Informix Product Family
Informix
Version 11.70

IBM Informix DB-Access User's Guide

IBM

SC27-3535-02

Informix Product Family
Informix
Version 11.70

IBM Informix DB-Access User's Guide

SC27-3535-02

Note

Before using this information and the product it supports, read the information in “Notices” on page D-1.

Edition

This edition replaces SC27-3535-01.

Contents

Introduction

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

About this publication

This publication describes how to use the DB-Access utility to access, modify, and retrieve information from IBM® Informix® database servers. Chapter 1, "Getting started with DB-Access," on page 1-1 explains how to create and work with the demonstration databases provided with your Informix database server.

Important: Use DB-Access with the current version of an Informix database server. If you use DB-Access with a database server from a different version, you might obtain inconsistent results, such as when you use a version that does not support long identifiers with a version that does.

Types of users

This publication is written for the following users:

- Database users
- Database administrators
- Database-application programmers

This publication assumes that you have the following background:

- A working knowledge of your computer, your operating system, and the utilities that your operating system provides
- Some experience working with relational databases or exposure to database concepts
- Some experience with computer programming

If you have limited experience with relational databases, SQL, or your operating system, see the following IBM Informix manuals:

- *IBM Informix GLS User's Guide*
- *IBM Informix Guide to SQL: Reference*
- *IBM Informix Guide to SQL: Syntax*
- *IBM Informix Guide to SQL: Tutorial*

Software dependencies

This publication assumes that you are using one of the IBM Informix, Version 11.70 database servers.

Assumptions about your locale

IBM Informix products can support many languages, cultures, and code sets. All the information related to character set, collation and representation of numeric data, currency, date, and time that is used by a language within a given territory and encoding is brought together in a single environment, called a Global Language Support (GLS) locale.

The IBM Informix OLE DB Provider follows the ISO string formats for date, time, and money, as defined by the Microsoft OLE DB standards. You can override that default by setting an Informix environment variable or registry entry, such as **DBDATE**.

If you use Simple Network Management Protocol (SNMP) in your Informix environment, note that the protocols (SNMPv1 and SNMPv2) recognize only English code sets. For more information, see the topic about GLS and SNMP in the *IBM Informix SNMP Subagent Guide*.

The examples in this publication are written with the assumption that you are using one of these locales: en_us.8859-1 (ISO 8859-1) on UNIX platforms or en_us.1252 (Microsoft 1252) in Windows environments. These locales support U.S. English format conventions for displaying and entering date, time, number, and currency values. They also support the ISO 8859-1 code set (on UNIX and Linux) or the Microsoft 1252 code set (on Windows), which includes the ASCII code set plus many 8-bit characters such as é, è, and ñ.

You can specify another locale if you plan to use characters from other locales in your data or your SQL identifiers, or if you want to conform to other collation rules for character data.

For instructions about how to specify locales, additional syntax, and other considerations related to GLS locales, see the *IBM Informix GLS User's Guide*.

What's new in DB-Access for IBM Informix, Version 11.70

This publication includes information about new features and changes in existing functionality.

For a complete list of what's new in this release, see the release notes or the information center at http://publib.boulder.ibm.com/infocenter/idshelp/v117/topic/com.ibm.po.doc/new_features.htm.

Table 1. What's new in IBM Informix DB-Access User's Guide for 11.70.xC3

Overview	Reference
Time series data in the **stores_demo** database The **stores_demo** database has three new tables that contain time series data. You can disable the creation of the time series tables when you create the **stores_demo** database by using the **-nots** option.	"Command-line syntax" on page 1-7

Table 2. What's new in IBM Informix DB-Access User's Guide for 11.70.xC1

Overview	Reference
New editions and product names IBM Informix Dynamic Server editions were withdrawn and new Informix editions are available. Some products were also renamed. The publications in the Informix library pertain to the following products: • IBM Informix database server, formerly known as IBM Informix Dynamic Server (IDS) • IBM OpenAdmin Tool (OAT) for Informix, formerly known as OpenAdmin Tool for Informix Dynamic Server (IDS) • IBM Informix SQL Warehousing Tool, formerly known as Informix Warehouse Feature	For more information about the Informix product family, go to http://www.ibm.com/software/data/informix/.

Example code conventions

Examples of SQL code occur throughout this publication. Except as noted, the code is not specific to any single IBM Informix application development tool.

If only SQL statements are listed in the example, they are not delimited by semicolons. For instance, you might see the code in the following example:

```
CONNECT TO stores_demo
...

DELETE FROM customer
   WHERE customer_num = 121
...

COMMIT WORK
DISCONNECT CURRENT
```

To use this SQL code for a specific product, you must apply the syntax rules for that product. For example, if you are using an SQL API, you must use EXEC SQL at the start of each statement and a semicolon (or other appropriate delimiter) at the end of the statement. If you are using DB–Access, you must delimit multiple statements with semicolons.

Tip: Ellipsis points in a code example indicate that more code would be added in a full application, but it is not necessary to show it to describe the concept being discussed.

For detailed directions on using SQL statements for a particular application development tool or SQL API, see the documentation for your product.

Additional documentation

Documentation about this release of IBM Informix products is available in various formats.

You can access or install the product documentation from the Quick Start CD that is shipped with Informix products. To get the most current information, see the Informix information centers at ibm.com®. You can access the information centers and other Informix technical information such as technotes, white papers, and IBM Redbooks® publications online at http://www.ibm.com/software/data/sw-library/.

Compliance with industry standards

IBM Informix products are compliant with various standards.

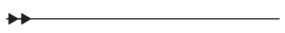
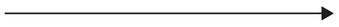
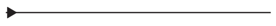
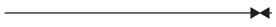
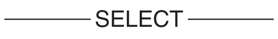
IBM Informix SQL-based products are fully compliant with SQL-92 Entry Level (published as ANSI X3.135-1992), which is identical to ISO 9075:1992. In addition, many features of IBM Informix database servers comply with the SQL-92 Intermediate and Full Level and X/Open SQL Common Applications Environment (CAE) standards.

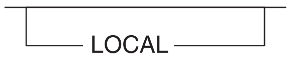
The IBM Informix Geodetic DataBlade® Module supports a subset of the data types from the *Spatial Data Transfer Standard (SDTS)—Federal Information Processing Standard 173*, as referenced by the document *Content Standard for Geospatial Metadata*, Federal Geographic Data Committee, June 8, 1994 (FGDC Metadata Standard).

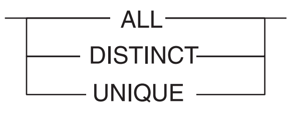
Syntax diagrams

Syntax diagrams use special components to describe the syntax for statements and commands.

Table 3. Syntax Diagram Components

Component represented in PDF	Component represented in HTML	Meaning
	`>>----------------------`	Statement begins.
	`----------------------->`	Statement continues on next line.
	`>-----------------------`	Statement continues from previous line.
	`-----------------------><`	Statement ends.
	`--------SELECT----------`	Required item.

	`--+-----------------+---`
	`  '------LOCAL------'`
		Optional item.

	`---+-----ALL-------+---`
	`   +--DISTINCT-----+`
	`   '---UNIQUE------'`
		Required item with choice. Only one item must be present.

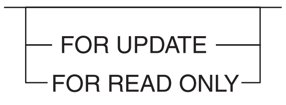
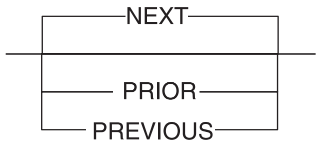
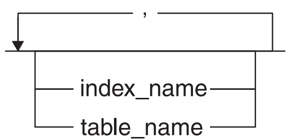
	`---+------------------+---`
	`    +--FOR UPDATE-----+`
	`    '--FOR READ ONLY--'`
		Optional items with choice are shown below the main line, one of which you might specify.

Table 3. Syntax Diagram Components (continued)

Component represented in PDF	Component represented in HTML	Meaning

	`    .---NEXT---------.`
	`----+----------------+---`
	`    +---PRIOR--------+`
	`    '---PREVIOUS-----'`
		The values below the main line are optional, one of which you might specify. If you do not specify an item, the value above the line is used by default.

	` .-------,-----------.`
	` V                   |`
	`---+-----------------+---`
	`    +---index_name---+`
	`    '---table_name---'`
		Optional items. Several items are allowed; a comma must precede each repetition.

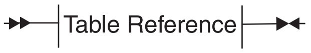
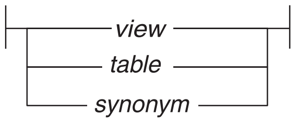
	`>>-| Table Reference |-><`	Reference to a syntax segment.

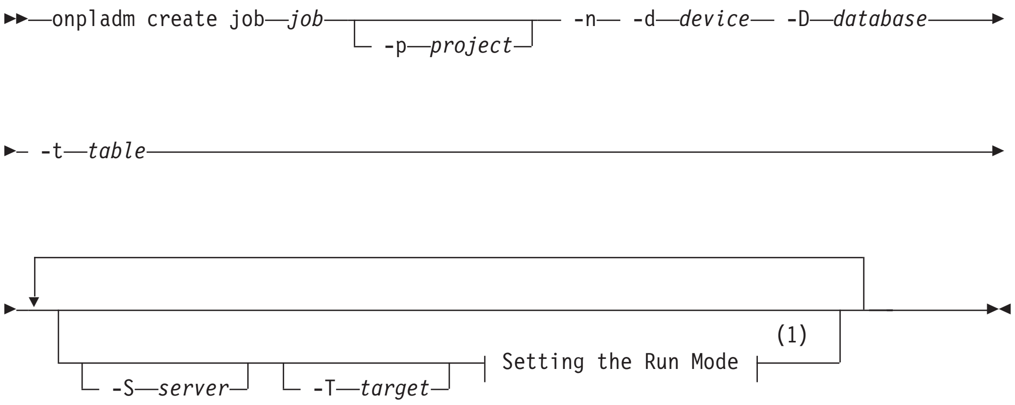
Table Reference 	`Table Reference`
	`|--+-----view--------+--|`
	`   +------table------+`
	`   '----synonym------'`
		Syntax segment.

How to read a command-line syntax diagram

Command-line syntax diagrams use similar elements to those of other syntax diagrams.

Some of the elements are listed in the table in Syntax Diagrams.

Creating a no-conversion job

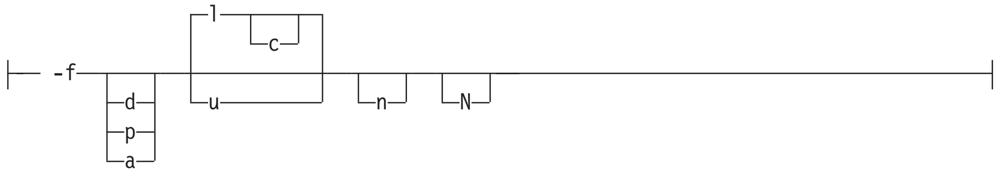


Notes:

1 See page Z-1

This diagram has a segment named “Setting the Run Mode,” which according to the diagram footnote is on page Z-1. If this was an actual cross-reference, you would find this segment on the first page of Appendix Z. Instead, this segment is shown in the following segment diagram. Notice that the diagram uses segment start and end components.

Setting the run mode:



To see how to construct a command correctly, start at the upper left of the main diagram. Follow the diagram to the right, including the elements that you want. The elements in this diagram are case-sensitive because they illustrate utility syntax. Other types of syntax, such as SQL, are not case-sensitive.

The Creating a No-Conversion Job diagram illustrates the following steps:

1. Type **onpladm create job** and then the name of the job.
2. Optionally, type **-p** and then the name of the project.
3. Type the following required elements:
   - **-n**
   - **-d** and the name of the device
   - **-D** and the name of the database
   - **-t** and the name of the table
4. Optionally, you can choose one or more of the following elements and repeat them an arbitrary number of times:
   - **-S** and the server name
   - **-T** and the target server name
   - The run mode. To set the run mode, follow the Setting the Run Mode segment diagram to type **-f**, optionally type **d**, **p**, or **a**, and then optionally type **l** or **u**.
5. Follow the diagram to the terminator.

Keywords and punctuation

Keywords are words reserved for statements and all commands except system-level commands.

When a keyword appears in a syntax diagram, it is shown in uppercase letters. When you use a keyword in a command, you can write it in uppercase or lowercase letters, but you must spell the keyword exactly as it appears in the syntax diagram.

You must also use any punctuation in your statements and commands exactly as shown in the syntax diagrams.

Identifiers and names

Variables serve as placeholders for identifiers and names in the syntax diagrams and examples.

You can replace a variable with an arbitrary name, identifier, or literal, depending on the context. Variables are also used to represent complex syntax elements that are expanded in additional syntax diagrams. When a variable appears in a syntax diagram, an example, or text, it is shown in *lowercase italic*.

The following syntax diagram uses variables to illustrate the general form of a simple SELECT statement.

```
►►──SELECT──column_name──FROM──table_name─────────────────────────────►◄
```

When you write a SELECT statement of this form, you replace the variables *column_name* and *table_name* with the name of a specific column and table.

How to provide documentation feedback

You are encouraged to send your comments about IBM Informix user documentation.

Use one of the following methods:

- Send email to docinf@us.ibm.com.
- In the Informix information center, which is available online at http://www.ibm.com/software/data/sw-library/, open the topic that you want to comment on. Click the feedback link at the bottom of the page, fill out the form, and submit your feedback.
- Add comments to topics directly in the information center and read comments that were added by other users. Share information about the product documentation, participate in discussions with other users, rate topics, and more!

Feedback from all methods is monitored by the team that maintains the user documentation. The feedback methods are reserved for reporting errors and omissions in the documentation. For immediate help with a technical problem, contact IBM Technical Support at http://www.ibm.com/planetwide/.

We appreciate your suggestions.

Chapter 1. Getting started with DB-Access

These topics introduce a new user to DB-Access. It provides information about how to set up your DB-Access environment and the demonstration database.

What is DB-Access?

DB-Access provides a user interface for entering, executing, and debugging Structured Query Language (SQL) statements and Stored Procedure Language (SPL) routines.

SQL and SPL allow you to perform data-definition tasks, such as specifying the number and type of data columns in a table, and data-management tasks, such as storing, viewing, and changing table data. The DB-Access interface allows you to apply powerful IBM Informix extensions to SQL and SPL.

You can use DB-Access for the following aspects of database processing:
- Using ad hoc queries that you execute infrequently
- Connecting to one or more databases, transferring data between the database and external text files, and displaying information about a database
- Displaying system catalog tables and the Information Schema, which are explained in the *IBM Informix Guide to SQL: Reference*
- Practicing the statements and examples provided in the *IBM Informix Guide to SQL: Tutorial* or the *IBM Informix Database Design and Implementation Guide*
- Testing applications that you intend to store for use in a production environment

Important: DB-Access is not intended as an application-development environment. You cannot branch conditionally or loop through SQL statements when you run them within DB-Access.

Requirements for the Informix Client Software Development Kit DB-Access utility

The client-side version of DB-Access that is included in the Client Software Development Kit has setup requirements.

The client-side version of the DB-Access utility can directly access the user-defined and system databases of Informix database server versions that the Client SDK instance supports. The same user interface that this documentation describes for DB-Access as a server utility is also supported by the dbaccess client-side utility for use with Informix database server instances with which Client SDK has established a client-server connection.

The Informix Client Software Development Kit DB-Access utility requires the following setup steps:
- Set the sqlhosts information.
- Set the INFORMIXDIR environment variable to the Client SDK installation directory.
- Set the INFORMIXSERVER environment variable for a default server name.

Related concepts:

The sqlhosts file and the SQLHOSTS registry key (Administrator's Guide)

Related reference:

INFORMIXDIR environment variable (SQL Reference)

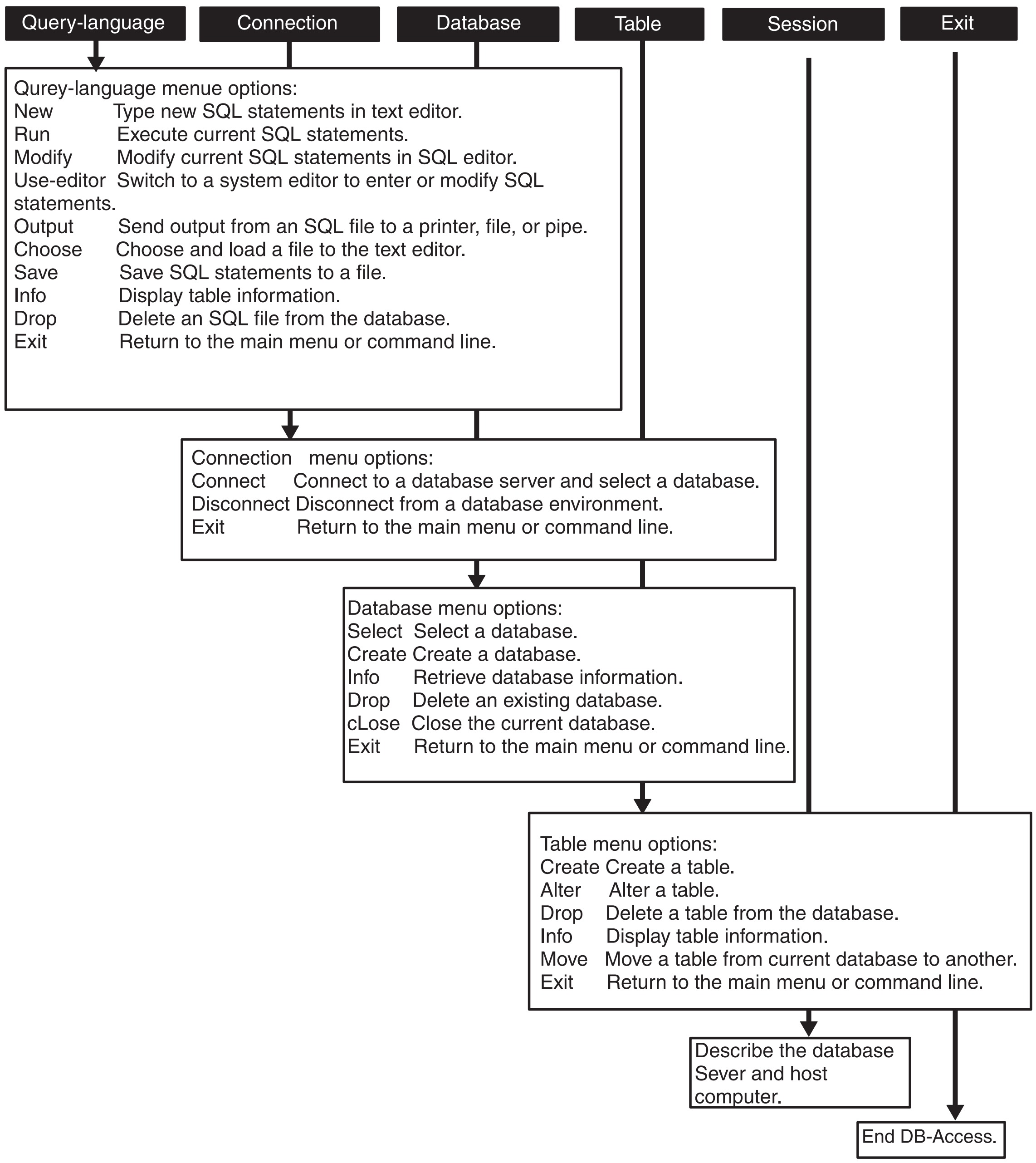
INFORMIXSERVER environment variable (SQL Reference)

Using DB-Access

You can use the DB-Access user interface to:

- Run statements interactively, discarding them after you achieve the results you wanted, or saving them in a file for repetitive execution.
- Type statements directly in the DB-Access text-entry screen or your preferred text editor.
- Start DB-Access in menu mode and select options from the menus.

The following figure illustrates the top two levels of the DB-Access menu hierarchy. The view of available options summarizes the types of database tasks that you can accomplish with DB-Access.



