotlinks *See* DDE

I

inclusion operator (!) 140–144
inclusive links 140–144
Indexed Fields list 35
indexes 11
borrowing 47
case-insensitive 35
case-sensitive 35, 37
creating 34, 35
creating SQL 45
creating, automatically 39
dBASE 27
deleting 36
naming 34, 35
naming SQL 46
returning information on 55
SQL 27, 45, 54
updating 34
inexact operators 107
Info Structure command 55
.INI files 11
Insert & Delete option (table rights) 41
Insert command, Record menu 77
Insert option (table rights) 41
INSERT query (reserved word) 123–126
Inserted tables (temporary) 125
inserting

fields 19
records 77, 123–126

J

Join Tables button 115
joins
See also links
asymmetrical outer 142
exclusive (defined) 140
inclusive 140–144
outer 142
symmetrical outer 143
Just Current Field lookup 32

K

key field indicator 26
key fields
adding 27, 49, 51
deleting 27
key violations 49
keys 26
composite 27, 38
removing 27
keys (fields) 5
status line and scrolling 70
violation, during INSERT query 124
keys, function (table) 149
Keyviol tables 37, 49

L

language drivers 43, 56
Last command, Record menu 70
lightbulb icon 2
LIKE operator 107
lines (in query images)
example elements 116
maximum length in .QBE files 146
multiple 110
creating 110
deleting 111
set queries 138
removing 77
single 109
linking tables 26
links
DDE 8
See also DDE
exclusive (defined) 140
inclusive 140–144
order of processing 141
pasting 76
rules for 144
literal characters 102
Lock command
Record menu 78
locking
columns 71
logical and (query statements) 109, 111
logical or (query statements) 110, 111
Lookup Field text box 32
Lookup Help command
Record menu 33, 79
lookup *See* table lookup
lookup tables 142
creating 31–33
returning information on 55
losing data 50

M

Maintained option (Define Secondary Index) 34
maintaining indexes 34
master passwords 40, 41
changing 41
matching values *See* queries
MAX operator (group summary) 131
maximum
finding in a group 131
line length in .QBE files 146
query image fields 101
tables in fields 118
values, setting 28
.MB files 11, 24
.MDX files 11
memo fields
constraints, size 25
MIN operator (group summary) 131
minimum
finding in a group 131
values, setting 28
modes
Edit 73–77
placement (for example elements) 115
replace (default data entry) 75
View (Table window) 69
money fields 23
converting data to 52
Move Help command
Record menu 79
moving
data 76
fields 19, 22
multi-line queries 110
multiple tables
joining with example elements 118
multi-table relationships 31

N

names
field, in .QBE files 146
table, in .QBE files 146
naming
fields 20, 21
indexes 34, 35
passwords 40
relationships, referential integrity 39
tables 44
.NDX files 11
networks
locks on records 78
private directory, need for 13
running queries 97
temporary tables 13, 98
New command
File menu 13
new fields 19, 20
Next command, Record menu 70
Next Set command, Record menu 70
NO (set comparison operator) 135–136
No check 96
None option (field rights) 42
nonproportional font (Courier) 145
NOT operator 107
example elements and 117
example of use (in figure) 142
Num Lock and keypad keys 151
number fields 23
converting data to 52
numbers 23
decimal code, ASCII 129
decimal portions 103
typing into queries 103
numeric values
constants 29

O

objects
copying 60–64
deleting 66–67
renaming 64–65
sorting 57–60

OEM character set, translating in Database Desktop 33
OLE fields 24, 26
ONLY operator (set comparison) 136
Open command, File menu 11, 69
Open Table button 18, 69
operators 102
! (inclusion) 140–144
multiple 142
+, alphanumeric values and 122
, (AND) 111
.. (wildcard) 109
= (equal to) 106
@ (wildcard) 109
ALL (group summary) 133
AND
example elements and 115
arithmetic (table of) 108, 113
AS 112
AVERAGE (group summary) 129
BLANK 108
CALC 121, 132–135
comparison 106
COUNT (group summary) 133
EVERY (set comparison) 140
EXACTLY (set comparison) 135
inclusion (!) 140–144
multiple 142
inexact 107
LIKE 107
with example elements 117
MAX (group summary) 134
MIN (group summary) 134
NO (set comparison) 135
NOT 107
with example elements 117
ONLY (set comparison) 136
OR 111
OR combined with AND 112
precedence of 112
set comparison (table of) 136
special 107, 111
SUM (group summary) 129, 132
summary (table of) 129
table of 102
TODAY 108
UNIQUE (group summary) 129
wildcard 108, 109
OR operator 111
order, sort *See* sort order
organizing data 17–18
outer joins (defined) 142