Query-language
Connection
Database
Table
Session
Exit
Qurey-language menue options:
New Type new SQL statements in text editor.
Run Execute current SQL statements.
Modify Modify current SQL statements in SQL editor.
Use-editor Switch to a system editor to enter or modify SQL statements.
Output Send output from an SQL file to a printer, file, or pipe.
Choose Choose and load a file to the text editor.
Save Save SQL statements to a file.
Info Display table information.
Drop Delete an SQL file from the database.
Exit Return to the main menu or command line.
Connection menu options:
Connect Connect to a database server and select a database.
Disconnect Disconnect from a database environment.
Exit Return to the main menu or command line.
Database menu options:
Select Select a database.
Create Create a database.
Info Retrieve database information.
Drop Delete an existing database.
cLose Close the current database.
Exit Return to the main menu or command line.
Table menu options:
Create Create a table.
Alter Alter a table.
Drop Delete a table from the database.
Info Display table information.
Move Move a table from current database to another.
Exit Return to the main menu or command line.
Describe the database Sever and host computer.
End DB-Access.

Figure 1-1. The DB-Access menu hierarchy

For detailed submenu instructions, see the following topics.

Option	Instructions
Query-language	Chapter 3, "The Query-language option," on page 3-1
Database	Chapter 4, "The Database option," on page 4-1
Table	Chapter 5, "The Table option," on page 5-1
Session	Chapter 6, "The Connection and Session options," on page 6-1

Option	Instructions
Connection	Chapter 6, “The Connection and Session options,” on page 6-1

Set up DB-Access

These topics contain information about setting up the environment and environment variables that you need to use DB-Access.

You can also use the Secure Sockets Layer (SSL) protocol, a communication protocol that ensures privacy and integrity of data transmitted over the network, for DB-Access connections with IBM Informix. For information about using the SSL protocol, see the "Secure Sockets Layer Communication Protocol" section of the *IBM Informix Security Guide*.

Pre-DB-Access installation

Before you set up the DB-Access environment, you must perform the following preparatory steps:

1. Install the database server and set environment variables.
2. Set up the Global Language Support (GLS) locale, where language globalization requires it.
3. Start the database server.

You can then run the initialization script for the demonstration database (optional) and start the DB-Access program, as described in “Start DB-Access” on page 1-8.

Environment variables

As part of the installation and setup process, the system or database administrator sets certain environment variables that enable IBM Informix products to work within a particular operating-system environment. This section lists the environment variables that affect your ability to use DB-Access.

Important: You must have `$INFORMIXDIR/bin` in your path if you use DB-Access on a UNIX platform or `%INFORMIXDIR%\bin` in your path if you use DB-Access on a Windows platform. Your operating system uses the path to locate the initialization script and the **dbaccess** executable file.

In a UNIX environment, the database server must have the appropriate terminal set up from among those listed in the **`INFORMIXTERM`** environment variable.

DB-Access will use the terminal definitions in the `terminfo` directory unless the **`INFORMIXTERM`** environment variable is set to the `termcap` file. If DB-Access fails to initialize the menus based on the **`INFORMIXTERM`** setting, DB-Access will try to use the other setting. For example, if DB-Access fails to initialize the menus using the `terminfo` directory, DB-Access will initialize the menus using the `termcap` file.

You can set the following optional environment variables:

`DBACCNOIGN`
Rolls back an incomplete transaction if you execute the LOAD command in menu mode.

DBCENTURY
Lets you choose the appropriate expansion for DATE and DATETIME values that have only a two-digit year, such as 04/15/98.

DBDATE
Specifies the end-user formats of DATE values. See *IBM Informix Migration Guide* for more information about this variable

DBEDIT
Sets the default DB-Access text editor without changing the default text editor associated with the operating-system shell.

For more information about how DB-Access uses the text editor that you specify as default, see "Enter a new SQL statement" on page 3-3.

DBFLTMASK
Sets the default floating-point values of data types FLOAT, SMALLFLOAT, and DECIMAL within a 14-character buffer.

The effect of this variable is limited to the DB-Access display size for numbers.

DELIMIDENT
Causes the database server to interpret double quoted (") text as identifiers rather than strings.

IFX_LONGID
Determines whether a client application can handle long identifiers.

If you use the **IFX_LONGID** environment variable to support SQL identifiers with up to 128 bytes, some error, warning, or other messages of DB-Access might truncate database object names that include more than 18 bytes in their identifiers. You can avoid this truncation by not declaring names that have more than 18 bytes.

For more information about environment variables, see the *IBM Informix Guide to SQL: Reference*.

Create and work with the demonstration databases

You can practice using DB-Access with a demonstration database or with a database that you create. If you use an IBM Informix demonstration database, you can add, delete, or change the provided data and scripts, then re-initialize the database to its original condition. This publication provides examples of statements run against the demonstration data, with illustrations that show the results of these statements.

Demonstration databases

Demonstration databases are provided with your IBM Informix database server products.

You can configure the following demonstration databases:

- The **stores_demo** database illustrates a relational schema with tables about a fictitious wholesale sporting-goods distributor, as well as other tables that are used in examples. Tables containing electricity usage and geographical location data illustrate time series and spatial information. Many examples in IBM Informix manuals are based on the **stores_demo** database.
- The **superstores_demo** database illustrates an object-relational schema. The **superstores_demo** database contains examples of extended data types, type and table inheritance, and user-defined routines.

For descriptions of the databases and their contents, see the *IBM Informix Guide to SQL: Reference*.

The scripts that you use to install the demonstration databases are in the `$INFORMIXDIR/bin` directory on UNIX platforms and in the `%INFORMIXDIR%\bin` directory in Windows environments.

The DB-Access demonstration scripts are designed for the default locale. If you use a non-default locale, such as **en_us.utf8**, some features, such as the SET COLLATION statement, might not function correctly.

The following table lists the databases available for each database server. To set up or reinitialize the demonstration database, run the corresponding initialization script that the following table shows.

Table 1-1. Demonstration databases

Server type	Database name	Model	Initialization script
All Informix database servers	**stores_demo**	Relational database	**dbaccessdemo**
IBM Informix	**superstores_demo**	Object-relational database	**dbaccessdemo_ud**

Demonstration installation

When you run the installation script for a demonstration database, the script asks you if you would like to copy sample SQL command files. Command files that the demo includes have a `.sql` extension and contain sample SQL statements that you can use.

Always initialize or run DB-Access from the directory in which you want to store SQL command files for the following reasons:
- Each time you create or reinitialize the demonstration database, the installation script prompts you to ask if you want a copy of the sample command files provided with the demonstration database saved in the current directory.
- DB-Access lists only the files that end in the extension `.sql` in the current directory.

Command files are described in Appendix B, “Demonstration SQL,” on page B-1.

Preparing a demonstration database

To prepare a demonstration database:
1. Create a new directory.

   You must have UNIX read and execute permissions for each directory in the path name that you create.
2. Change directories to the new directory and run the initialization script.

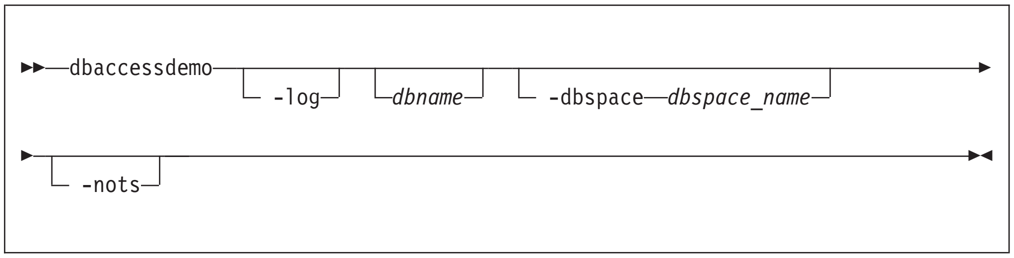
   Table 1-1 describes the various database models available for each database server version. For options that you can use with the initialization script, see “Command-line syntax” on page 1-7.
3. The initialization script displays a series of messages on the screen as the database is created. The final message of the script prompts you to make a choice. If you want to copy the command files into the directory that you created, press **Y**.

Tip: If you want to discard changes that you made to your database or to the command files, rerun the demonstration initialization script. When the script prompt message displays, press **Y** to replace the command files with the original versions.

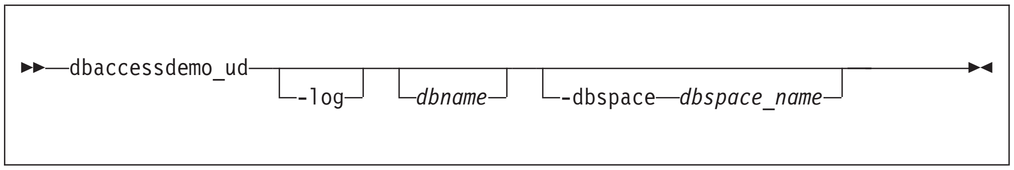
Command-line syntax

Use the command-line syntax to create the demonstration databases.

Syntax for stores_demo



Syntax for superstores_demo



-log Requests transaction logging for the demonstration database.

dbname
Substitutes for the default database name. For object-name guidelines, see the *IBM Informix Guide to SQL: Syntax*.

-dbspace
Requests a particular dbspace location for the demonstration database.

dbspace_name
Houses the demonstration database. If you do not specify a dbspace name, by default, the data for the database is put in the root dbspace. To create a dbspace, use the **onspaces** utility as described in your *IBM Informix Administrator's Guide*.

-nots Prevents the creation of the time series tables in the stores demo database. For more information on the time series tables, see the *IBM Informix Guide to SQL: Reference*.

Examples

- The following command creates a database named **stores_demo**:
```
dbaccessdemo
```
- The following example creates an instance of the **stores_demo** database named **demo_db**:
```
dbaccessdemo demo_db
```
- The following command initializes the **stores_dem** database and also initiates log transactions:
```
dbaccessdemo -log
```

- The following command creates an instance of the **stores_demo** database named **demo_db** in **dbspace_2**:

```
dbaccessdemo demo_db -dbspace dbspace_2
```

Privileges for the demonstration database

When you run the initialization script to create an instance of the demonstration database under your login, the database server recognizes your login as the owner and database administrator (DBA) of that database. As DBA, you automatically acquire some exclusive privileges over objects in your database. You can extend those privileges to others with the GRANT statement, which is described in the *IBM Informix Guide to SQL: Syntax*.

Permissions for the SQL command files

Some operating systems require that you have execute permissions to run SQL command files, read permissions to open these files or their contents in DB-Access, or write permissions to save modified or new files.

Use the UNIX **chmod** command to enable execution of the SQL files that the initialization script installed.

Start DB-Access

For more information about how to start DB-Access, see the following references:

- To show all the menus, start DB-Access at its main menu. See “Display the main menu” on page 1-9.
- To start and exit from a specific DB-Access menu or screen, see “Display other menus or options” on page 1-9.
- To open a file that contains SQL statements without showing the DB-Access menus, see “Execute a command file” on page 1-13.
- To type DB-Access options at the command line, without the full-screen menu interface, see “DB-Access interactively in non-menu mode” on page 1-14.
- To check the DB-Access version or transfer nonprintable characters in hexadecimal form, see “Activate the XLUF feature for nonprintable characters” on page 1-14.

On Windows, you can set up the DB-Access program icon to perform any of the commands that this chapter shows.

If the **TERM**, **TERMCAP**, or **TERMINFO** environment variables on UNIX do not enable DB-Access to recognize the type of terminal you use, the main menu does not show. Instead, a message similar to the following text is displayed:

```
Unknown terminal type.
```

If you use a Windows terminal to run DB-Access on a UNIX database server, the terminal-emulation window must emulate a terminal type that DB-Access can recognize, or the database server shows an unknown terminal-type message in the terminal-emulation window.

For more information about **INFORMIXTERM** and the appropriate terminal setup, see your *IBM Informix Installation Guide* or the *IBM Informix Guide to SQL: Reference*.

Tip: If your operating system cannot find **dbaccess**, place the full path before the program name, as follows:

```
$INFORMIXDIR/bin/dbaccess
```

Start the DB-Access menu interface

To familiarize yourself with the DB-Access menu structure, see Figure 1-1 on page 1-3.

Display the main menu

Typically, you start DB-Access with the main menu as the top-level menu from which you reach submenus and their options.

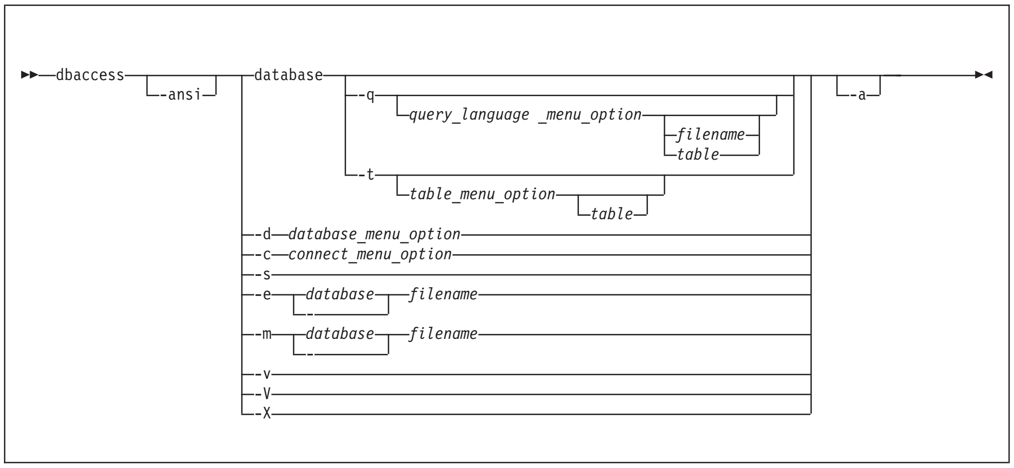
If you use a window interface, such as OpenWindows, on a UNIX terminal, issue the **dbaccess** command from a nonscrolling console window.

If your path includes `$INFORMIXDIR/bin`, the **dbaccess** command is the simplest way to start DB-Access.

Without arguments, the single word **dbaccess** starts the main menu with no database selected and no options activated. You can then select submenus from the main menu.

Display other menus or options

You can specify the options shown in the following diagram to access menus directly.



-ansi Causes DB-Access to generate a warning whenever it encounters an IBM Informix extension to ANSI-compliant syntax. For more information, see "Check for ANSI compliance" on page 1-13.

-a Stops a process directly after the first error is encountered. Stopping a process from continuing after the first error can ensure greater database consistency.

-c Starts with the CONNECTION menu as the top-level menu.

-d Starts with the DATABASE menu as the top-level menu.

-e Echoes each line from a command file designated by *filename*.

-m Displays all error messages generated by multiple levels of the server that pertain to an SQL statement in command files.

-q Starts at the query-language menu (SQL-menu) as the top-level menu.

-s Connects you to the main DB-Access menu and displays information about the current session.

This information includes database server name, database server type, the host computer, server capabilities, and other settings.

-t Starts at the TABLE menu as the top-level menu.

-V Displays the version number and serial number for DB-Access without launching the application. You cannot use any other options with **-V**.

-version
Displays the version number and build information for DB-Access, including the GLS library version, without launching the application. You cannot use any other options with **-version**.

-X Activates the hexadecimal format for LOAD and UNLOAD statements. The **-X** option can be used after the **-ansi** option and before any of the other options.

-filename
Indicates that you are not specifying a database in the command line because one is specified in a DATABASE statement in the command file.

database
Name of the database that you want DB-Access to connect to at the startup of your current session.

filename
Names a command file to load with the SQL menu.

table Specifies a table in the database.

connect_menu_option
Option or suboption on the connect menu. See "CONNECTION menu options."

database_menu_option
Option or suboption on the database menu. See "DATABASE menu options" on page 1-11.

query_language_menu_option
Option or suboption on the query-language menu. See "QUERY-LANGUAGE menu options" on page 1-11.

table_menu_option
Option or suboption on the table menu. See "TABLE menu options" on page 1-12.

If you exit from a submenu or option that you specified from the command line, you will exit directly to the operating-system command line.

Menu suboptions

The following menu suboptions allow you to access submenus directly.

CONNECTION menu options:

-cc Chooses the Connect option on the CONNECTION menu.

-cd Chooses the Disconnect option on the CONNECTION menu.

DATABASE menu options:

-dc Chooses the Create option on the DATABASE menu.

- **-dcl** Takes you to the LOG option on the CREATE DATABASE menu

-dd Chooses the Drop option on the DATABASE menu.

-di Chooses the Info option on the DATABASE menu. With this option, you can add another letter as follows to go to the next menu level and view:

- **-dib** The dbspaces information for the current database (Informix OnLine only)
- **-din** The NLS information for the current database
- **-dip** Stored procedures in the current database

If you do not include a database name before any **-di** option, you must choose a current database from the SELECT DATABASE screen.

-dl Chooses the CLose option on the DATABASE menu.

-ds Chooses the Select option on the DATABASE menu.

QUERY-LANGUAGE menu options:

-qc Chooses the Choose option on the SQL menu.

-qd Chooses the Drop option on the SQL menu.

-qi Chooses the Info option on the SQL menu. With this option, you can add another letter as shown in the following list (and specify a table) to go to the next menu level and view:

- **-qic** Columns in the table
- **-qif** Information about fragmentation strategy for the table
- **-qig** Information about triggers in the table
- **-qii** Indexes on the table
- **-qio** Table constraints
- **-qip** Access privileges on the table
- **-qir** Table-level references privilege on the table
- **-qis** Table status information

If you do not include a table name with the **-qi** option, you must choose one from the INFO FOR TABLE screen.

-qm Chooses the Modify option on the SQL menu.

-qn Chooses the New option on the SQL menu.

-qs Chooses the Save option on the SQL menu.

-qu Chooses the Use-editor option on the SQL menu.

If you do not include a database name before a **-q** option, you must choose a current database from the SELECT DATABASE screen.

When you select the Modify option on the QUERY-LANGUAGE menu, you must first select a command file to modify from the CHOOSE menu. The MODIFY screen is then displayed and shows the text.

Restriction: You cannot go directly to the Run or Output option on the SQL menu. Trying to do so results in an error message.

TABLE menu options:

-ta	Chooses the Alter option on the TABLE menu.
-tc	Chooses the Create option on the TABLE menu.
-td	Chooses the Drop option on the TABLE menu.
-ti	Chooses the Info option on the TABLE menu. With this option, you can add another letter as shown in the following list (and specify a table) to go to the next menu level and view:

-tic	Columns in the table
-tif	Information about fragmentation strategy for the table
-tig	Information about triggers in the table
-tii	Indexes on the table
-tio	Table constraints
-tip	Access privileges on the table
-tir	Table-level references privilege on the table
-tis	Table status information

If you do not include a table name with the **-ti** option, you must choose one from the INFO FOR TABLE screen.

If you do not include a database name before a **-t** option, you must choose a current database from the SELECT DATABASE screen.

Examples of command-line syntax

Assume that the database server that you have online contains a database named **mystores**. To make the **mystores** database the current database, start DB-Access with the following command:

```
dbaccess mystores
```

You can specify a database on a database server that is not online. For example, either of the following commands selects the **newstores** database on the **xyz** database server:

```
dbaccess newstores@xyz
dbaccess //xyz/newstores
```

When DB-Access starts, the database and database server name that you specify are displayed on the dashed line, as the following figure shows.

```
DB-Access:   Query-language  Connection  Database  Table  Session  Exit


---------------- newstores@xyz ---------------------Press CTRL-W for Help ---
```

Figure 1-2. The DB-Access main menu with database and database server name

Execute a command file

When you start DB-Access from the command line, you can specify a database as current, execute a file that contains one or more SQL statements, and display multilevel error messages.

The following sample command executes the SQL statements in a file named `sel_stock.sql` on the **mystores** database:

```
dbaccess mystores sel_stock
```

The following sample command executes the SQL statements in the `sel_all.sql` file on the database that file specifies:

```
dbaccess - sel_all.sql
```

View the information Schema

Use DB-Access to execute the `xpg4_is.sql` file in the `$INFORMIXDIR/etc` directory. This SQL file creates the Information Schema and installs the views for a specified database. The following command creates the Information Schema for database **mystores**:

```
dbaccess mystores $INFORMIXDIR/etc/xpg4_is.sql
```

The Information Schema adds to the database four information-only views that conform to X/Open XPG4 with IBM Informix extensions. After you run `xpg4_is.sql`, use DB-Access to retrieve information about the tables and columns that you have access to in the specified database. For more information about Information Schema views, see the *IBM Informix Guide to SQL: Reference*.

Tip: It is not recommended that you install these XPG4-compliant views on an ANSI database, because their format differs considerably from that of the ANSI-compliant Information Schema views that the SQL standards committee has defined.

Check for ANSI compliance

To check your SQL statements for compliance with ANSI standards, include the **-ansi** option or set the **DBANSIWARN** environment variable. Use the **-ansi** option with other **dbaccess** options such as **-dc** (to create a database), **-tc** or **-ta** (to create or alter a table), or **-qc** ***filename*** (to choose a command file). The following command checks for ANSI compliance while DB-Access creates the database **research**:

```
dbaccess -ansi -dc research
```

You do not need to specify the **-ansi** option on the command line if the **DBANSIWARN** environment variable is set.

DB-Access displays the SQLSTATE value with the warning under the following circumstances:

- You include the **-ansi** option or set the **DBANSIWARN** environment variable.
- You access or create an ANSI database.
- You run DB-Access in line mode or specify a `.sql` input file.
- Execution of a SQL statement generates a warning rather than an error.

For more information about ANSI-compliant databases, see the *IBM Informix Guide to SQL: Reference* and the *IBM Informix Guide to SQL: Tutorial*. For more information about SQLSTATE values, see the GET DIAGNOSTICS statement in the *IBM Informix Guide to SQL: Syntax*. The *IBM Informix Guide to SQL: Syntax* also provides information about ANSI compliance in IBM Informix SQL syntax.

Check the scale of DECIMAL(p) values

In an ANSI-compliant database, columns declared as a DECIMAL(p) data type have a precision of **p** and a scale of zero, so that only integer values are stored. In a database that is not ANSI-compliant, DECIMAL(p) is a floating-point data type whose scale is large enough to store the exponential notation for a value.

For example, the following calculation shows how many bytes a DECIMAL(5) column requires in the default locale (where the decimal point occupies a single byte):

```
1 byte for the sign of the data value
1 byte for the 1st digit
1 byte for the decimal point
4 bytes for the rest of the digits (precision of 5 - 1)
1 byte for the e symbol
1 byte for the sign of the exponent
3 bytes for the exponent
-------------------------------------------------
12 bytes total
```

Thus, "12345" in a DECIMAL(5) column is displayed as "12345.00000" (that is, with a scale of 6) in a database that is not ANSI-compliant.

Activate the XLUF feature for nonprintable characters

You can use the **-X** option to activate the hexadecimal load and unload format (XLUF) feature in DB-Access at the command line. When you start DB-Access with the **-X** option, the LOAD and UNLOAD SQL statements can format nonprintable ASCII signs in hexadecimal format. The **-X** option can be used after the **-ansi** option and before any of the other options. A `.unl` file that the UNLOAD statement produces contains the hexadecimal format changes.

Important: The `.unl` files that contain data in a hexadecimal format are not compatible with IBM Informix database servers before Version 6.0. However, `.unl` files generated without the XLUF functionality are fully compatible with Version 6.0 or later database servers.

For more information, see the descriptions of the LOAD and UNLOAD statements in the *IBM Informix Guide to SQL: Syntax*. Also see the discussion of the various SQL utilities in the *IBM Informix Migration Guide* and the data types information in the *IBM Informix Guide to SQL: Reference*.

DB-Access interactively in non-menu mode

If you do not want to use the menus and do not have a prepared SQL file, use your keyboard or standard input device to enter SQL statements.

Read from the keyboard or standard input device

When you start DB-Access without a menu argument (such as **-q**) and with a hyphen as the final argument, DB-Access processes commands from the standard input device (on UNIX platforms) or the keyboard (on Windows platforms). DB-Access reads what you type until you indicate that the input is completed.

Then DB-Access processes your input and writes the results to the standard output device (on UNIX platforms), or the command window (on Windows).

Interactive input: DB-Access reads and executes SQL statements from the terminal keyboard interactively. While DB-Access runs interactively, the greater than (>) prompt marks the line where you type your next SQL statement.

When you type a semicolon (;) to end a single SQL statement, DB-Access processes that statement. When you press **CTRL-D** to end the interactive session, DB-Access stops running. The following example shows user input and results in an interactive session:

```
dbaccess - -
>database stores_demo;

Database selected.

>select count(*) from systables;

(count(*))

     21

1 row(s) retrieved.

>^D

dbaccess - -
>database stores_demo;

Database selected.

>select count(*) from systables;

(count(*))

     21

1 row(s) retrieved.

>^D
```

Batch command input on UNIX platforms: You can use an inline shell script to supply one or more SQL statements. For example, you can use the UNIX C, Bourne, or Korn shell with inline standard input files:

```
dbaccess mystores- <<EOT!
select avg(customer_num) from customer
where fname matches '[A-G]*';
EOT!
```

You can use a pipe to supply SQL statements, as in this UNIX example:

```
echo 'select count(*) from systables' | dbaccess mystores
```

DB-Access interprets any line that begins with an exclamation mark (!) as a shell command. You can mix shell escape lines with SQL statements and put them in SQL statements, as follows:

```
dbaccess mystores -
>select
!echo hello
>hello
count(*) from systables;
>
(count(*))
```

```
     21

1 row(s) retrieved.
>
```

Connect to a database environment in non-menu mode

You can use the CONNECT . . . USER syntax in SQL statements that you issue in interactive mode. However, DB-Access does not support the USER clause of the CONNECT statement when you connect to a default database server.

Connect in interactive non-menu mode: When you include the USER '*user identifier*' clause in a CONNECT statement in interactive mode, DB-Access prompts you to enter a password.

The following two command examples show how to connect to a database server in interactive mode. The first example uses the CONNECT statement without specifying a user identifier.

```
dbaccess -nohistory- -

> connect to '@starfish';

Connected.
```

If you include the USER clause in a CONNECT statement, as the second example shows, DB-Access uses echo suppression to prompt you for a password:

```
> connect to '@starfish' user 'marae';

ENTER PASSWORD:

Connected.
```

Restriction: For security reasons, do not enter the password on the screen where it can be seen. Also, do not include the USING *password* clause in a CONNECT statement when you use DB-Access interactively. If you are in interactive mode and attempt to enter a password before the prompt, an error message is displayed.

Connect with a file or shell file in background mode: You can execute the USER clause of a CONNECT statement in a DB-Access file that includes the USER clause. The following example uses a command file that contains a CONNECT statement with a USING clause to connect to a database server:

```
dbaccess - connfile.sql
```

Important: An SQL command file that contains the statement

```
CONNECT TO '@dbserver' USER 'user_id' USING password
```

should be protected from access by anyone other than the **user_id** that the USER clause identifies.

For UNIX, the following example uses a shell file to connect to a database server. DB-Access prompts you for a password.

```
dbaccess - - <<\!
connect to '@starfish' user 'marae';
!

ENTER PASSWORD:
```

Here the delimiting quotation marks preserve letter case in the database server name and in the authorization identifier of the user.

Chapter 2. The full-screen menu interface

These topics provide introductory information.

If you are familiar with DB-Access, you might prefer to skip these introductory topics.

The DB-Access user interface

The DB-Access user interface combines the following features:

- A hierarchy of menus
- Screens that prompt you for brief responses and choices from selection lists
- Contextual HELP screens
- The interactive Schema Editor that helps you structure tables
- An SQL programmer environment, which includes the following features:
  - The built-in SQL editor where you enter and modify SQL and SPL statements
  - An option to use another editor of your choice
  - The database server syntax checker and runtime debugger
  - Storage, retrieval, and execution of SQL and SPL routines
- A choice of output for database queries and reports

Nonprogrammers build their databases with the interactive Schema Editor described in Chapter 5, "The Table option," on page 5-1 Programmers use the SQL programmer environment described in Chapter 3, "The Query-language option," on page 3-1.

The keyboard with DB-Access

The terminal keyboard has special keys that you use to instruct DB-Access. Before you begin to use DB-Access, locate the following keys:

Arrows

The arrow keys are usually found at the lower right of your keyboard.

The up arrow [↑] key moves the cursor up one line. If your terminal does not have a [↑] key, type **CTRL-K**.

The down arrow [↓] key moves the cursor down one line. If your terminal does not have a [↓] key, type **CTRL-J**.

The left arrow [←] key moves the cursor back one position on the screen. If your terminal does not have a [←] key, type **CTRL-H**.

The right arrow [→] key moves the cursor forward one position on the screen. If your terminal does not have a [→] key, use the **CTRL-L** key.

BACKSPACE

The **BACKSPACE** key is usually located at the upper right of the keyboard and might be marked with a left arrow.

CONTROL

The **CONTROL** key is often labeled **CTRL** or **CNTRL** and is usually located on the left side of the keyboard. This manual refers to the **CONTROL** key as **CTRL**. On some systems, type **CTRL-C** to cancel or stop a program or to leave a current menu and return to the menu one level above. This manual refers to **CTRL-C** as the Interrupt key.

DELETE
The **DELETE** key is sometimes labeled **RUBOUT**, **CANCEL**, or **DEL**.

On most systems, use the **DELETE** key to cancel or stop a program or to leave a current menu and return to the menu one level above. This manual refers to the **DELETE** key as the **DEL** key or the Interrupt key.

ESCAPE
The **ESCAPE** key is usually located on the upper left corner of your keyboard and might be labeled **ESC**.

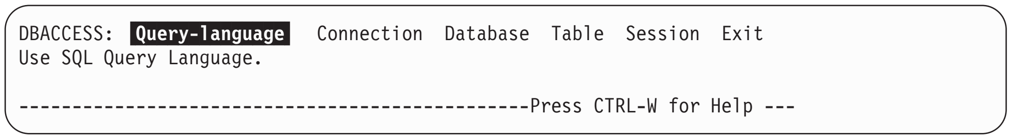
RETURN
The **RETURN** key is located on the right side of the keyboard. It might be labeled **RETURN** or **NEWLINE** or it might be indicated with a bent left arrow.

SPACEBAR
The **SPACEBAR** is generally unlabeled.

Interrupt
Issue the command **stty -a** on the command line and check the "intr" setting to see if your terminal recognizes **CTRL-C** or **DELETE** (described earlier) or another key as the Interrupt key.

Menus

Menus are shown at the top of each screen. Generally, the DB-Access main menu, as the following figure shows, is the top level of the menu hierarchy. You can select a submenu as the top level if you start DB-Access with a submenu option, such as those described in "Execute a command file" on page 1-13.



```
DBACCESS:  Query-language   Connection  Database  Table  Session  Exit
Use SQL Query Language.

-----------------------------------------------Press CTRL-W for Help ---
```

Figure 2-1. The DB-Access main menu

A message below the option menu briefly describes the function of the highlighted option. If you highlight another option, the description changes. To find out what an option does, highlight it and read the description.

The dashed line at the bottom of the header shows the name of the current database, if one is selected, and a reminder to press **CTRL-W** for Help.

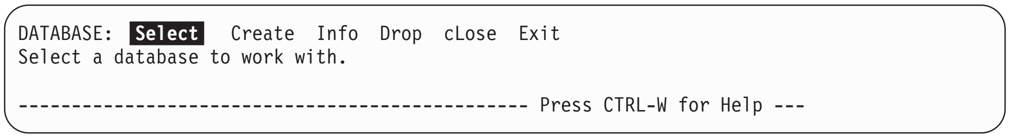
Choose a menu option

In Figure 2-1, the box, or highlight, indicates that Query-language is the current option. Press **RETURN** to choose the highlighted option or choose another option as follows:

- Use the **SPACEBAR** or the left and right arrow keys to move the highlight. Options are arranged in a ring. If you move the highlight to the right, past the end of the list, the first option becomes current again.

  Press **RETURN** to choose the highlighted option.
- Type the shortcut letter shown in uppercase in the name of the option that you want to select. (You can type the shortcut letter in either uppercase or lowercase.)

Because some menus have multiple options that start with the same letter, the shortcut key is not always the first letter of an option name. For example, in the following figure, the DATABASE menu, both `Create` and `cLose` start with the letter `c`. As indicated by the uppercase shortcut keys, press the **C** key to select the Create option and press the **L** key to select the cLose option.



```
DATABASE:  Select   Create  Info  Drop  cLose  Exit
Select a database to work with.

------------------------------------------------ Press CTRL-W for Help ---
```

Figure 2-2. A sample menu screen

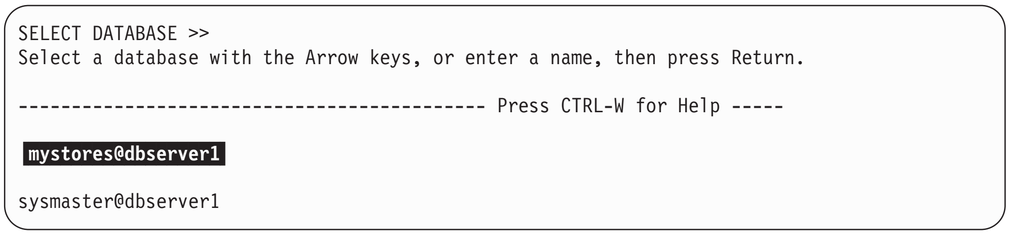
DB-Access displays the screen for the menu option that you select.

Exit a menu screen

If the menu has an Exit option, press the **E** key to exit the menu. If no Exit option exists, use the Interrupt key (usually **DEL** or **CTRL-C**) to exit the menu. You then return to the menu, screen, or command line from which you selected the exit menu.

The response screens

Some screens, such as the SELECT DATABASE screen in the following figure, prompt you for a name or value.



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

-------------------------------------------- Press CTRL-W for Help -----

 mystores@dbserver1

sysmaster@dbserver1
```

Figure 2-3. A sample response screen

If Global Language Support (GLS) is enabled, the listed items are sorted according to the code-set collation order of the current locale.

Read the screen header

The top line of a response screen shows a prompt as the screen name followed by double angle brackets (>>) and the cursor. A message below the prompt gives brief instructions, such as Press **CTRL-W** for Help. The dashed line identifies the database that you select as current.

Select or enter values on the screen

Where you enter your response depends on the operation, as the following two choices show:

- You might type a value in the header of an entry screen, after the double angle brackets (>>) at the top of the screen. For example, you can type a valid database name after SELECT DATABASE>>, as Figure 2-3 shows.

  Press **RETURN** when you finish typing, and DB-Access displays the next screen or takes other appropriate action.
- Some screens have a list on the lower part of the screen from which you can choose an item instead of typing your selection in the header. For example, the SELECT DATABASE screen in Figure 2-3 lists the databases available on the current database server.

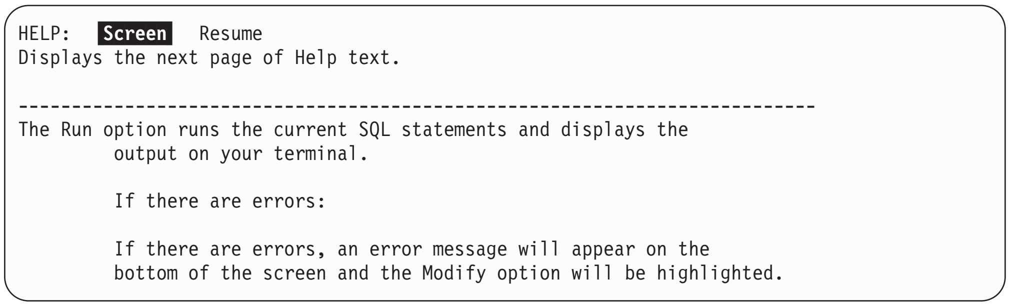
Use the arrow keys to highlight the item that you want and then press **RETURN**. DB-Access displays the next screen or takes other appropriate action.

Exit a response screen

Without a menu, a screen cannot have an Exit option. Press the Interrupt key (usually **DEL** or **CTRL-C**) to return to the previous menu or screen.

The HELP screen

Press **CTRL-W** to display a HELP screen that provides information about the current menu option or screen function. The following figure shows some of the text that is displayed when you press **CTRL-W** for information about the Run option on the SQL menu.



```
HELP:   Screen   Resume
Displays the next page of Help text.

---------------------------------------------------------------------------
The Run option runs the current SQL statements and displays the
         output on your terminal.

         If there are errors:

         If there are errors, an error message will appear on the
         bottom of the screen and the Modify option will be highlighted.
```

Figure 2-4. A partial HELP screen with text

If the Help text is longer than one page, the Screen option is highlighted. Press **RETURN** to view the next screen. To select the Resume option, highlight it with the **SPACEBAR** or the right arrow key and then press **RETURN** or the **R** key.

If the Help text is only one page, the Resume option is highlighted, and you need only press **RETURN**.

For an illustration of how to read the syntax diagrams that are displayed when you request online Help for creating, modifying, or editing an SQL statement, see Appendix A, "How to read online help for SQL statements," on page A-1.

An editor screen

You must use a text editor to prepare and modify SQL statements and command files when you select the Query-language option on the main menu. DB-Access provides two ways to edit the text of SQL statements and command files before you run them: the *SQL editor* and a *system editor*. Because you only use these screens with the Query-language option, see Chapter 3, "The Query-language option," on page 3-1 for detailed instructions.

Alternative approaches

These topics illustrate some common database tasks and list alternative procedures to use them. These topics can help you determine your preferred method of using the DB-Access interface.

Database-level tasks

The database you use is called the *current* database. To select an existing database as current, you can:

- Use the following command-line syntax:

  ```
  dbaccess databasename
  ```

  For more information about using DB-Access from the command line, see "Start DB-Access" on page 1-8.
- Use any method from within DB-Access that is described in the following table.

Menu screen	Option or SQL statement	Action	Reference
CONNECTION	Connect	Prompts for database server, user name, password, and database name. Connects you according to the choices you make.	Chapter 6, "The Connection and Session options," on page 6-1
SQL	CONNECT	Connects to a database. If you use a fully-qualified database name, you simultaneously connect to the database server.	Chapter 3, "The Query-language option," on page 3-1
DATABASE	Select	Chooses a current database from a list of databases available on the current database server.	Chapter 4, "The Database option," on page 4-1
SQL	DATABASE	Chooses a database as current.	Chapter 3, "The Query-language option," on page 3-1

To create your own database, use one of the following methods.

Menu screen	Option or SQL statement	Action	Reference
DATABASE	Create	Prompts for a database name, dbspace, and log options and then creates the new database.	Chapter 4, "The Database option," on page 4-1
SQL	CREATE DATABASE	Creates the database you name in the statement with the appropriate setup for the keywords you use.	Chapter 3, "The Query-language option," on page 3-1

To drop a database, use one of the following methods.

Menu screen	Option or SQL statement	Action	Reference
DATABASE	Drop	Lists available databases and drops the database you choose from the list.	Chapter 4, "The Database option," on page 4-1
SQL	DROP DATABASE	Drops the database named in the command.	Chapter 3, "The Query-language option," on page 3-1

To close a database, use one of the following methods.

Menu screen	Option or SQL statement	Action	Reference
DATABASE	cLose	Closes the current database.	Chapter 4, “The Database option,” on page 4-1
SQL	CLOSE DATABASE	Closes the current database.	Chapter 3, “The Query-language option,” on page 3-1
CONNECTION	Disconnect	Closes the current database and disconnect from a database server.	Chapter 6, “The Connection and Session options,” on page 6-1
SQL	DISCONNECT CURRENT	Closes the current database and disconnect from the current database server.	Chapter 3, “The Query-language option,” on page 3-1

To display information about a database, such as the dbspaces that contain it, choose the Info option on the DATABASE menu.

Table-level tasks

To create a table, allocate storage, or apply fragmentation, use one of the following methods.

Menu screen	Option or SQL statement	Action	Reference
TABLE	Create	Uses the Schema Editor.	Chapter 5, “The Table option,” on page 5-1
SQL	CREATE TABLE	Uses the SQL editor or system editor and SQL statements.	Chapter 3, “The Query-language option,” on page 3-1

To make changes to the structure of a table, use one of the following methods.

Menu screen	Option or SQL statement	Action	Reference
TABLE	Alter	Guides you, with menus, through the available choices for changing an existing table.	Chapter 5, “The Table option,” on page 5-1

Menu screen	Option or SQL statement	Action	Reference
CREATE TABLE	Modify	Enables you to change a schema before you build the table.	Chapter 5, "The Table option," on page 5-1
SQL	ALTER TABLE	Changes an existing table according to the keywords you include with the statement.	Chapter 3, "The Query-language option," on page 3-1
SQL	ALTER FRAGMENT	• Changes an existing fragmentation strategy (for a table or index). • Creates the table fragments.	Chapter 3, "The Query-language option," on page 3-1

To drop a table from a database, use one of the following methods.

Menu screen	Option or SQL statement	Action	Reference
TABLE	Drop	Drops the current table.	Chapter 5, "The Table option," on page 5-1
SQL	DROP TABLE	Drops the table named in the command.	Chapter 3, "The Query-language option," on page 3-1

To move a table from current database to another database, use one of the following methods.

Menu screen	Option or SQL statement	Action	Reference
TABLE	Move	Guides you, with menus, through available choices for moving a table from the current database to another database.	Chapter 5, "The Table option," on page 5-1

Menu screen	Option or SQL statement	Action	Reference
SQL	Move Table	Moves the table named in the command.	Chapter 3, "The Query-language option," on page 3-1

To display information about the structure and characteristics of a table, use one of the following methods.

Menu screen	Option or SQL statement	Action	Reference
TABLE	Info	Provides a menu of available table information categories.	Chapter 5, "The Table option," on page 5-1
SQL	Info (option)	Prompts you to select from a list of available tables and then provides the same menu as the Info option on the TABLE screen.	Chapter 3, "The Query-language option," on page 3-1
SQL	INFO (SQL statement)	Provides the information specified in the keywords you include with the INFO statement.	Chapter 3, "The Query-language option," on page 3-1

To display the data stored in a table, run a SELECT statement from the SQL editor. Use the procedures described in Chapter 3, "The Query-language option," on page 3-1.

Tip: Several command files are included with DB-Access that contain sample practice SELECT statements. Appendix B, "Demonstration SQL," on page B-1 lists the demonstration files that are supplied with the **stores_demo** database.

Chapter 3. The Query-language option

These topics describe how to use the Query-language option on the DB-Access main menu. When you select the Query-language option, the SQL menu is displayed. Use the various SQL menu options to enter, modify, save, retrieve, and run SQL statements.

Use the Query-language option to:

- Learn SQL and SPL.

  For example, use the Query-language option to practice the examples in the *IBM Informix Guide to SQL: Tutorial*.
- Create and alter table structures as an alternative to the DB-Access Schema Editor.
- Select, display, add, update, and delete data.

  DB-Access has no menu options for data manipulation.

These topics also describe support for SPL routines.

Overview of the SQL menu

As you use the various SQL menu options, DB-Access retains the statements, if any, in the editor. These statements are called the *current* statements.

Use the following steps to create and test SQL:

- To enter a new SQL statement or to enter multiple statements separated by semicolons, use the New option.

  If you prefer to enter or modify a query with an editor of your choice, use the Use-editor option. This option provides an alternative to the New and Modify options.
- To execute your statements, use the Run option.

  When you choose the Run option, a message is displayed or the data retrieved by a query is displayed with the number of rows retrieved.

  To send the query results to an output file or printer instead of your terminal, use the Output option.
- If a syntax error occurs when you run the query, or if you want to change the query, use the Modify or Use-editor option to revise the text of the query.

  To reexecute the query, use the Run option.
- To save the text of the query in a command file, use the Save option.

  To run or modify a query that you saved in a command file, use the Choose option to select the file.
- To delete a command file in which you saved a query, use the Drop option.

A system editor

When you want to enter or modify a long SQL statement or series of statements, you might prefer the flexibility and familiarity of a system editor to the SQL editor. Select the Use-editor option from the SQL menu to use the system editor.

If you have not set the **DBEDIT** environment variable, you must select a text editor to use for the session. If you select Use-editor, DB-Access prompts you to accept or override the default system editor once each session, as the following figure shows.

```
USE-EDITOR >>vi
Enter editor name. (RETURN only for default editor)

--------------- mystores@dbserver1 ------------ Press CTRL-W for Help -----
```

Figure 3-1. Sample system editor screen for entering and modifying SQL statements

The default editor that DB-Access displays depends on the preference that you establish for your operating system:

- Common UNIX system editors are **vi** and **ex**.
- If you use a text editor as the system default, you must save the `.sql` files as text.

Press **RETURN** to select the default editor you named after the USE-EDITOR prompt. To use a different editor, type the name of that editor and press **RETURN**.

The Query-language option

To select SQL, choose the Query-language option from the main menu. Press the **Q** key or highlight the Query-language option and press **RETURN**. The Query-language option opens the SQL menu, as the following figure shows.

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

-----------------------------------------------Press CTRL-W for Help -----
```

Figure 3-2. The SQL menu

If you select the Query-language option on the main menu and have not selected a database, the SELECT DATABASE screen opens. Specify a database at the prompt or press the Interrupt key to display the SQL menu. From the SQL menu, create or choose and run an SQL statement to specify the current database.

Important: In the SELECT DATABASE screen, the names of databases are limited to 18 characters. If a database name is longer than 18 characters, you will see the first 17 characters of the name followed by a '+' sign. Enter a '+' sign to display the complete long name in **vi**. To exit from **vi**, press **ESC ZZ**.

The SQL menu has the following options.

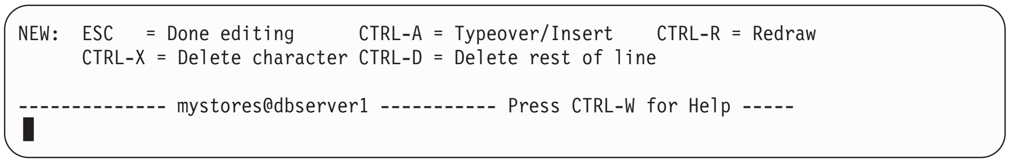
Option	Purpose	Instructions
New	Clears current statements and positions cursor in SQL editor.	"Enter a new SQL statement" on page 3-3
Run	Executes current SQL statements.	"Run an SQL statement" on page 3-5
Modify	Allows you to modify current SQL statements in SQL editor.	"Modify an SQL statement" on page 3-10
Use-editor	Starts a system editor so that you can modify current statements or create new statements. Use-editor is interchangeable with New and Modify.	"A system editor" on page 3-1

Option	Purpose	Instructions
Output	Redirects Run-option output to a file, printer, or system pipe.	"Redirect query results" on page 3-11
Choose	Lists SQL command files so that you can choose a file to execute or modify.	"Choose an existing SQL statement" on page 3-13
Save	Saves current SQL statements in a file for later use.	"Save the current SQL statement" on page 3-14
Info	Shows table information, such as columns, indexes, privileges, constraints, triggers, status, and fragmentation strategy.	"Display table information" on page 3-15
Drop	Deletes a specified SQL command file.	"Drop an SQL statement" on page 3-16
Exit	Returns to main menu.	none

Enter a new SQL statement

To enter an SQL statement, perform one of the following actions:

- Select the New option on the SQL menu. The NEW screen opens and indicates that you are using the SQL editor, as the following figure shows.



```
NEW:  ESC   = Done editing      CTRL-A = Typeover/Insert    CTRL-R = Redraw
      CTRL-X = Delete character CTRL-D = Delete rest of line

------------- mystores@dbserver1 ----------- Press CTRL-W for Help -----
█
```

Figure 3-3. The NEW screen for entering new SQL statements

- If you prefer to work with a different editor from the one that is installed on your system, select the Use-editor option. This option is described in "A system editor" on page 3-1.

As the preceding figure shows, the NEW screen starts with the cursor moved below the header, which indicates where you enter text on the screen. Use the editor to enter statements and edit them before you run them. To string several SQL statements together, separate them with a semicolon.

The editing keys

The editing keys listed at the top of the SQL editor screen perform the following special functions when you enter SQL statements:

CTRL-A
: Switches between *insert* mode and *typeover* mode. You are automatically in typeover mode when you begin to use the SQL editor.
  - In insert mode, the text beneath the cursor shifts to the right as you enter new characters.
  - In typeover mode, characters you enter replace the text beneath the cursor.

CTRL-D
: Deletes characters from the current cursor position through the end of the line.

CTRL-R
Redraws the current screen. Use it when you receive an electronic message or some other interference that makes it difficult to read the SQL statement text that you enter.

CTRL-X
Deletes a character beneath the cursor.

ESC Returns you to the SQL menu when you finish entering or editing the SQL statement. You can then run or modify the statement or select the system editor for more extensive editing.

For more information about arrow and cursor-positioning keys, see "The keyboard with DB-Access" on page 2-1.

Editing restrictions

The SQL editor does not display more than 80 characters on a line and does not wrap lines, as the following two such restrictions show:

- If you choose an existing command file in which the characters in a line extend beyond the 80th column, DB-Access displays a percent sign (%) in the 80th column to indicate an overflow. You cannot see all the characters beyond the percent sign, but the statement should run correctly.
- If you type characters in a new command file so that a line extends beyond the 80th column, DB-Access overwrites all the characters in the 80th column. You cannot see the overflow, and the statement does not run correctly.

To make the full text show on the screen, press **RETURN** at a logical place in the first 80 characters of each line. If you must type a quoted character string that exceeds 80 characters, such as an insert into a long CHAR column, use a system editor instead of the SQL editor.

If you want to include comments in the text:

- Use double minus signs for ANSI-compliant databases.
- Preface each comment line with a double minus sign (--) comment indicator. The comment indicator spans the entire line.
- Use braces ({ }) for databases that are not ANSI-compliant. Enclose the entire comment indicator between the braces.

When you use the SQL editor, you can type as many lines of text as you need. You are not limited by the size of the screen, although you might be limited by the memory constraints of your system or the maximum SQL statement size of 64 KB.

If you insert more lines than one screen can hold, the SQL editor scrolls down the page with the additional text. The beginning and ending line numbers of the current page are displayed on the fourth line of the text-entry screen, as the following figure shows.

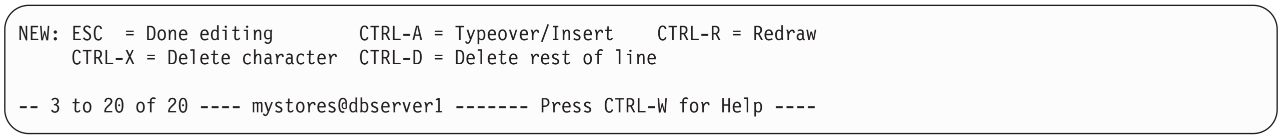

```
NEW: ESC  = Done editing        CTRL-A = Typeover/Insert    CTRL-R = Redraw
     CTRL-X = Delete character  CTRL-D = Delete rest of line

-- 3 to 20 of 20 ---- mystores@dbserver1 ------- Press CTRL-W for Help ----
```

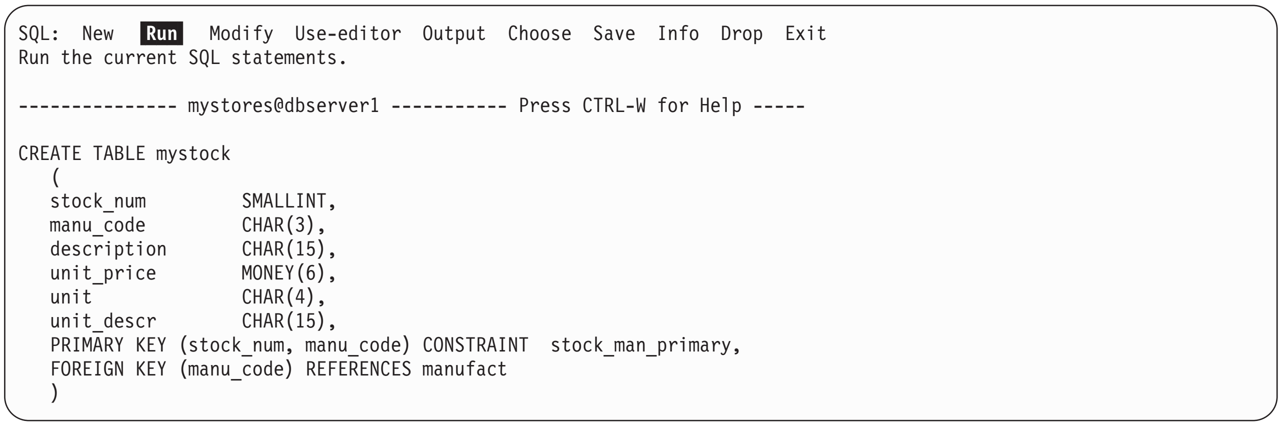
Figure 3-4. SQL statement text-entry screen with scrolling

When you finish entering a new SQL statement or statements, press **ESC** to return to the SQL menu.

Important: If you do not use the Save option to save your typed statements, they will be deleted the next time you select an option that clears the SQL editor (such as New or Choose).

Run an SQL statement

After you exit the editor screen, the SQL menu reopens with the Run option highlighted and the statement text is displayed in the bottom of the screen, as the following figure shows.



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

--------------- mystores@dbserver1 ----------- Press CTRL-W for Help -----

CREATE TABLE mystock
   (
   stock_num        SMALLINT,
   manu_code        CHAR(3),
   description      CHAR(15),
   unit_price       MONEY(6),
   unit             CHAR(4),
   unit_descr       CHAR(15),
   PRIMARY KEY (stock_num, manu_code) CONSTRAINT  stock_man_primary,
   FOREIGN KEY (manu_code) REFERENCES manufact
   )
```

Figure 3-5. The SQL menu with SQL statement text ready to run

Press **RETURN** or the **R** key to select the Run option. DB-Access first checks each statement to ensure that it conforms to the SQL syntax and usage rules. If your statements contain no syntax mistakes, DB-Access processes them.

Tip: You can check your SQL statements for ANSI compatibility if you set the **DBANSIWARN** environment variable or start DB-Access with the **-ansi** option.

Statements that the Run option supports

The following is a list of SQL statements that you can execute with the Run option.

- ALLOCATE COLLECTION
- ALLOCATE DESCRIPTOR
- ALLOCATE ROW
- ALTER ACCESS_METHOD
- ALTER FRAGMENT
- ALTER FUNCTION
- ALTER INDEX
- ALTER PROCEDURE
- ALTER ROUTINE
- ALTER SECURITY LABEL COMPONENT
- ALTER SEQUENCE
- ALTER TABLE
- BEGIN WORK

- CLOSE
- CLOSE DATABASE
- COMMIT WORK
- CONNECT
- CREATE ACCESS_METHOD
- CREATE AGGREGATE
- CREATE CAST
- CREATE DATABASE
- CREATE DISTINCT TYPE
- CREATE EXTERNAL TABLE
- CREATE FUNCTION
- CREATE FUNCTION FROM
- CREATE INDEX
- CREATE OPAQUE TYPE
- CREATE OPCLASS
- CREATE PROCEDURE
- CREATE ROLE
- CREATE ROUTINE FROM
- CREATE ROW TYPE
- CREATE SCHEMA
- CREATE SECURITY LABEL COMPONENT
- CREATE SECURITY LABEL
- CREATE SECURITY POLICY
- CREATE SEQUENCE
- CREATE SYNONYM
- CREATE TABLE
- CREATE TRIGGER
- CREATE VIEW
- CREATE XADATASOURCE
- CREATE XADATASOURCE TYPE
- DATABASE
- DEALLOCATE COLLECTION
- DEALLOCATE DESCRIPTOR
- DEALLOCATE ROW
- DECLARE
- DELETE
- DESCRIBE
- DESCRIBE INPUT
- DISCONNECT
- DROP ACCESS METHOD
- DROP AGGREGATE
- DROP CAST
- DROP DATABASE
- DROP FUNCTION
- DROP INDEX

- DROP OPAQUE TYPE
- DROP OPCLASS
- DROP PROCEDURE
- DROP ROLE
- DROP ROW TYPE
- DROP SECURITY LABEL COMPONENT/POLICY/LABEL
- DROP SEQUENCE
- DROP SYNONYM
- DROP TABLE
- DROP TRIGGER
- DROP TYPE
- DROP VIEW
- DROP XADATASOURCE
- DROP XADATASOURCE TYPE
- EXECUTE
- EXECUTE FUNCTION
- EXECUTE IMMEDIATE
- EXECUTE PROCEDURE
- FETCH
- FLUSH
- FREE
- GET DESCRIPTOR
- GET DIAGNOSTICS
- GRANT
- GRANT DBSECADM
- GRANT DEFAULT ROLE
- GRANT EXEMPTION
- GRANT FRAGMENT
- GRANT SECURITY LABEL
- INFO
- INSERT
- LOAD
- LOCK TABLE
- MERGE
- OPEN
- OUTPUT
- PREPARE
- PUT
- RENAME COLUMN
- RENAME DATABASE
- RENAME INDEX
- RENAME SEQUENCE
- RENAME TABLE
- REVOKE
- REVOKE DBSECADM

- REVOKE DEFAULT ROLE
- REVOKE EXEMPTION
- REVOKE FRAGMENT
- REVOKE SECURITY LABEL
- ROLLBACK WORK
- SAVE EXTERNAL DIRECTIVES
- SELECT
- SET AUTOFREE
- SET COLLATION
- SET CONNECTION
- SET CONSTRAINTS
- SET DATASKIP
- SET DEBUG FILE TO
- SET DEFERRED PREPARE
- SET DESCRIPTOR
- SET ENCRYPTION PASSWORD
- SET ENVIRONMENT
- SET EXPLAIN
- SET ISOLATION
- SET LOCK MODE
- SET LOG
- SET OPTIMIZATION
- SET PDQPRIORITY
- SET ROLE
- SET SESSION AUTHORIZATION
- SET STATEMENT CACHE
- SET TRANSACTION
- START VIOLATIONS TABLE
- STOP VIOLATIONS TABLE
- TRUNCATE
- UNLOAD
- UNLOCK TABLE
- UPDATE
- UPDATE STATISTICS
- WHENEVER

For information about additional statements for Optical Subsystem, see the *IBM Informix Optical Subsystem Guide*.

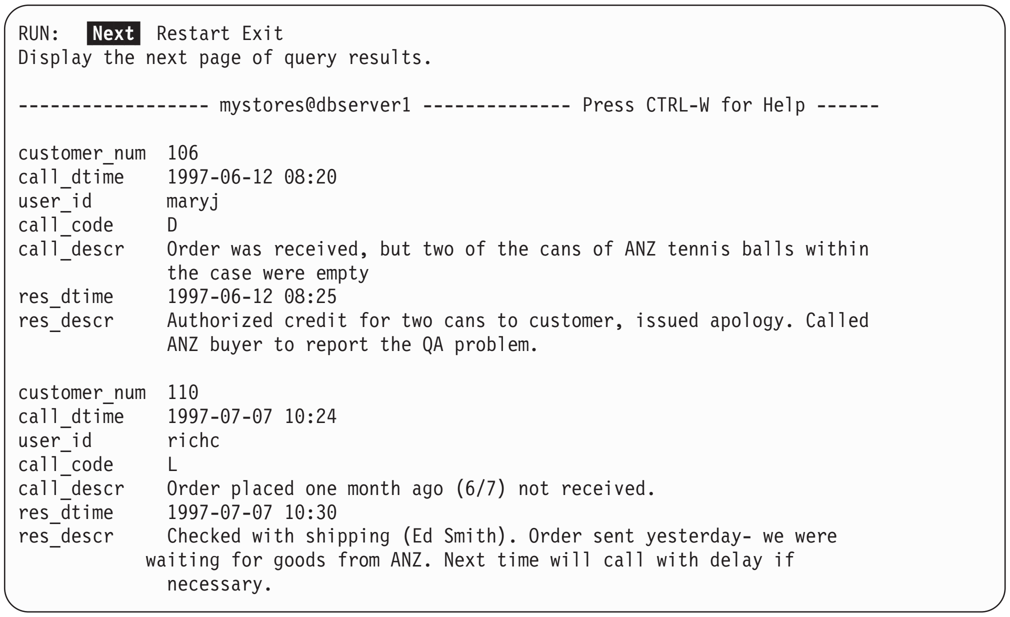
Tip: To execute statements that are not listed, use the SQL menu options New (or Use-editor) and Save to enter and save them, and then run the saved file from the command line.

View successful results

If a statement other than a query is completed successfully, a message is displayed at the bottom of the screen. For the statement that Figure 3-5 on page 3-5 shows, the following message is displayed:

```
Table created.
```

If you use the Run option with a SELECT statement and that SELECT statement runs correctly, DB-Access displays the requested results below the header. If your query retrieves more rows than can fit on a single screen, the results screen has a menu at the top, as the following figure shows.