P

Paradox
compatibility 31, 38, 52
indexes 11
OEM character set translation 33
parent tables 36, 37
parentheses (for setting precedence) 112
pass-through SQL *See* SQL Editor
Password Security dialog box 40, 41
Auxiliary Passwords option 40
Password Security option 40
passwords 39–42
case sensitivity 40
changing 41, 42
creating 40, 42
deleting 42
naming 40
opening tables with 42
removing 43
Passwords command, Utilities menu 42
Passwords list (Auxiliary Passwords) 42
Paste command, Edit menu 76
Paste Link command, Edit menu 76
pasting
links 76
values 76
paths, aliases 14
patterns, matching characters in 108
persistent field view 75
PERSONAL.DB (sample file used in figure) 119
phone numbers 23, 28
Picture Assistance dialog box 29, 30
Sample Pictures option 30
Picture check box 29
picture strings 28–30
changing 30
characters allowed 29
creating 29
getting help 29
picture validity checks 29, 30
placement mode (for example elements) 115
placing checkmarks 96
plus sign (operator)
alphanumeric values 122
date values 108
pointers
columns and 72
customizing Table view 71
hot zones and 72
multiple selection and 74
rows and 72
scroll lock and 71
Post/Keep Locked command Record menu 78
posting
edits to a table 76
precedence
checkmarks 104
operators 112
Previous command, Record menu 70
Previous Set command, Record menu 70
primary indexes 11, 26
primary keys 5
:PRIV: (private directory) 13
private directory 13
Private Directory command File menu 13
Problems tables 49
Prohibit option (Referential Integrity) 38, 39
prohibiting auxiliary tables 97
properties
Answer table 99, 100, 101
Answer table, restoring 101
Answer table, saving 100
table 71–73
default, restoring 73
Properties menu
Answer Sort 99, 100
Answer Table 99, 101
Delete 73
Restore 73
Save 72
protecting
fields 41
tables 41
punctuation in queries 102
.PX files (primary index) 11

Q

QBE (query by example)
See also queries

defined 93
.QBE files 11, 13
Answer table section in 145
syntax 145–148
queries 6, 93–98
Answer tables *See* Answer tables
AVERAGE (group) 129
CALC with summary operators (table of) 133
calculations
CALC operator and 121
example elements and 117
CHANGETO 123–124, 127
undoing 127
checkmarks *See* checkmarks
COUNT (group) 130
creating new files 13
dates, operations and 108
DELETE 123–124, 125, 126
duplicate records 99, 105, 126
duplicate values, including 133
editing 103
example elements in 114
exclusive links (defined) 140
Fast Queries option 97
field view and 104
fields, including in results 106
file syntax 145–148
function keys for (table of) 150
groups of records 128–135
inclusive links 140–144
INSERT 123–126
Join Tables button 115
logical operations (and/or) 109
matching values
exactly 105
inexactly 107
MAX (group) 131
multi-line 110
operators *See* operators
persistent field view and 104
problems running 96
punctuation in 102
.QBE files 98
query image (defined) 94
See also query images
query statements (defined) 101
See also query statements
renaming fields with AS 112
reserved characters, including 102
results, in Answer table 7
running 34, 96
running, on networks 97
saving 13, 98
selection conditions 101
multiple 111
set 135
comparing 137
defining 137
GroupBy check and 137, 139
multiple lines 138
ONLY operator and 136
Set and the GroupBy check 96
set comparison operators (table of) 136
set definitions and checkmarks 136
sort order
ascending 99
checkmarks for 95
descending 99
setting 100
SQL translation of 90, 98
status dialog box 96
SUM (group) 131
syntax 145–148
tables
adding 95
cascading 95
joining 115
maximum 118
multiple 118
removing 95
source 124
tiling 95
wildcard operators in 108
Query, reserved word in .QBE files 145
query by example 6
See also queries
defined 93
query images
CALC reserved word and 121
defined 94
maximum characters in each field 101
single line and 109, 110
query statements 101
case sensitivity and 105
creating 106
editing 103
literal characters and 102
multiple selection conditions 111
wildcard operators in 108
quotation marks
literal characters and 103
query statements and 102