```
RUN:  Next  Restart Exit
Display the next page of query results.

------------------ mystores@dbserver1 -------------- Press CTRL-W for Help ------

customer_num  106
call_dtime    1997-06-12 08:20
user_id       maryj
call_code     D
call_descr    Order was received, but two of the cans of ANZ tennis balls within
              the case were empty
res_dtime     1997-06-12 08:25
res_descr     Authorized credit for two cans to customer, issued apology. Called
              ANZ buyer to report the QA problem.

customer_num  110
call_dtime    1997-07-07 10:24
user_id       richc
call_code     L
call_descr    Order placed one month ago (6/7) not received.
res_dtime     1997-07-07 10:30
res_descr     Checked with shipping (Ed Smith). Order sent yesterday- we were
            waiting for goods from ANZ. Next time will call with delay if
              necessary.
```

Figure 3-6. The menu for displaying additional query results

Important: When running a query that returns more than one screen of data, it is important to note that a cursor is still open and its corresponding locks are still held until all data is returned.

Advance through the output as follows:

- With the Next option highlighted, press **RETURN** to advance the display through the selected rows. The last screen of data has a message at the bottom that indicates the number of rows that the query returned.
- Select Restart to return to the first screen of query output.
- Select Exit to exit the output display and return to the SQL menu. If you exit before you reach the last output screen, a message at the bottom of the screen indicates that you interrupted the query.

What happens when errors occur

If you make any syntax or typing mistakes in an SQL statement, DB-Access does not process the statement. Instead, it continues to display the text of the statement with a message that describes the error. For example, if a syntax error occurs, the following message is displayed at the bottom of the screen:

```
201: A syntax error has occurred.
```

If an execution or runtime error occurs, DB-Access continues to process the statement and returns an error message. For example, if you try to create a table that was already created, the following message is displayed at the bottom of the screen:

```
310: Table (mavis.mystock) already exists in database.
```

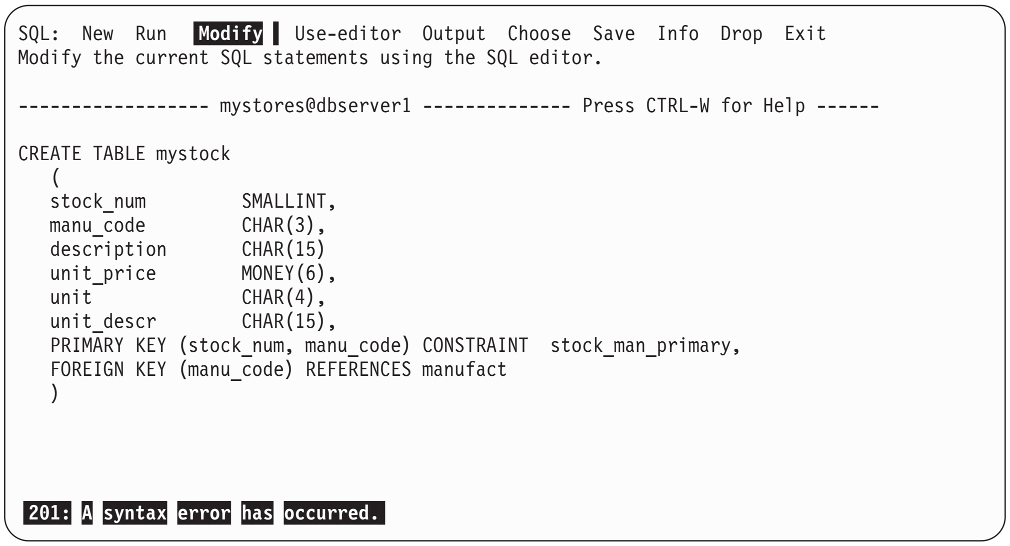
If you try to execute a statement that contains more than one SQL statement, you might not see the error message immediately. If, for example, the first statement is a SELECT statement that runs correctly and the next statement contains a typing error, the data that the first statement retrieved shows on the screen before the error message is displayed for the second statement.

When DB-Access detects an error, it gives you an opportunity to edit the statement that caused the error. Processing stops when the Modify option on the SQL menu is highlighted. Select one of the following methods to correct the statement:

- Press **RETURN** to choose Modify, which returns you to the SQL editor.
- Select the Use-editor option to use the default editor of your choice.

Modify an SQL statement

When DB-Access finds an error in an SQL statement that you are trying to run, the Modify option is highlighted on the SQL menu, and the current statement text and error message are displayed, as the following figure shows.



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

----------------- mystores@dbserver1 ------------- Press CTRL-W for Help ------

CREATE TABLE mystock
   (
   stock_num         SMALLINT,
   manu_code         CHAR(3),
   description       CHAR(15)
   unit_price        MONEY(6),
   unit              CHAR(4),
   unit_descr        CHAR(15),
   PRIMARY KEY (stock_num, manu_code) CONSTRAINT  stock_man_primary,
   FOREIGN KEY (manu_code) REFERENCES manufact
   )




 201: A syntax error has occurred.
```

Figure 3-7. The SQL menu with SQL statement text to be modified

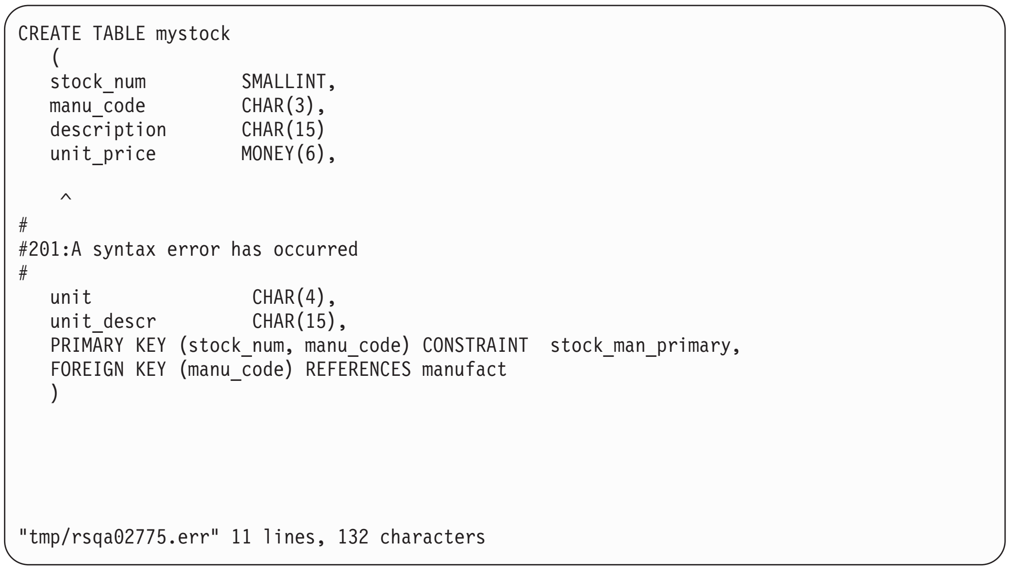
If you press **RETURN**, DB-Access calls the SQL editor and positions the cursor on the line with the first error. You can correct the error with the SQL editor, or you can press **ESC** to exit to the SQL menu and select the Use-editor option to edit the statement with your system editor. To exit, perform one of the following steps:

- If you use the SQL editor to make changes, press **ESC** when you finish editing the statement.
- If you use the system editor to make changes, exit the file according to the convention for that editor.

You then return to the SQL menu, where you can press **RETURN** to run the statement again.

If the SQL statement is new, the screen is blank. If you want to change or call up this statement with the Choose option, the text of the SQL statement is displayed on the screen.

If an error occurs while you run an SQL statement, the edit screen contains the error message with a pointer to the likely cause. The following figure shows how an editor screen might look after a syntax error. The editor used in this example displays the name of the temporary file assigned to the SQL statement.



```
CREATE TABLE mystock
   (
   stock_num        SMALLINT,
   manu_code        CHAR(3),
   description      CHAR(15)
   unit_price       MONEY(6),

    ^
#
#201:A syntax error has occurred
#
   unit              CHAR(4),
   unit_descr        CHAR(15),
   PRIMARY KEY (stock_num, manu_code) CONSTRAINT  stock_man_primary,
   FOREIGN KEY (manu_code) REFERENCES manufact
   )




"tmp/rsqa02775.err" 11 lines, 132 characters
```

Figure 3-8. A temporary text-editing file with SQL statement text to be corrected

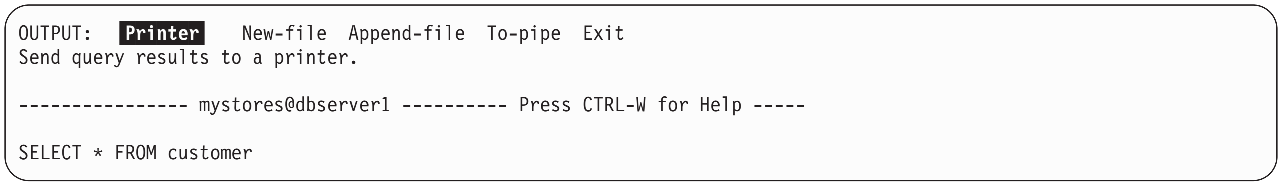
Make your corrections to the text. When you finish entering or modifying your SQL statement or statements, exit the editor as you normally do. The SQL menu reopens with the Run option highlighted. The statement text is displayed in the bottom half of the screen.

Press **RETURN** to run the statement or select another menu option to save the statement in a command file or direct its output.

Redirect query results

The output from a SELECT statement is normally displayed on the screen. You can, instead, use the Output option on the SQL menu to route query results to the printer, store them in a system file, or pipe them to a program. This option has the same functionality as the OUTPUT statement of SQL.

The SELECT statement must be on the screen as the current statement. Then you can select the Output option from the SQL menu, which displays the OUTPUT menu, as the following figure shows.



```
OUTPUT:   Printer   New-file  Append-file  To-pipe  Exit
Send query results to a printer.

---------------- mystores@dbserver1 ---------- Press CTRL-W for Help -----

SELECT * FROM customer
```

Figure 3-9. The OUTPUT menu for redirecting query results

To return to the SQL menu without redirecting query results from an OUTPUT screen, press the Interrupt key.

Send output to a printer

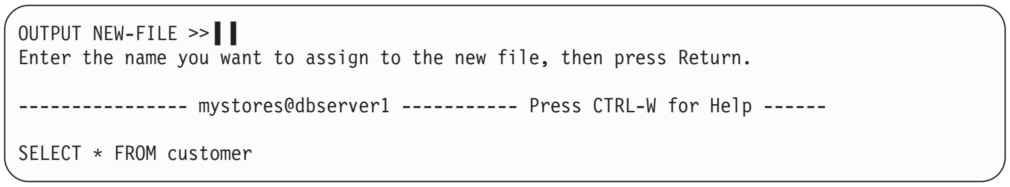
To send your query results directly to a printer, select the Printer option from the OUTPUT menu. DB-Access sends the results to your default printer and displays a message on the bottom of the screen that indicates how many rows were retrieved. The query results do not show on the screen. You can set the **DBPRINT** environment variable to specify a default printer.

Send output to a file

You can write query results to a new file or append the results to an existing file. If you do not specify a path when DB-Access prompts you for a file name, the file will be stored in the directory that you were in when you started DB-Access.

The New-file option

To create a new file to store query results, select the New-file option on the OUTPUT menu. The OUTPUT NEW-FILE screen opens, as the following figure shows.



```
OUTPUT NEW-FILE >>
Enter the name you want to assign to the new file, then press Return.

--------------- mystores@dbserver1 ----------- Press CTRL-W for Help ------

SELECT * FROM customer
```

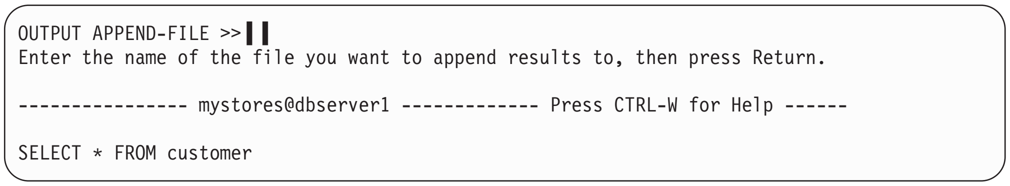
Figure 3-10. The OUTPUT NEW-FILE screen

Type a name for the file and press **RETURN**. DB-Access forwards the results of the query to that file and displays a message that indicates how many rows were retrieved. The query results do not show on the screen.

Warning: If you enter the name of an existing file, this procedure overwrites the existing file with the query results.

The Append-file option

To add your query results to the end of an existing file without replacing its contents, select the Append-file option on the OUTPUT menu. The OUTPUT APPEND-FILE screen opens, as the following figure shows.



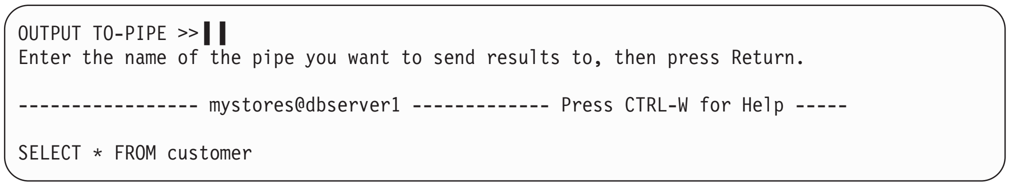
```
OUTPUT APPEND-FILE >>▌
Enter the name of the file you want to append results to, then press Return.

---------------- mystores@dbserver1 ------------- Press CTRL-W for Help ------

SELECT * FROM customer
```

Figure 3-11. The OUTPUT APPEND-FILE screen

Type the name of an existing file where you want to append the query results and press **RETURN**. DB-Access appends the query results to the end of that file and displays a message that indicates how many rows were retrieved. The query results do not show on the screen.

Send output to a pipe

If you want to send query results to a pipe, select the To-pipe option on the OUTPUT menu. The OUTPUT TO-PIPE screen opens, as the following figure shows.



```
OUTPUT TO-PIPE >>▌
Enter the name of the pipe you want to send results to, then press Return.

----------------- mystores@dbserver1 ------------- Press CTRL-W for Help -----

SELECT * FROM customer
```

Figure 3-12. The OUTPUT TO-PIPE screen

Specify a target program, such as **more**, through which to pipe output. DB-Access sends the results to that pipe.

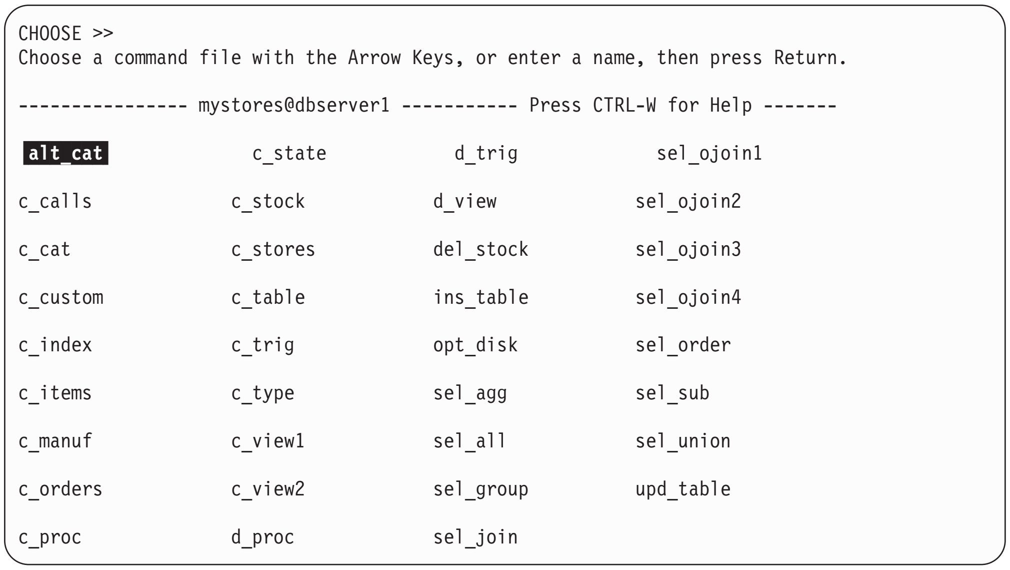
On UNIX systems, you must have permission to run the target program.

On Windows systems, the **cat** utility can serve as a target program through which to pipe output.

Choose an existing SQL statement

When you save SQL statements in a command file, as described in “Save the current SQL statement” on page 3-14, you can retrieve the command file and run or edit the SQL statements at any time.

Select the Choose option on the SQL menu to display the CHOOSE screen with a list of the command files that you can access. These files have the extension `.sql`, although the extension is not shown. For example, the following figure lists the command files included in the demonstration database.



```
CHOOSE >>
Choose a command file with the Arrow Keys, or enter a name, then press Return.

---------------- mystores@dbserver1 ----------- Press CTRL-W for Help -------

 alt_cat             c_state           d_trig            sel_ojoin1

c_calls            c_stock           d_view            sel_ojoin2

c_cat              c_stores          del_stock         sel_ojoin3

c_custom           c_table           ins_table         sel_ojoin4

c_index            c_trig            opt_disk          sel_order

c_items            c_type            sel_agg           sel_sub

c_manuf            c_view1           sel_all           sel_union

c_orders           c_view2           sel_group         upd_table

c_proc             d_proc            sel_join
```

Figure 3-13. The CHOOSE screen listing current `.sql` files

If no current database exists, the list includes all the command files located in the current directory and in any directories that the **DBPATH** environment variable specifies.

Important: This list includes only those file names that have the `.sql` extension. If you create a new SQL file outside of DB-Access and save it without the `.sql` extension, it will not show in the list of files to choose. Add the `.sql` extension to the file name and then select Choose again.
DB-Access can only recognize files that are stored in the directory from which you started DB-Access. If the Choose command results in an empty list, and you know you have command files, exit DB-Access, change directories to the directory that contains your `.sql` files, and restart DB-Access.

To select a command file, use the arrow keys to highlight its name or enter the name of the file at the prompt.

When the SQL menu reopens, it shows the command file statements on the screen as the current statements. To modify, run, edit, or output these statements, choose the appropriate menu option.

To leave the CHOOSE screen without selecting a command file, press the Interrupt key, which returns you to the SQL menu.

Save the current SQL statement

You can save SQL statements in a file for later use, such as to invoke the statements from the command line (see "Execute a command file" on page 1-13) or retrieve the saved statements with the Choose option on the SQL menu.

To save the current SQL statement or statements in a file, select the Save option on the SQL menu. The SAVE screen opens and prompts you to enter a name for the command file, as the following figure shows.

```
SAVE >>
Enter the name you want to assign to the command file.

-------------------mystores@dbserver1 --------- Press CTRL-W for Help -----

select max (ship_charge), min (ship_charge)
        from orders;
```

Figure 3-14. The SAVE screen for saving statements in a file

You assign the left portion of the file name. Use 1 to 10 characters. Start with a letter, then use any combination of letters, numbers, and underscores (_). Press **RETURN** to save the file.

For UNIX, you can use uppercase and lowercase letters in the name. However, remember that UNIX operating systems are case-sensitive. The file `orders` is not the same as `Orders` or `ORDERS`.

DB-Access appends the extension `.sql` to the name that you assign when it stores the statements in a file. For example, if you name your file `cust1`, DB-Access stores the file with the name **cust1.sql**. The CHOOSE screen still lists `cust1`, but the operating system identifies the same file as `cust1.sql` if you list the directory files from the command line.

To leave the SAVE screen without assigning a name to a command file, press the Interrupt key, and you return to the SQL menu.