R

Read Only option
field rights 42
table rights 41
rearranging fields 19
Record menu
Cancel Changes 76
Delete 77
First 70
Insert 77
Last 70
Lock 78
Lookup Help 33
Next 70
Next Set 70
Post/Keep Locked 78
Previous 70
Previous Set 70
records
access order 11
adding 79–82
ascending order 99
buttons to navigate 70
changing 123–124, 127
cutting 76
dBASE, retrieving deleted 76
defined 3
deleting 76, 123–124, 126
descending order 99
display order 11
duplicate 99, 105, 126
groups, queries on 128–135
inclusion operator and 140–144
inserting 123–126
keyed tables and 77
new 77
locking 78
modifying with CHANGETO 127
moving among 70
Paradox, retrieving deleted 76
removing 77
restoring after a CHANGETO query 127
selecting 74
sort order 5
See also sort order
status line and numbers in 70
subtracting 82–84
undoing edits to (restoring) 76
updating view with locks 78

redefining secondary indexes 35
referential integrity 36–39, 52
changing 39
defined 36
enforced 38
lookup tables vs. 31
relationships, naming 39
returning information on 55
settings, saving 38
specifying 37–38
updating tables 38
Referential Integrity dialog box 37
Cascade option 38
Prohibit option 38, 39
Strict Referential Integrity option 38
Referential Integrity option 37, 55
relational tables, defined 4
relationships, multi-table 31
Remove Field arrow 37
removing
field values 77
fields 19
records 77
tables 95
Rename command, Utilities menu 64
Rename dialog box 64
renaming
fields 20, 21, 50
objects 64–65
tables 64–65
reordering fields 19
repeating fields 17
repositioning fields 19
required fields 28
reserved symbols (table of) 102
reserved words
CALC 121
table of 102
table operations (table of) 124
resizing
column headings 72
fields 49, 50
Restore command
Properties menu 73
Restore Original button 30
Restore Original option (Picture Assistance) 30
restoring records 76
Restructure command 48
Restructure Table dialog box 46, 48
Restructure Warning dialog box 49
restructuring tables 46, 47–53
language drivers and 50
SQL 54
reversing an action *See* undoing
rows
removing 77
resizing 72
See also lines (in query images)
running queries
networks 97

S

Sample Pictures option (Picture Assistance) 30
Save As button 43, 53
Save As command, File menu 13
Save button 53
Save command
File menu 13
Properties menu 72
Save Index As dialog box 35
Save Referential Integrity As dialog box 38
Save Table As dialog box 43, 44
Display Table option 44
saving 43
referential integrity settings 38
tables 43–44
tables, to different directories 44
values *See* posting
scientific notation 107
scroll bars, tables 70
scroll locks
defined 71
placing 71
scrolling
columns 70
display record numbers 70
search and replace
CHANGETO, similarity to 123–124, 127
searching *See* queries
secondary indexes 11, 33–36
composite 34, 35
creating 34, 35
deleting 36
maintaining 34
naming 35
redefining 35
returning information on 55
Secondary Indexes option 34, 55
Select All command, Edit menu 74
selecting
columns, all values in 76
fields 74
multiple fields, keys for 149
records 74
tables, all values in 76
selection conditions 101
example elements and 116, 119
group queries and 134
multiple 111
reversing with NOT 107
single line and 109, 110
server (DDE) 8
set comparison operators (table of) 136
set queries 135
comparison operators 136
short fields 23
converting data to 52
shortcut keys
keypad combinations (table of) 152
single records 105
small tables 18
Sort command, Utilities menu 57
sort order 5, 43
Answer tables and 99, 100
ascending (default) 95, 99
ASCII 129
descending 96
overriding 33
.QBE files and 146
queries 100
Sort Table dialog box 58
sorting
objects 57–60
tables 57–60
sound (binary fields) 24
source tables
adding from 124
example elements and 125
special operators 107
specifying
field types 22
referential integrity 37–38
validity checks 28
SQL
copying tables 61
creating new files 13
creating tables 44
field types 26
files 15
indexes 27, 45, 46, 54

local tables 91
naming indexes 46
restructuring tables 54
SQL Editor 15, 87–91
statements 87–91
translated from QBE 90, 98
valid field types 26
validity checks 31
.SQL files 11
starting Database Desktop 9
stationary columns 71
statistics, group 132
Strict Referential Integrity option 38
Strict Translation command, Table menu 85
strings 107
character 28
picture 28–30
Structure Information dialog box 31, 55
structures, table
returning information on 55–56
structuring tables 19, 46
Subtract command, Utilities menu 82
Subtract dialog box 82
subtracting records 82–84
SUM operator (group summary) 131
summary operators (table of) 129
summing groups 131
super tab keys 150
symmetrical outer joins 143

T

Table Language dialog box 43
Table Language option 43, 56
table lookup 78–79
All Corresponding Fields 32
Fill No Help option 32
Help and Fill option 33
Just Current Field 32
Lookup Help command 33
types of (table) 33
Table Lookup dialog box 32
Table Lookup option 32–33, 55
Table menu
Strict Translation 85
table properties
changing 71–73
default, restoring 73
undoing changes 73
Table Properties list
Password Security option 40
Referential Integrity option 37
Secondary Indexes option 34
Table Language option 43
Table Lookup option 32–33
Validity Checks option 28–30
table rights 41
table type 19
table utilities 55–56
Table window 69
editing keys for (table of) 151
tables 3–5, 18, 46
adding to Query window 95
Answer 7, 98–101
renaming 145
saving 145
arranging in Query window 95
cascading 95
changing 47
characters, translating 33
copying 60–64
creating 17–47
creating SQL 44
customizing 71–73
dBASE
creating 147
field types (table of) 25–26
DDE link to 8
defined 3
deleting 66–67
editing 73–77
emptying 84
field view and 74
joining 115, 118
key values, status line and scrolling 70
Keyviol, during INSERT query 124
linked 141
locking columns 71
looking up values 31–33, 78–79
lookup *See* lookup table
master (in links) 141
moving within 70
multiple 118
names, in .QBE files 146
naming 44
opening 69
opening password-protected 42
operations on 123
Paradox
field types (table of) 23–25
protecting 41
records *See* records
relational 4
removing from Query window 95
renaming 64–65
repositioning, in Table window 72
restructuring 47–53
restructuring SQL 54
returning information about 55–56
sample 4
saving 43–44
saving, to different directories 44
scrolling columns of 70
Select File dialog box and 95
small 18
sort order 5
sorting 57–60
source
adding from 124
INSERT queries and 125
structure vs. view 99
super tab keys for navigating 150
target
example elements and 125
INSERT queries and 125
target, creating with DOS 125
temporary 98, 124
Changed 127
Deleted 126
Inserted 125
saving a copy 125
tiling 95
translating characters 33
undoing changes 73
updating 38
values, editing 73–77
view
customizing 71–73
default, restoring 73
vs. structure 71
viewing 69
target tables
example elements and 125
telephone numbers 23, 28
templates, data entry 28
temporary tables 48, 98
Changed 127
Deleted 126
Errorins, Errordel, Errorchg 123
Inserted 125
Keyviol, during INSERT query 124

Test Value option (Picture Assistance) 30
text
formatting 24
tile, open Table or Query windows 95
Tile command, Window menu 10
Tile Tables command
View menu 95
tiling tables 95
TODAY operator 108
computer calendar and 107
Toolbars
application window 10
Query window 94
SQL Editor 88
Table window 70
translating OEM and ANSI characters 33
trends, set queries and 135
triangle *See* scroll lock
trimming data 49
.TV files (Paradox table view properties) 11, 72, 73, 100
.TVF files (dBASE table view properties) 11, 72, 73, 100
type conversion 49, 50–52
TYPE, reserved word (.QBE files) 147