Display table information

Use the Info option on the SQL menu to display information about the columns, indexes, access privileges, reference privileges, constraints (referential, primary-key, check, unique, and defaults), triggers, status, and fragmentation strategy of a table. The Info option has the same purpose as the SQL statement INFO.

The INFO menu displays the following options.

Option	Purpose	Instructions
Columns	Lists the columns in the specified table, shows the data type, and shows whether null values are allowed	"Display column information" on page 5-28
Indexes	Lists the name, owner, and type (unique or duplicate) of each index for a specified table. (The display also shows if the index is clustered, the access method used, and the names of the columns that are indexed.)	"Display index information" on page 5-31
Privileges	Lists the users who have table-level Select, Update, Insert, Delete, Index, and Alter privileges	"Display table-level privileges" on page 5-32
References	Lists the users who have the table-level References privilege for the specified table and which columns they can reference	"Display references privileges" on page 5-32

Option	Purpose	Instructions
Status	Lists the table name, the table owner, the size of the row (in number of bytes), the number of rows in the table (as of the last UPDATE STATISTICS statement), the number of columns in a row, and the date the table was created	none
cOnstraints	Displays referential, primary-key, check, and unique constraints and default values for the columns in the specified table	"Display column constraints and defaults" on page 5-32
triGgers	Displays header and body information for a specified trigger	"Display triggers" on page 5-34
Table	Redisplays the INFO FOR TABLE menu so that you can select a new table to request information about the INFO menu	none
Fragments	Displays fragmentation strategy for the selected table	"Drop a table" on page 5-36
Exit	Returns to the SQL menu	none

The Table option on the main menu displays a TABLE menu, which in turn has an Info option. The Info option screens are the same for both the SQL menu and TABLE menu. For more information about the Info option, see "Display table information" on page 5-26.

To leave the INFO FOR TABLE screen without requesting table information, press the Interrupt key.

Drop an SQL statement

Your current database directory stores files that contain SQL statements. You might have installed some of these files with the demonstration database. You might have created other files and put them in the directory with the Save option on the SQL menu. Additional files might have become available when you installed a language supplement with DB-Access.

To remove command files from the current database directory, select the Drop option on the SQL menu. The DROP COMMAND FILE screen opens with an alphabetical list of command files in the current database, as the following figure shows.

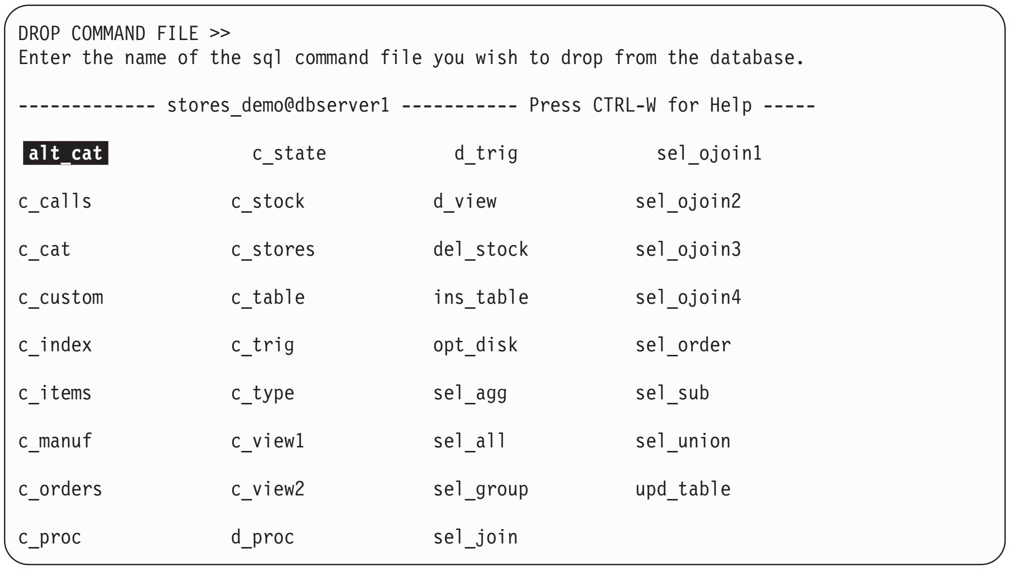

```
DROP COMMAND FILE >>
Enter the name of the sql command file you wish to drop from the database.

------------- stores_demo@dbserver1 ----------- Press CTRL-W for Help -----

 alt_cat            c_state           d_trig            sel_ojoin1

c_calls            c_stock           d_view            sel_ojoin2

c_cat              c_stores          del_stock         sel_ojoin3

c_custom           c_table           ins_table         sel_ojoin4

c_index            c_trig            opt_disk          sel_order

c_items            c_type            sel_agg           sel_sub

c_manuf            c_view1           sel_all           sel_union

c_orders           c_view2           sel_group         upd_table

c_proc             d_proc            sel_join
```

Figure 3-15. The DROP COMMAND FILE screen with sample files

If GLS is enabled, the order in which DB-Access lists the names of command files might vary, depending on the locale you use.

Type the name of the command file that you want to drop or highlight it with the arrow keys and press **RETURN**. A special menu opens that asks for confirmation before it drops the command file as the following figure shows.

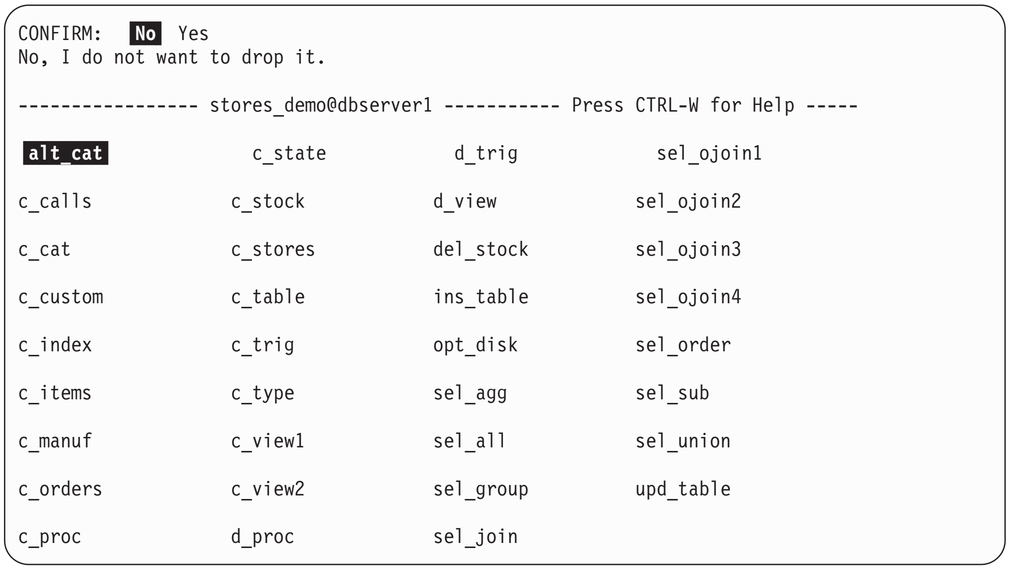

```
CONFIRM:  No  Yes
No, I do not want to drop it.

----------------- stores_demo@dbserver1 ----------- Press CTRL-W for Help -----

 alt_cat            c_state           d_trig            sel_ojoin1

c_calls            c_stock           d_view            sel_ojoin2

c_cat              c_stores          del_stock         sel_ojoin3

c_custom           c_table           ins_table         sel_ojoin4

c_index            c_trig            opt_disk          sel_order

c_items            c_type            sel_agg           sel_sub

c_manuf            c_view1           sel_all           sel_union

c_orders           c_view2           sel_group         upd_table

c_proc             d_proc            sel_join
```

Figure 3-16. The DROP COMMAND FILE confirmation menu

The default is No to help prevent you from deleting a command file by mistake. To drop the highlighted command file, press the **Y** key or use the right arrow key to highlight Yes and press **RETURN**. DB-Access drops the command file and returns you to the SQL menu.

To leave the DROP COMMAND FILE screen without dropping a command file, press the **N** key, **RETURN**, or the Interrupt key. You then return to the SQL menu.

Support for SPL Routines

You can create and execute routines written in SPL from the SQL menu.

You can store the SPL routine in a separate command file and then call it from an application or execute it as a stand-alone program. After you create the SPL routine, you can execute it within DB-Access with the appropriate SQL statement. The following example details the steps.

To create and execute an SQL routine

1. To create the text of the routine, type directly in the NEW screen or the Use-editor screen. Enter the SPL and SQL statements for your routine in the statement block of a CREATE PROCEDURE statement.

   Use the CREATE FUNCTION statement if the routine returns values.

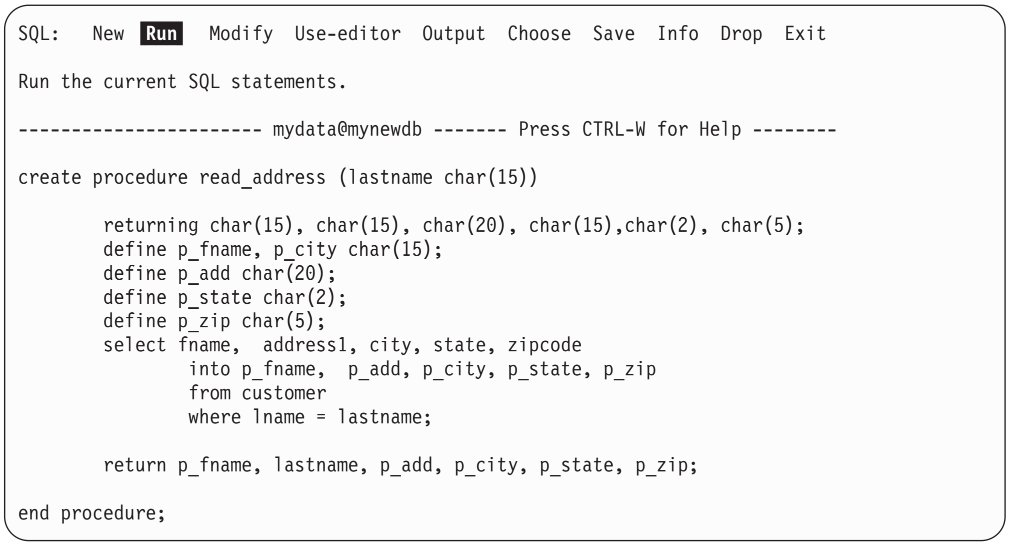
   For more information about the CREATE FUNCTION statement, see the *IBM Informix Guide to SQL: Syntax*.
2. Use the Run option to create the routine and register it in the **sysprocedures** system catalog table.
3. Use the NEW screen to enter an EXECUTE PROCEDURE statement that names the routine that you want to run.

   If you use IBM Informix and created your routine with the CREATE FUNCTION statement, enter an EXECUTE FUNCTION statement to run the function.
4. Use the Run option to execute the routine and display the results.

The following figure shows the text of the routine in the `c_proc.sql` command file, which is supplied with the demonstration database. To try this routine, use the Choose option and then select **c_proc**.

If you use Informix, change the word *procedure* in `c_proc.sql` to *function* because the routine returns a value.

To register the routine in the database, select the Run option, as the following figure shows.



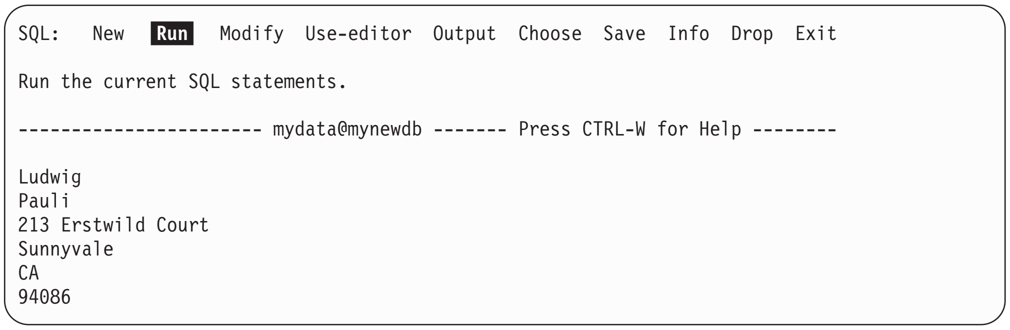
```
SQL:   New  Run   Modify  Use-editor  Output  Choose  Save  Info  Drop  Exit

Run the current SQL statements.

----------------------- mydata@mynewdb ------- Press CTRL-W for Help --------

create procedure read_address (lastname char(15))

        returning char(15), char(15), char(20), char(15),char(2), char(5);
        define p_fname, p_city char(15);
        define p_add char(20);
        define p_state char(2);
        define p_zip char(5);
        select fname,  address1, city, state, zipcode
                into p_fname,  p_add, p_city, p_state, p_zip
                from customer
                where lname = lastname;

        return p_fname, lastname, p_add, p_city, p_state, p_zip;

end procedure;
```

Figure 3-17. Displaying the text of an SPL routine on the SQL menu

DB-Access displays a message to indicate that the database server created the routine. To execute the routine, select New from the SQL menu and then enter the appropriate EXECUTE statement. In the following example, the user requests the address of a customer whose last name is Pauli:

```
EXECUTE PROCEDURE read_address ("Pauli")
```

After you enter the EXECUTE PROCEDURE or EXECUTE FUNCTION statement on the NEW screen, press **Esc** to return to the SQL menu. Select Run from the SQL menu to execute the routine. The following figure shows the result of executing the routine.



```
SQL:   New  Run   Modify  Use-editor  Output  Choose  Save  Info  Drop  Exit

Run the current SQL statements.

----------------------- mydata@mynewdb ------- Press CTRL-W for Help --------

Ludwig
Pauli
213 Erstwild Court
Sunnyvale
CA
94086
```

Figure 3-18. Result of executing an SPL routine on the SQL menu

Tip: SPL routines are stored in the system catalog tables in executable format. Use the Routines option on the DATABASE INFO menu to display a list of the routines in the current database or to display the text of a specified routine.

Chapter 4. The Database option

These topics describe how to use the Database option. To perform any of the following actions, select the Database option from the main menu.

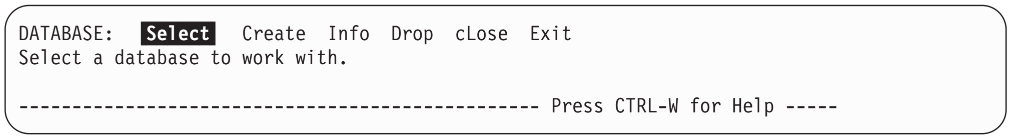
- Create a database or select a database.

  The database you work with is called the *current* database.
- Retrieve and display information about a database, such as available dbspaces and the text of routines.
- Delete an existing database or close the current database.
- Commit or rollback transactions.

You can only access databases that are on the current database server. To select a database server as current, you can specify a database server when you start DB-Access, you can use the Connection menu, or you can run a CONNECT statement from the SQL menu. If you do not explicitly select a database server, DB-Access uses the default database server that the **$INFORMIXSERVER** environment variable specifies as the current database.

Select a database menu option

The following figure shows the DATABASE menu. To reach the DATABASE menu from the main menu, press the **D** key or highlight the Database option and press **RETURN**.



```
DATABASE:  Select  Create  Info  Drop  cLose  Exit
Select a database to work with.

------------------------------------------------ Press CTRL-W for Help -----
```

Figure 4-1. The DATABASE menu

The DATABASE menu displays the following options.

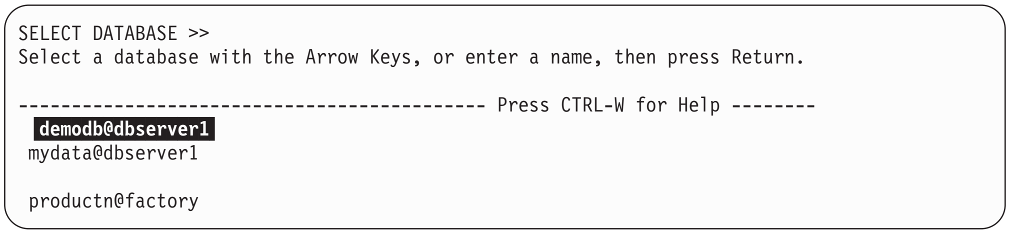
Option	Purpose	Instructions
Select	Makes a database the current database	Figure 4-5 on page 4-4
Create	Builds a new database and makes that database the current database	"Create a database" on page 4-3
Info	Displays information about the current database	"Display database information" on page 4-5
Drop	Removes a database from the system	"Delete a database" on page 4-8
cLose	Closes the current database	"Close a database" on page 4-9
Exit	Exits the DATABASE menu and returns you to the main menu	none

If you select or create a database when another database is already open, DB-Access closes that database before it makes your selection the current or new database. Figure 4-17 on page 4-10 shows the TRANSACTION menu that opens if you attempt to open a new database without first terminating a transaction.

If you enter the name of a nonexistent database or a database that DB-Access cannot locate, an error message is displayed.

Select a database

To work with an existing database, choose the Select option from the DATABASE menu. The SELECT DATABASE screen opens, as the following figure shows.



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

------------------------------------------- Press CTRL-W for Help --------
  demodb@dbserver1
 mydata@dbserver1

 productn@factory
```

Figure 4-2. The SELECT DATABASE screen

The SELECT DATABASE screen also opens whenever you must specify a database, such as when you choose the Table or Query-language option without specifying a database on the DB-Access command line.

Important: In the SELECT DATABASE screen, the names of databases are limited to 18 characters. If a database name is longer than 18 characters, you will see the first 17 characters of the name followed by a '+' sign. Enter a '+' sign to display the complete long name in **vi**. To exit from **vi**, press `ESC ZZ`.

To leave the SELECT DATABASE screen and return to the DATABASE menu without selecting a database, press the Interrupt key.

List of available databases

When the SELECT DATABASE screen opens, the first database in the list of available databases is highlighted, accompanied by the names of database servers. The list is organized alphabetically by database server and then by database for each database server. You can display a maximum of 512 database names on the SELECT DATABASE screen.

The list of available databases that is displayed depends on two factors:

- The settings of certain environment variables.
  - If you use one database server, DB-Access displays the names of all databases on the current database server and in your **DBPATH** setting.
  - If you use multiple database servers, the **ONCONFIG** environment variable determines the current database server.
- The current connection. For example:
  - If no explicit connection exists, DB-Access displays the databases in the **DBPATH** setting.
  - If a current explicit connection exists, all databases in the **DBPATH** setting that pertain to the current database server are displayed.

Specify a database

You can select a database name on the SELECT DATABASE screen in any of the following ways:

- To select the first database on the list, which is already highlighted, press **RETURN**.

- Use the arrow keys to highlight the name of the database and press **RETURN**.
- Type the database name and press **RETURN**.

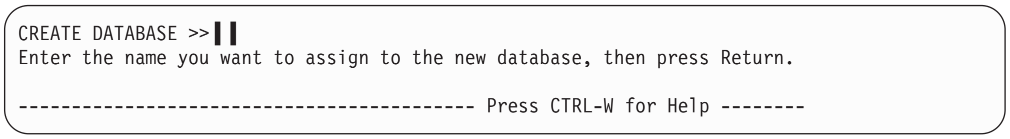
For example, to select the **demodb** database on the current database server, type `demodb` or highlight **demodb@dbserver1** and press **RETURN**.

To specify a database on another database server, include the database server with the database name. For example, to select the **productn** database on the **factory** database server, type the name or highlight **productn@factory** and press **RETURN**.

The name of the database that you select open on the dashed line below the screen header.

Create a database

To create a new database instead of selecting an existing one, select the Create option from the DATABASE menu. The CREATE DATABASE screen opens, as the following figure shows.



```
CREATE DATABASE >>
Enter the name you want to assign to the new database, then press Return.

------------------------------------------- Press CTRL-W for Help --------
```

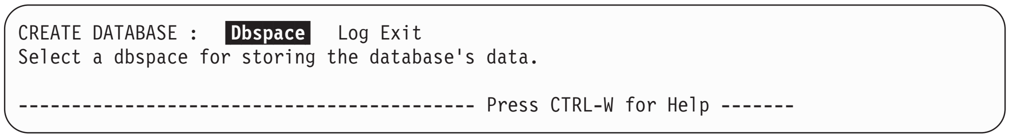
Figure 4-3. The CREATE DATABASE screen

Enter a name for the database that you want to create and press **RETURN**.

Tip: You can assign any name to your database, provided that you follow the syntax guidelines described in the *IBM Informix Guide to SQL: Syntax*.

To create a database on another database server, specify the server name with the database name. Follow the syntax guidelines described in the *IBM Informix Guide to SQL: Syntax*.

After you name the new database, the CREATE DATABASE menu opens as the following figure shows.



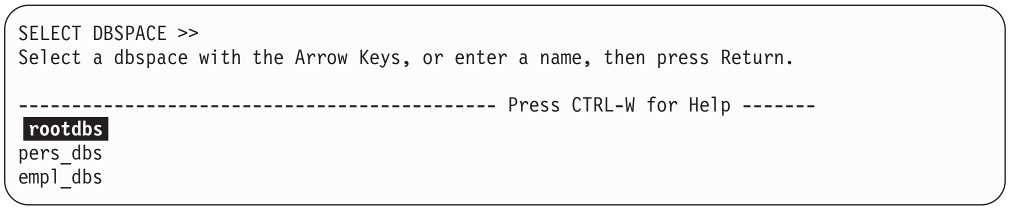
```
CREATE DATABASE :  Dbspace  Log Exit
Select a dbspace for storing the database's data.

------------------------------------------- Press CTRL-W for Help -------
```

Figure 4-4. The CREATE DATABASE menu

You can specify that a database be stored in a nonroot dbspace or create a database with or without buffered transaction logging or create an ANSI-compliant database.

Specify a dbspace

Select the Dbspace option and the SELECT DBSPACE screen opens, as the following figure shows.



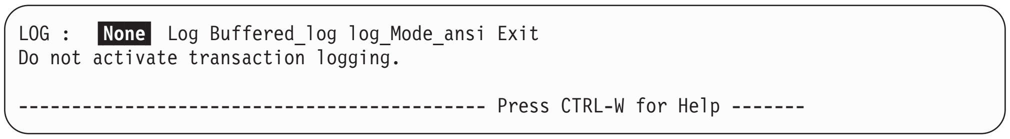
```
SELECT DBSPACE >>
Select a dbspace with the Arrow Keys, or enter a name, then press Return.

------------------------------------------- Press CTRL-W for Help -------
 rootdbs
pers_dbs
empl_dbs
```

Figure 4-5. The SELECT DBSPACE screen

From the list of dbspaces, select an available dbspace in which to store database data. The default is **rootdbs** or a dbspace that you create as the default. You then return to the CREATE DATABASE menu.

Specify logging

To specify the type of transaction logging, select the Log option. The LOG menu opens, as the following figure shows.



```
LOG :   None  Log Buffered_log log_Mode_ansi Exit
Do not activate transaction logging.

------------------------------------------- Press CTRL-W for Help -------
```

Figure 4-6. The LOG menu

The LOG menu displays the following options.

None Default, does not activate transaction logging.

Log Specifies unbuffered transaction logging.

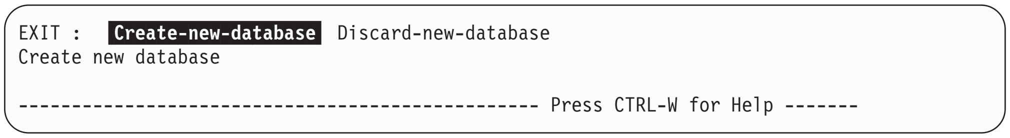
Buffered_log
Specifies buffered transaction logging.

log_Mode_ansi
Creates an ANSI-compliant database with unbuffered transaction logging.

Exit Exits the LOG menu and returns you to the CREATE DATABASE menu.

Exit the CREATE DATABASE menu

When you exit the CREATE DATABASE menu, you must confirm or discard the new database, as the following figure shows.



```
EXIT :   Create-new-database  Discard-new-database
Create new database

----------------------------------------------- Press CTRL-W for Help -------
```

Figure 4-7. The CREATE DATABASE confirmation screen

The default is Create-new-database. Press **RETURN** to create a database with the specified parameters, make it the current database, and return to the DATABASE menu. If you do not want to create the new database, press the **D** key or use the right arrow key to move the cursor to the Discard-new-database option and press **RETURN**. DB-Access does not create a new database.

If you exit without specifying a value for dbspace or logging, the defaults apply to the database.

Display database information

Use the Info option on the DATABASE menu to display information about dbspaces and nondefault locale settings and to read the body of routines in the current database.

Tip: This menu option provides information about the database that the SQL statement INFO does not display.

When you select the Info option on the DATABASE menu, the SELECT DATABASE screen opens, as Figure 4-2 on page 4-2 shows.

After you select a database, the DATABASE INFO menu opens, with the database you selected identified in the dashed line.

The following figure shows the DATABASE INFO menu for IBM Informix.

```
DATABASE INFO: dBspace  Nls  Routine  Databases  Exit
Display DBSPACE information for a database.

------------ demodb@dbserver1 ------- Press CTRL-W for Help -----
```

Figure 4-8. The DATABASE INFO menu for IBM Informix

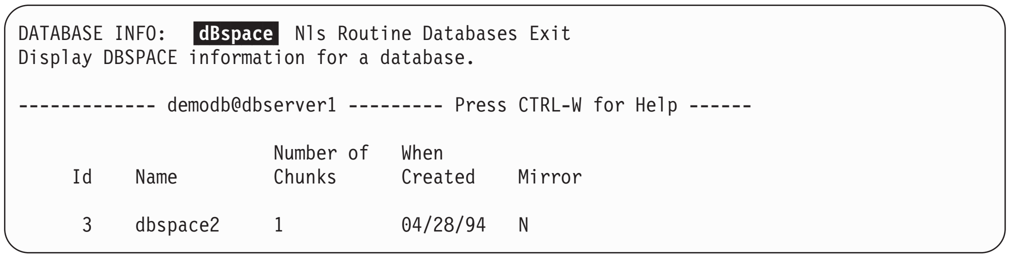
The DATABASE INFO menu displays the following options.

Option	Purpose	Instructions
dBspace	Retrieves information about the dbspaces in the current database.	"Retrieve information about dbspaces"
NlS	Displays Native Language Support (NLS) settings for a database that supports NLS. This option is provided for compatibility with earlier database versions that support NLS.	"Retrieve nondefault locale information" on page 4-6
Routine	In Informix, the routine option lists the procedures and functions in the current database. Select a routine name to see the body of that routine as text.	"Retrieve information about routines" on page 4-7
Databases	Selects a different database about which to retrieve information.	"Select a different database" on page 4-8
Exit	Leaves the DATABASE INFO menu and returns you to the DATABASE menu.	none

To leave the DATABASE INFO menu without displaying information about the current database, press the Interrupt key to return to the DATABASE menu.

Retrieve information about dbspaces

To retrieve information about the dbspaces in the current database, select the dBspace option from the DATABASE INFO menu, as the following figure shows.



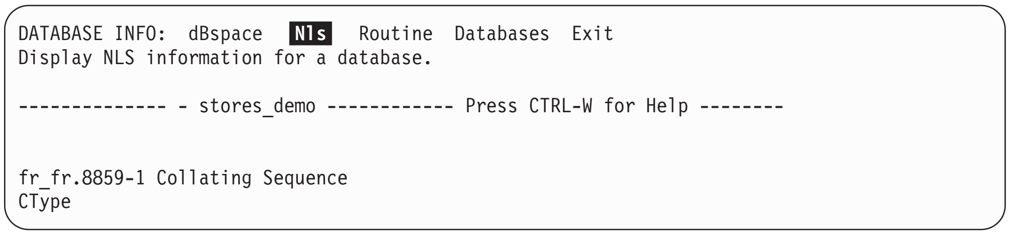
```
DATABASE INFO:   dBspace  Nls Routine Databases Exit
Display DBSPACE information for a database.

------------- demodb@dbserver1 --------- Press CTRL-W for Help ------

                        Number of   When
     Id    Name         Chunks      Created    Mirror

      3    dbspace2     1           04/28/94   N
```

Figure 4-9. The DATABASE INFO menu with dbspaces information displayed

Retrieve nondefault locale information

Global Language Support (GLS) and Native Language Support (NLS) affect the order in which lists are displayed in DB-Access. GLS enables the display and appropriate ordering of non-English-language data. Earlier database server versions used NLS for this purpose.

If the current database supports NLS, you can select the Nls option on the DATABASE INFO menu to display information about collating sequence and C CType (character classification type), as the following figure shows.



```
DATABASE INFO:  dBspace   Nls  Routine  Databases  Exit
Display NLS information for a database.

-------------- - stores_demo ------------ Press CTRL-W for Help --------


fr_fr.8859-1 Collating Sequence
CType
```

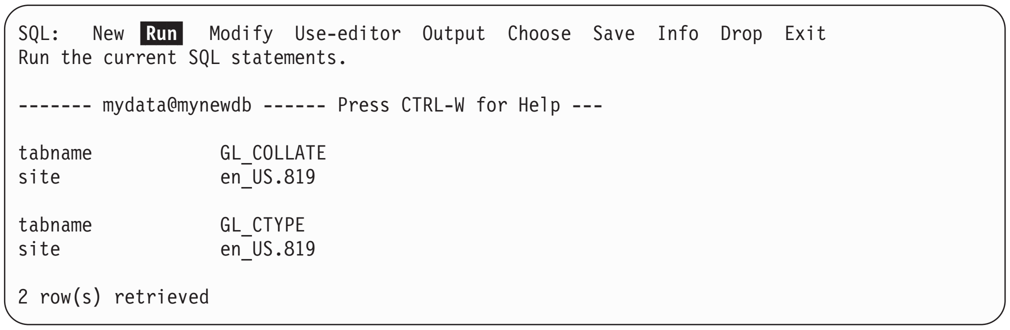
Figure 4-10. The DATABASE INFO menu with NLS information displayed

An error message is displayed if the database does not support NLS or the environment variables for NLS are not properly set.

DB-Access does not provide an option on the DATABASE INFO menu for displaying the GLS collating sequence and character classification type. To obtain information about the GLS locale enabled for your database server, enter the following query with the SQL editor that is described in Chapter 3, "The Query-language option," on page 3-1:

```
SELECT tabname, site FROM systables
WHERE tabid = 90 OR tabid = 91
```

The row with tabid 90 stores the COLLATION category of the database locale. The row with tabid 91 stores the CTYPE category of the database locale. The following figure shows the result of the preceding query for the default U.S. English locale.



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

------- mydata@mynewdb ------ Press CTRL-W for Help ---

tabname            GL_COLLATE
site               en_US.819

tabname            GL_CTYPE
site               en_US.819

2 row(s) retrieved
```

Figure 4-11. Retrieving GLS information

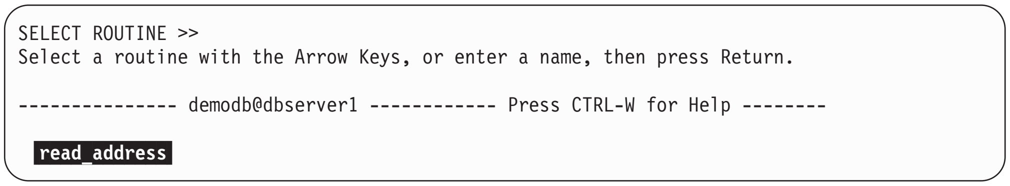
For further information about the COLLATION and CTYPE categories in a GLS locale file, see the *IBM Informix GLS User's Guide*.

Retrieve information about routines

Depending on the database server product that you use, select either the Procedures or Routine option on the DATABASE INFO menu to display the text from a selected routine. The SELECT PROCEDURE or SELECT ROUTINE screen opens and shows a list of SPL routines in the current database.

Restriction: Routine and Procedures options do not display system-created routines. They only display user-defined routines and SPL routines.

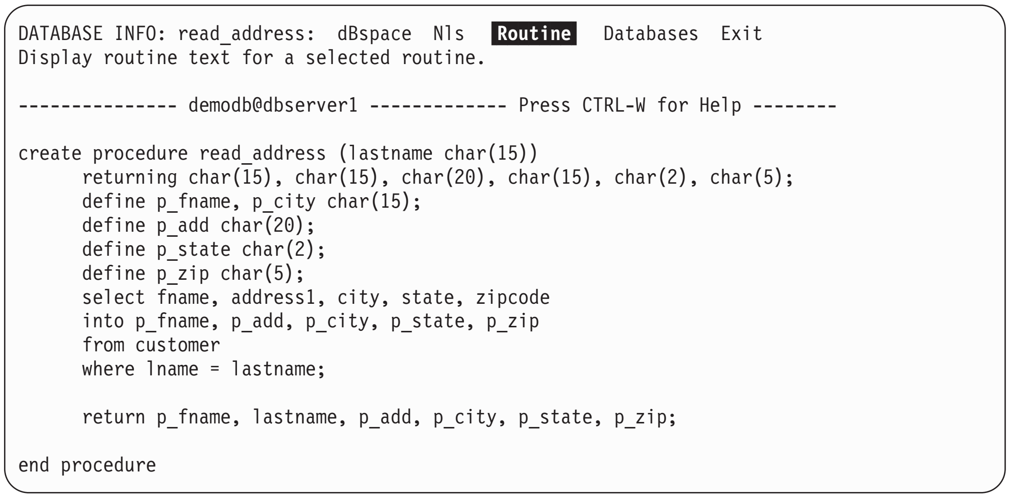
The following figure shows the SELECT ROUTINE screen. The SELECT PROCEDURE screen looks the same although the prompt uses different wording for an SPL routine.



```
SELECT ROUTINE >>
Select a routine with the Arrow Keys, or enter a name, then press Return.

--------------- demodb@dbserver1 ------------ Press CTRL-W for Help --------

  read_address
```

Figure 4-12. The SELECT ROUTINE screen

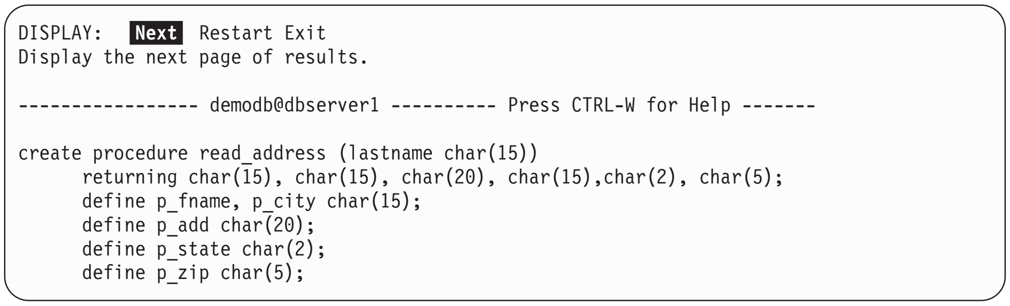
If the routine exists in the system catalog and it fits on the DATABASE INFO menu, the text shows on the screen, as the following figure shows.



```
DATABASE INFO: read_address:  dBspace  Nls   Routine   Databases  Exit
Display routine text for a selected routine.

--------------- demodb@dbserver1 ------------- Press CTRL-W for Help --------

create procedure read_address (lastname char(15))
      returning char(15), char(15), char(20), char(15), char(2), char(5);
      define p_fname, p_city char(15);
      define p_add char(20);
      define p_state char(2);
      define p_zip char(5);
      select fname, address1, city, state, zipcode
      into p_fname, p_add, p_city, p_state, p_zip
      from customer
      where lname = lastname;

      return p_fname, lastname, p_add, p_city, p_state, p_zip;

end procedure
```

Figure 4-13. The DATABASE INFO menu with text of selected routine displayed

If the routine text does not fit on one screen, the DISPLAY menu opens with partial text, as the following figure shows.



```
DISPLAY:   Next  Restart Exit
Display the next page of results.

----------------- demodb@dbserver1 ---------- Press CTRL-W for Help -------

create procedure read_address (lastname char(15))
      returning char(15), char(15), char(20), char(15),char(2), char(5);
      define p_fname, p_city char(15);
      define p_add char(20);
      define p_state char(2);
      define p_zip char(5);
```

Figure 4-14. The DISPLAY menu with partial routine text displayed

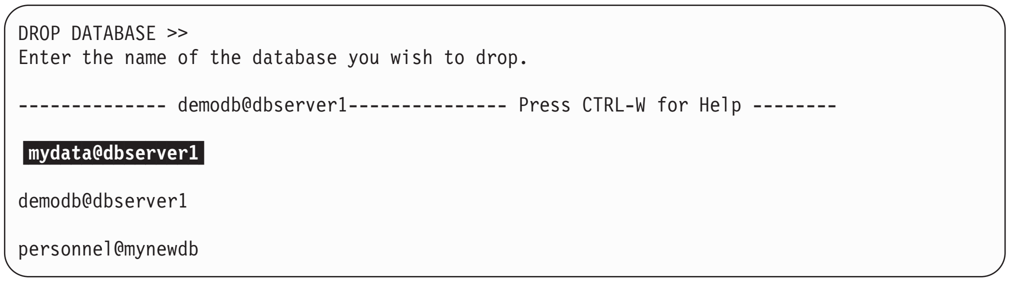
To display the next page of text, select the Next option. To display text from the beginning, select the Restart option.

Select a different database

To display information about a different database, select the Database option on the DATABASE INFO menu. The SELECT DATABASE screen opens and you can select a database, as described in "Select a database" on page 4-2. You can then use the other options of the DATABASE INFO menu.

Delete a database

To delete an existing database on a specified database server, select the Drop option from the DATABASE menu. The DROP DATABASE screen opens, as the following figure shows.



```
DROP DATABASE >>
Enter the name of the database you wish to drop.

------------- demodb@dbserver1--------------- Press CTRL-W for Help --------

 mydata@dbserver1

demodb@dbserver1

personnel@mynewdb
```

Figure 4-15. The DROP DATABASE screen

You cannot delete the current database. The current database is the database whose name shows in the dashed line below the header of the display.

The DROP DATABASE screen

To delete a database, use the DROP DATABASE screen in either of the following ways:

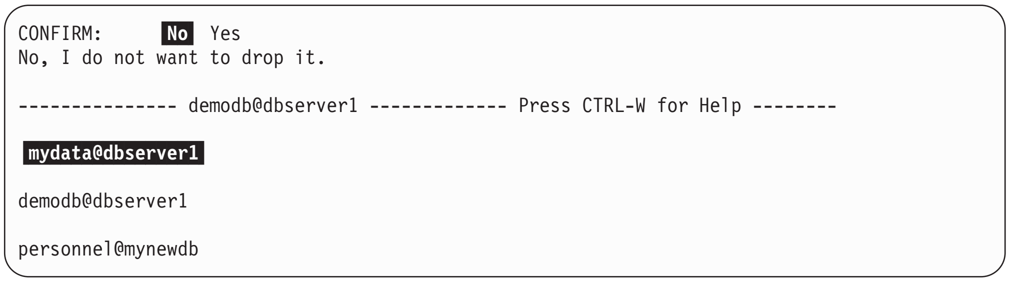
- Type the database name and press **RETURN**.
- Use the arrow keys to highlight the name of the database that you want to delete and press **RETURN**.

For example, to delete the **mydata** database, type `mydata` or highlight **mydata@dbserver1** with an arrow key and press **RETURN**.

To leave the DROP DATABASE screen without deleting a database, press the Interrupt key. You return to the DATABASE menu.

Confirm your decision to delete a database

When you delete a database, DB-Access displays a special menu that asks for confirmation before it deletes the database, as the following figure shows.



```
CONFIRM:      No  Yes
No, I do not want to drop it.

-------------- demodb@dbserver1 ------------- Press CTRL-W for Help --------

 mydata@dbserver1

demodb@dbserver1

personnel@mynewdb
```

Figure 4-16. The DROP DATABASE confirmation menu

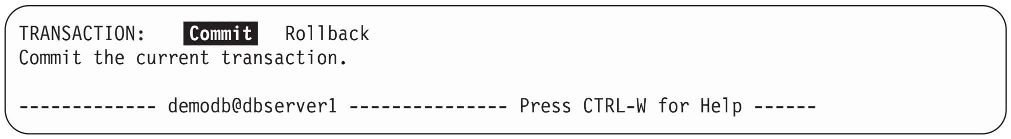
The default is No, which helps prevent deleting a database by mistake. If you want to delete the highlighted database, press the **Y** key or use the right arrow key to highlight Yes, and press **RETURN**. DB-Access deletes the database and all data that it contains. Be absolutely sure that you choose the correct database to delete.