U

unauthorized users 39
underscore characters
example elements and 114
Undo command, Edit menu 76
undoing
CHANGETO query 127
DELETE query 125
Edit | Undo 76
edits 76
Record | Cancel Changes 76
table view changes 73
UNIQUE operator (group summary) 129
Update option (table rights) 41
updating
indexes 34
tables 38
utilities 55–56
Utilities menu
Add 79
Copy 60
Delete 66
Empty 84
Passwords 42
Rename 64
Sort 57
Subtract 82

V

.VAL files 11
validity checks 28–30
borrowing 47
changing 50
picture 29, 30
removing 31
returning information on 55
specifying 28
SQL 31
Validity Checks option 28–30, 55
values
absence of 108
alphanumeric, calculations on 122
average, calculating 130
blank, finding 107
blank spaces, with 103
constants 121
copying 76
cutting, copying, and pasting 76
date 108
DDE links and 76
deleting 76
duplicate
group queries 132, 133, 134
editing 73–77
example elements, changing 128
example elements, defining 116
inexact, matching 107
literal 102
new, creating with INSERT 124
pasting 76
posting 76
queries, exact matches and 105
range of, and example elements 116
ranges, matching 106
removing with Delete 77
saving 76
string 107
typing to replace entries 73
undoing edits to (restoring) 76
unique
COUNT operator and 133
variable (example elements) 113
variable-length fields 24
Verify Syntax option (Picture Assistance) 30
version compatibility 38
Paradox 3.5 31
view
field *See* field view
persistent field 75
table 71
customizing 73–77
default, restoring 73
vs. structure of tables 99
View menu
Cascade Tables 95
Edit Data 73
Field View 75
Tile Tables 95
viewing
field types 20
violation of key values, during INSERT query 124
VOLUMES.DB (sample table used in figure) 138

W

whole numbers 23
width, changing column 72
wildcard operators 108
.. operator 109
@ operator 109
Window menu 10
windows
application *See* application window
arranging 10, 95
Query 93
SQL Editor 87–91
Table
editing keys for (table of) 151
Table View 69
Windows Control Panel 108
Working Directory command
File menu 13

X

.X*nn* files (composite secondary index) 11
.X*nn* files (secondary index) 11

Y

.Y*nn* files (composite secondary index) 11
.Y*nn* files (secondary index) 11

User's Guide

Version 5.1

Borland®
Database Desktop

Borland International, Inc., 100 Borland Way
P.O. Box 660001, Scotts Valley, CA 95067-0001

Borland may have patents and/or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents.

COPYRIGHT © 1992, 1995 Borland International, Inc. All rights reserved. All Borland product names are trademarks or registered trademarks of Borland International, Inc. Other brand and product names are trademarks or registered trademarks of their respective holders.

Printed in the U.S.A.

1E0R195
9495969798-987654321
W1

Contents

Introduction 1

Inside this manual 1
Conventions 2
Using this book 2

Chapter 1
Database concepts 3

Understanding tables 3
Relational tables. 4
Using indexes 4
Paradox primary indexes 5
Paradox secondary indexes 5
dBASE indexes 5
Querying databases 6
Query by example 6
Answer tables. 7
SQL queries 7
Using DDE links 8

Chapter 2
The Database Desktop window 9

Starting Database Desktop 9
The Database Desktop window 9
The application window Toolbar 10
Arranging windows 10
Managing files. 11
Opening files 11
The Drive (or Alias) list 12
The File Type list 12
Tables with passwords 12
Creating new files 12
Saving files. 13
Setting up a working directory 13
Setting up a private directory. 13
Aliases 14
Creating an alias 14
Changing an alias definition 15
Removing an alias 15
.SQL files 15

Chapter 3
Creating tables 17

Planning tables 17
Paradox, dBASE, or SQL table?. 18
Creating a new table 18
Guidelines for creating tables. 19
Creating a simple Paradox table 20
Defining fields 20
Field names 20
Rules governing Paradox field names . . . 20
Rules governing dBASE field names 21
Rules governing SQL field names 21
To delete a field 21
To insert a field 21
To edit a field name 21
To reorder fields 22
Specifying field type. 22
Paradox field types and sizes 22
BLOB fields 25
Valid dBASE field types. 25
Valid SQL field types 26
Keys in Paradox tables 26
Defining keys 26
Removing keys 27
A dBASE table's index 27
An SQL table's index 27
Specifying validity checks 28
Picture patterns 28
Getting assistance with pictures. 29
Viewing a field's validity checks 30
Removing a validity check 31
Creating table lookup 31
The difference between table lookup and referential integrity 31
Why use table lookup?. 31
Lookup options 32
Defining secondary indexes 33
The Maintained option 34
The Case Sensitive option 35
Composite secondary indexes 35
Modifying secondary indexes 35
Erasing secondary indexes 36
Defining referential integrity. 36
Update Rule options 38
Using Strict Referential Integrity 38
Saving the referential integrity relationship . 38
Changing or deleting referential integrity . . 39
Creating self-referential integrity. 39
Establishing password security 39
Table rights 41
Field rights. 41
Using passwords 42
Choosing a table language 43
Saving the new table 43
Creating an SQL table. 44
Creating indexes on SQL tables 45
Naming SQL indexes 46
Borrowing a table structure 46
Borrow options. 47

Restructuring Paradox tables 47
General rules of restructuring 48
Shortening a field 50
Adding fields to an existing table 50
Deleting fields from an existing table 50
Editing a field name in an existing table 50
Converting a non-keyed field to a keyed field . 51
Changing field types in Paradox tables 51
Alpha field conversions 52
Number, money, and short field conversions 52
Date field conversions 52
Restructuring tables that are linked by referential integrity 52
Saving the restructured table 53
Restructuring an SQL table 54
Prefixing the index name with the table name 54
Getting table information 55

Chapter 4
Managing objects 57

Sorting a table 57
Sorting keyed tables 57
Sorting unkeyed tables. 57
Using Sort 58
Specifying the sort order 58
Adding fields to the Sort Order list 59
Removing selected fields from the Sort Order list. 59
Removing all fields from the Sort Order list . 59
Rearranging fields in the Sort Order list . . . 59
Using ascending or descending sort order . . 59
The Same Table/New Table option. 60
The Sort Just Selected Fields option 60
The Display Sorted Table option. 60
Performing the Sort 60
Sorting on a network 60
Copying objects 60
Copying SQL tables 61
Copying on a network 62
Copying referential integrity 62
Copying to a different table type 63
Field type conversions. 63
Index conversions 64
Renaming objects 64
Rules for renaming objects 65
Using Rename on a network 65
Deleting objects 66
Using Delete on a network 67

Chapter 5
Entering and editing data 69

Viewing tables. 69
Moving through a table's records 70
Using scroll bars 70
Using scroll lock 71
Customizing a table view. 71
Rearranging and resizing columns 72
Resizing rows 72
Resizing column headings 72
Saving a custom view 72
Undoing changes to a view 73
Restoring the default view 73
Editing data 73
Selecting fields and records 74
Selecting multiple fields 74
Selecting all records 74
Field view 74
Exiting field view. 75
Persistent field view 75
Cutting, copying, and pasting data 76
Using Undo. 76
Removing entries from fields 76
Inserting and deleting records 77
Exiting Edit mode 77
Field types 77
Locking Records. 78
Manually locking records. 78
Fields with validity checks. 78
Looking up table values 78
Using Move Help 79
Adding records to a different table. 79
Adding records to a different table type 81
Adding records to keyed tables 81
Append and Update options. 81
Adding on a network 82
Subtracting records 82
Rules for subtracting records. 83
Subtracting on a network. 83
Using Empty on a network. 84
Limiting character sets 85

Chapter 6
Running SQL statements 87

What is the SQL Editor? 87
Opening the SQL Editor 89
Specifying an alias. 89
Entering an SQL statement 89
Running an SQL statement 90
Saving an SQL statement 90
Viewing the SQL translation of a QBE query. . 90
Using SQL against local tables 91

Chapter 7
Querying table data with QBE 93

Using the Query window 93
Using the Query window Toolbar 94
Creating a new query 94
Adding tables 95
Arranging tables 95
Moving between tables 95
Removing tables 95
Including fields in a query 95
Placing checkmarks 96
Running the query 96
Setting query options 96
Handling table updates 97
Setting auxiliary table options 97
Choosing a default checkmark. 98
Setting SQL options 98
For more information on using SQL data with Database Desktop, search online help for the subject "SQL" and the topic "SQL."Viewing the Answer table 98
Saving the query 98
Modifying the Answer table 99
Sorting Answer tables 99
Sort order of records. 99
Using ascending order 99
Using descending order. 99
Performing the sort 100
Changing the view of an Answer table 100
Changing the name or table type. 101
Renaming an Answer table. 101
Changing the Answer table type 101
Defining queries in the Query window 101
Including punctuation 103
Typing numbers into queries. 103
Editing queries 103
Using field view. 104
Using persistent field view 104
Using different checkmark types. 104
Displaying duplicate records 105
Changing the sort order 105
Matching exact values 105
Typing values in queries. 106
Matching ranges of values. 106
Using comparison operators 106
Matching inexact values 107
Using the LIKE operator. 107
Using the NOT operator. 107
Using the BLANK operator 108
Using the TODAY operator. 108
Matching patterns of characters 108
Matching a series of characters 109
Matching single characters 109
Using multiple conditions in queries 109
Using multiple lines 110
Creating additional lines 110
Deleting lines 111
Using multiple checkmarks. 111
Using special operators. 111
Using the AND (,) operator 111
Using the OR operator 111
Combining OR with AND conditions. 112
Using the AS operator. 112
Using arithmetic expressions 112
Using example elements 113
Creating an example element 114
Using the Toolbar 115
Using the keyboard 115
Editing example elements 115
Defining the value of example elements. . . . 116
Defining the value as a range 116
Calculations using example elements 117
Using NOT or LIKE with an example element. . 117
Using example elements with dates 118
Using multiple tables in a query 118
Using selection conditions 119
Calculating with queries 121
Calculating math expressions 121
Combining calculations. 121
Using CALC with alphanumeric values 122
Performing table operations. 123
Inserting records 124
Creating an INSERT query 125
Undoing an INSERT query 125
Deleting records 126
Creating a DELETE query. 126
A sample DELETE query 126
Modifying records 127
Creating a CHANGETO query. 127
Undoing changes made by CHANGETO . .127
Changing values with example elements. . .128
Changing multiple values with CHANGETO . . 128
Performing queries on groups of records . . . 128
Using summary operators 129
Averaging groups of records 129
Counting groups of records. 130
Grouping records by a maximum or minimum value 131
Summing groups of records 131
Calculating group statistics 132
Forming groups to calculate 132
Counting unique values. 133
Counting all values. 133

Summarizing group values 134
Grouping multiple fields 134
Using sets 135
Creating a Set query 136
Comparing records to a set 136
Defining a set 137
Comparing a set. 137
A sample Set query. 137
Defining a two-line set 138
Using the GroupBy check 139
Using inclusive links 140
An example of an inclusive link 141
Processing order for links 141
Using multiple ! operators. 142
Using ! in arithmetic expressions. 143
Avoiding ! operator conflicts 144
Rules for linking tables. 144
QBE file syntax 145
Defining the Answer table name. 145
Overriding the sort order 146
Defining tables and fields 146
Wrapping long lines 146
Creating a dBASE Answer table 147
Translating field types. 147

Appendix A
Using the keyboard 149

Keyboard techniques. 149
Data entry shortcuts 149
Function keys 150
Super tab 151
Keyboard shortcuts for table operations . . . 151
Ctrl+letter shortcuts. 151
Keypad shortcut keys 152

Tables

2.1 File extensions 11
3.1 Paradox field types 23
3.2 dBASE field types. 25
3.3 Paradox validity checks 28
3.4 Picture pattern characters 29
3.5 Lookup types and options. 33
3.6 Auxiliary password field and table rights . . 42
3.7 Changing field types in Paradox tables 51
3.8 Converting strings to dates 52
4.1 Copying from a Paradox to a dBASE table. . 63
4.2 Copying from a dBASE to a Paradox table. . 64
5.1 Cut, copy, and paste operation commands. . 76
7.1 Checkmark types 95
7.2 Reserved symbols and words, and query operators 102
7.3 Using backslash (\) characters 103
7.4 Checkmark precedence 104
7.5 Comparison operators 106
7.6 Arithmetic operations on dates 108
7.7 The .. wildcard operator 109
7.8 The @ wildcard operator 109
7.9 Arithmetic operators 113
7.10 Reserved words for table operations 123
7.11 Types of summary operators 129
7.12 Combining summary operators with CALC 133
7.13 Set operators 136
7.14 Querying a Paradox table to a dBASE Answer table 147
7.15 Querying a dBASE table to a Paradox Answer table 148
A.1 Data entry shortcut keys 149
A.2 Function key actions 150
A.3 Using Super tab 151
A.4 *Ctrl+letter* shortcut combinations 151
A.5 Keypad combinations. 152

Figures

1.1 A table 4
1.2 A query and its results 7
2.1 The Database Desktop application window . 10
2.2 Application window Toolbar 10
2.3 The Alias Manager dialog box 14
3.1 The Create Table dialog box 19
3.2 The Picture Assistance dialog box 29
3.3 Changing a secondary index 36
3.4 Referential integrity 36
3.5 The Auxiliary Passwords dialog box 41
3.6 The Enter Password(s) dialog box 43
3.7 The Save Table As dialog box. 44
3.8 The Create Table dialog box for SQL tables 45
3.9 The Define Index dialog box for SQL indexes 45
3.10 The Restructure table dialog box. 48
3.11 The Restructure Warning dialog box 49
3.12 The Restructure Table dialog box for SQL tables 54
3.13 The Structure Information dialog box. 55
3.14 The Structure Information dialog box for SQL tables 56
4.1 The Sort Table dialog box 58
4.2 The Copy dialog box 61
4.3 Copying to or from an SQL table. 62
4.4 The Rename dialog box 65
4.5 The Delete dialog box 66
5.1 The Table window Toolbar 69
5.2 A scroll lock in the Table window 71
5.3 Hot zone pointers in the Table window . . . 72
5.4 The *Customer* table in Edit mode 74
5.5 The Add dialog box 80
5.6 The Subtract dialog box 83
5.7 The Empty dialog box 84
6.1 The SQL Editor 88
6.2 The SQL Editor and an *Answer* table 88
6.3 SQL Editor Toolbar. 88
6.4 The Select Alias dialog box 89
6.5 SQL statement in the SQL Editor. 90
6.6 SQL translation of a QBE query 91
7.1 The Query window Toolbar 94
7.2 The Query Options dialog box 97
7.3 The Sort Answer dialog box 100
7.4 Multiple lines in a query image. 110
7.5 Combined OR and AND operations 112
7.6 Example elements in single-table queries. . 114
7.7 Using an example element to match a specific value 116
7.8 An example element for a range of values . 117
7.9 Combining example elements with operators 117
7.10 Example elements and dates. 118
7.11 Example elements used to join tables 119
7.12 A multi-line, multi-table query 120
7.13 Example elements on multiple lines 121
7.14 A simple example element. 122
7.15 Using CALC to concatenate values. 123
7.16 Inserting records into a target table. 125
7.17 Deleting records with a DELETE query . . . 127
7.18 CHANGETO with multiple example elements. 128
7.19 A sample AVERAGE query 130
7.20 A sample COUNT query. 131
7.21 A sample MAX query. 131
7.22 A sample SUM query 132
7.23 An example query using CALC COUNT . . 133
7.24 An example query using CALC COUNT ALL. 134
7.25 A sample query using CALC MIN 134
7.26 Multiple fields in a group query 135
7.27 A sample Set query 138
7.28 A two-line Set query definition 138
7.29 Joining another table to a set query. 139
7.30 A Set query using the GroupBy check 140
7.31 Using the inclusion operator ! 141
7.32 The ! operator in Customer 142
7.33 The ! operator in Bookord 142
7.34 Multiple ! operators in a query 143
7.35 A query using the ! operator in an expression. 144
7.36 A query and its .QBE file 145
7.37 A query for an Answer table in dBASE format 147