Close a database

To close the current database, choose the cLose option from the DATABASE menu and press **RETURN**. The message Database Closed is displayed at the bottom of the screen, which indicates that the current database is closed. The database name disappears from the Help line, but the database server name remains.

If you select the cLose option when no database name is on the Help line, an error message is displayed.

If you begin a transaction but do not commit it or roll it back, and then try to close a database with transactions, the TRANSACTION menu opens, as the following figure shows.



```
TRANSACTION:    Commit   Rollback
Commit the current transaction.

------------- demodb@dbserver1 --------------- Press CTRL-W for Help ------
```

Figure 4-17. The TRANSACTION menu for databases with transactions

The TRANSACTION menu ensures that you either commit or roll back an active transaction before you close the current database. The following list shows the two menu options:

- The default option is Commit.

  Press the **Y** key or **RETURN** and DB-Access commits the transactions and closes the database.
- If you want to roll back the transactions, use an arrow key to move the highlight to the Rollback option.

  Press **RETURN**, and DB-Access rolls back the transactions and closes the database.

Important: Select an option carefully. You might commit transactions you do not want if you select Commit, and you will lose any new transactions if you select Rollback.

If you press the Interrupt key, DB-Access displays the DATABASE menu without committing or rolling back the transactions.

The TRANSACTION menu also opens whenever you attempt to open a new database or try to leave the DB-Access menu system without first terminating a transaction. These instances are the only times when you can access the TRANSACTION menu from DB-Access.

Important: If you begin a transaction in an ANSI-compliant database but do not issue a COMMIT statement or ROLLBACK statement, then try to close the database using a non-menu mode, DB-Access will commit the transaction for you. If you do not want to commit the transaction, issue both a ROLLBACK statement and a CLOSE DATABASE statement from the command line.

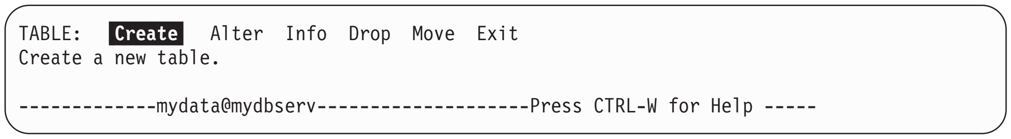
Chapter 5. The Table option

These topics describe how to use the features of the Table option on the main menu. Select this option if you want to perform any of the following table management tasks without SQL programming:

- Create a new table
- Define fragmentation strategy for a new or existing table
- Alter, delete, or display information about an existing table

The TABLE menu

When you press the **T** key or select the Table option from the main menu, the TABLE menu opens, as the following figure shows.



```
TABLE:   Create   Alter  Info  Drop  Move  Exit
Create a new table.

-------------mydata@mydbserv--------------------Press CTRL-W for Help -----
```

Figure 5-1. The TABLE menu

If no current database exists when you select the Table option, the SELECT DATABASE screen opens. Select from a list of databases defined for the current database server or press the Interrupt key to display the main menu.

Important: In the SELECT DATABASE screen, the names of databases are limited to 18 characters. If a database name is longer than 18 characters, you will see the first 17 characters of the name followed by a + sign. Enter the + sign to display the complete long name in **vi**. To exit from **vi**, press **ESC ZZ**.

Use the TABLE menu options as the following table shows.

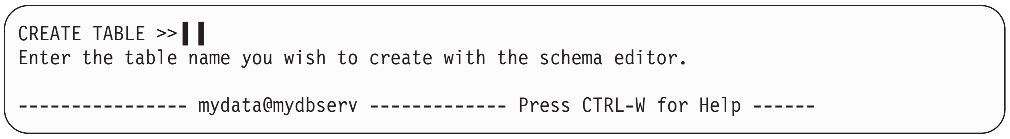
Option	Purpose	Instructions
Create	Enables you to define the structure of a new table	"Create or alter a table"
Alter	Enables you to alter the structure of an existing table	"Create or alter a table"
Info	Displays information about the structure of a table	"Display table information" on page 5-26
Drop	Deletes a table from the database	"Drop a table" on page 5-36
Move	Moves a table from the current database to another database.	
Exit	Returns to the DB-Access main menu	none

Create or alter a table

The Create option on the TABLE menu provides a menu that guides you through the functions of a CREATE TABLE statement. The Alter option provides menus that guide you through the functions of an ALTER TABLE or ALTER FRAGMENT statement.

The CREATE TABLE screen

When you select the Create option on the TABLE menu, the CREATE TABLE screen opens, as the following figure shows.

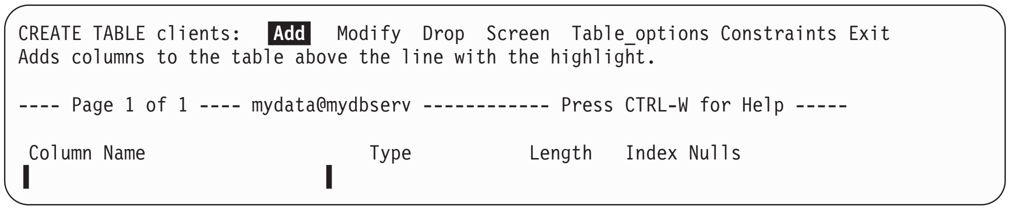


```
CREATE TABLE >>
Enter the table name you wish to create with the schema editor.

---------------- mydata@mydbserv ------------- Press CTRL-W for Help ------
```

Figure 5-2. The CREATE TABLE screen

At the prompt, type the name of the new table and press **RETURN**. You can assign any name to the table if you follow the syntax guidelines for naming database objects described in the *IBM Informix Guide to SQL: Syntax*.

After you enter the new table name, the CREATE TABLE menu opens, as the following figure shows.

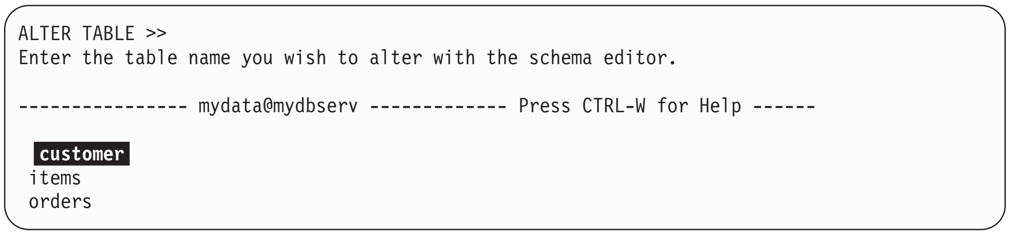


```
CREATE TABLE clients:  Add  Modify  Drop  Screen  Table_options Constraints Exit
Adds columns to the table above the line with the highlight.

---- Page 1 of 1 ---- mydata@mydbserv ------------ Press CTRL-W for Help -----

 Column Name                     Type           Length   Index Nulls
```

Figure 5-3. The CREATE TABLE menu

The ALTER TABLE screen

When you select the Alter option on the TABLE menu, the ALTER TABLE prompt and a list of the tables defined in the current database are displayed, as the following figure shows.



```
ALTER TABLE >>
Enter the table name you wish to alter with the schema editor.

---------------- mydata@mydbserv ------------- Press CTRL-W for Help ------

  customer
 items
 orders
```

Figure 5-4. The ALTER TABLE prompt

Enter the name of the table that you want to alter after the prompt or use the arrow keys to highlight the table name in the list. After you press **RETURN**, the ALTER TABLE menu and the table schema are displayed, as the following figure shows.

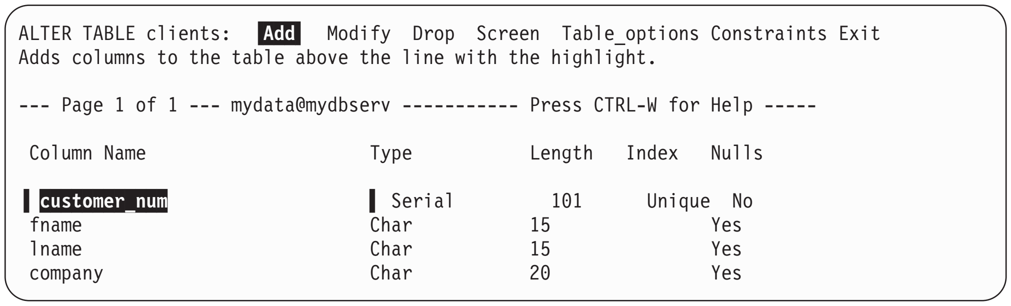

```
ALTER TABLE clients:   Add   Modify  Drop  Screen  Table_options Constraints Exit
Adds columns to the table above the line with the highlight.

--- Page 1 of 1 --- mydata@mydbserv ----------- Press CTRL-W for Help -----

 Column Name                     Type          Length   Index   Nulls

 customer_num                    Serial          101     Unique  No
 fname                           Char          15               Yes
 lname                           Char          15               Yes
 company                         Char          20               Yes
```

Figure 5-5. The ALTER TABLE menu

Important: You must have the Alter privilege to successfully alter a table. Without the privilege, you can use the menus from the ALTER TABLE screen, but an error results when you attempt to select Build-new-table from the EXIT screen. For references explaining the Alter privilege and other table-level privileges, see "The TABLE menu" on page 5-1.

To use the LOAD statement to insert data into a table, you must have both Insert and Select privileges for the table. You need the Select privilege because DB-Access must read SELECT information about the columns before inserting data into the table. If you do not have the Select privilege, the LOAD command fails and you get error message -272, "No SELECT permission." If you have the Select but not the Insert privilege, you get error -275, "No INSERT permission."

The Schema Editor

Both the CREATE TABLE and ALTER TABLE menus have the same options, which are described in the following table.

Option	Purpose	Instructions
Add	Displays the Schema Editor, from which you can add a new column to the table	"Add columns to a table (Add option)" on page 5-4
Modify	Displays the columns that you defined with the Add option so that you can modify the column structure before building the table	"Modifying columns (Modify option)" on page 5-8
Drop	Drops an existing column from the table	"Deleting columns (Drop option)" on page 5-9
Screen	Displays the next screen of column definitions in the Schema Editor	none
Table_options	Enables you to display and select storage spaces for a new table. Displays choices from which to set a fragmentation strategy for a new table. Enables you to set extent sizes and lock mode for a new table. Adds or deletes rowids for an existing fragmented table.	"Arrange storage and locking (Table_options)" on page 5-9
Constraints	Enables you to define primary-key, foreign-key, check, and unique constraints, and to set default column values	"Define constraints" on page 5-19 "Define default values" on page 5-24

Option	Purpose	Instructions
Exit	Builds, rebuilds, or discards the schema and structure that you specified with the other options, and then returns to the TABLE menu	"Build the table (Exit option)" on page 5-8

Important: You must use the **SPACEBAR** to move between menu options, because the arrow keys control cursor movement in the Schema Editor.

To leave the CREATE TABLE menu or ALTER TABLE menu and return to the TABLE menu without building or altering a table, press the Interrupt key.

Add columns to a table (Add option)

To add a new column to a table, you define a new line in the Schema Editor that is displayed below the dashed line. When you create or alter the columns of a table, the Schema Editor issues prompts to assist you.

Important: Before you use the Add option from the ALTER TABLE menu, you must position the highlight in the Schema Editor to indicate where you want to insert the new column or columns. To move the highlight within the displayed columns, use the up and down arrow keys. To scroll more of the column list onto the screen, use the Screen option on the menu. When you select the Add option, the highlighted line moves down to make an empty line for the new column.

The Schema Editor progresses from left to right, completing one horizontal line of description for each column, with the name of the column at the left. Use the right arrow key to move the highlight to each field. To accept the default entry for each field, press **RETURN** or an arrow key.

As you finish one column, the cursor moves to the next line down, so that you can type another column name. Thus, the columns that make up the table are listed vertically.

You can change or bypass any field entry in a line before you move to the next line in either of the following ways:

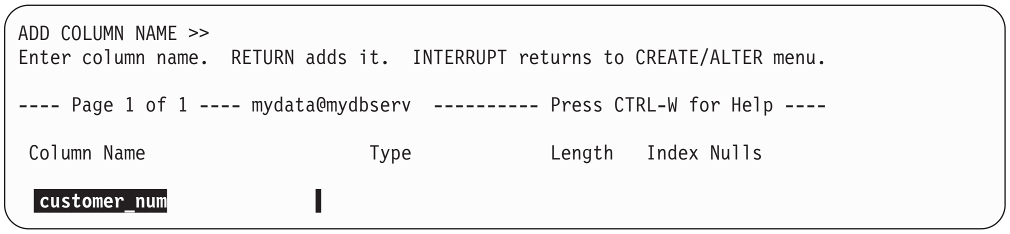
- Use the left arrow key to move back to a field that you passed on the current line.
- Press the Interrupt key to cancel a prompt without inserting a value into the current (highlighted) field.

After you move the cursor to another line, you must use the Modify option on the CREATE TABLE menu to change your entry, as "Modifying columns (Modify option)" on page 5-8 describes.

Column name

The Add option on the CREATE TABLE (or ALTER TABLE) menu places the cursor on an empty line and displays the ADD COLUMN NAME prompt. Type the name of the column after the ADD COLUMN NAME prompt and press **RETURN**. You can assign any name if you follow the identifier syntax guidelines described in the *IBM Informix Guide to SQL: Syntax*.

DB-Access enters the specified name under Column Name, as the following figure shows.

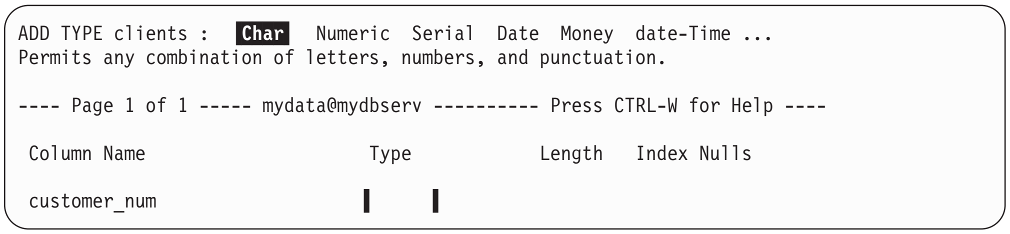


```
ADD COLUMN NAME >>
Enter column name.  RETURN adds it.  INTERRUPT returns to CREATE/ALTER menu.

---- Page 1 of 1 ---- mydata@mydbserv  ---------- Press CTRL-W for Help ----

 Column Name                    Type            Length   Index Nulls

 customer_num
```

Figure 5-6. The ADD COLUMN NAME screen with column name entered

Column data type

The ADD TYPE menu displays various data types, as the following figure shows.



```
ADD TYPE clients :  Char  Numeric  Serial  Date  Money  date-Time ...
Permits any combination of letters, numbers, and punctuation.

---- Page 1 of 1 ----- mydata@mydbserv ---------- Press CTRL-W for Help ----

 Column Name                    Type           Length   Index Nulls

 customer_num
```

Figure 5-7. The ADD TYPE menu for defining column data types

To select the data type for the column, type the first capitalized letter of the data type, using either uppercase or lowercase letters or the **SPACEBAR** to highlight it and then press **RETURN**.

Important: Use the **SPACEBAR** to move to your choice. Use the arrow keys to control cursor movement in the lower part of the screen.

The CREATE TABLE menu provides options for built-in data types. To define a column with one of the extended data types, such as smart large objects, user-defined (opaque) data types, or a collection data type, use the SQL menu to enter and run a CREATE TABLE statement.

If you select one of the following data type categories from the ADD TYPE menu, DB-Access displays one or two submenus for that category.

ADD TYPE category	Data type submenu	Additional submenu
Numeric	Integer	
	Smallint	
	Decimal	
	Float	Smallfloat or Float
Char	Char (press **C** to select)	
	Nchar (press **N** to select)	
Variable-length	Varchar	Varchar (press **V** to select)
		Nvarchar (press **N** to select)
	Text or Byte	Table
		Blobspace

Tip: Although some data types described above are not included in the menu mode, you can use any data types in interactive, non-menu mode.

Locale character data:

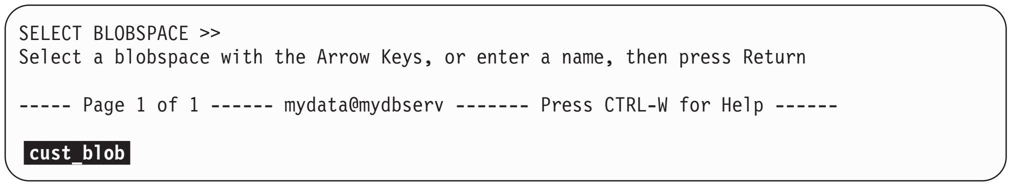
If you use character data in a default locale, select Char for fixed-length data or Varchar if the table will have varying-length entries in that column.

If you use a nondefault locale, select Nchar for fixed length or Nvarchar for varying length.

Large object storage location:

If you select VARIABLE-LENGTH TEXT or BYTE data type, perform one of the following actions to indicate where that large-object data should be:
- Select Table to store the full data directly in the column.
- Select Blobspace to store the actual TEXT or BYTE data in a large-object space (blobspace). The table column then holds the blobspace location.

  DB-Access displays the SELECT BLOBSPACE screen, as the following figure shows. Use the arrow keys to choose a blobspace from the alphabetical list or type the blobspace name at the top of the screen.



```
SELECT BLOBSPACE >>
Select a blobspace with the Arrow Keys, or enter a name, then press Return

----- Page 1 of 1 ------ mydata@mydbserv ------- Press CTRL-W for Help ------

 cust_blob
```

Figure 5-8. The SELECT BLOBSPACE screen for storing variable-length data

Data length or range

If you select any of the following data types for the column, a new ADD screen opens. Enter the appropriate information in the Length field.

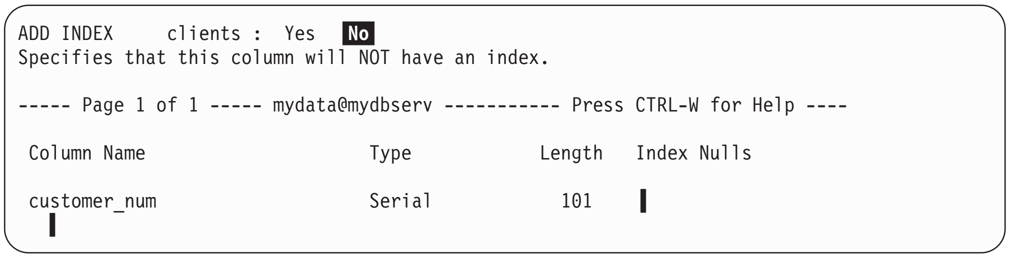
Data type	Length or range
Char	Enter length (the default is 20).
Nchar	Enter length (the default is 20).
Numeric	For the fixed-point form of the DECIMAL type, enter the precision and scale (the default is 16, 2). For the floating-point form of the DECIMAL type, enter the precision only.
Serial	Enter the starting number (the default is 1).
Money	Specify a length (the default is 16, 2).
Datetime	Specify first to last datetime qualifiers.
Interval	Specify first to last interval qualifiers.
Varchar	Specify a maximum length (from 1 to 255 bytes) and a minimum space (from 0 to 255 bytes).

Nvarchar

Specify a maximum length (from 1 to 255 bytes) and a minimum space (from 0 to 255 bytes).

Column index

DB-Access can construct only a nonclustered, ascending B-tree column index. Select the Yes option to create this type of index with the ADD INDEX menu, as the following figure shows.



```
ADD INDEX     clients :  Yes  No
Specifies that this column will NOT have an index.

----- Page 1 of 1 ----- mydata@mydbserv ----------- Press CTRL-W for Help ----

 Column Name                     Type            Length   Index Nulls

 customer_num                    Serial            101
```

Figure 5-9. The ADD INDEX menu

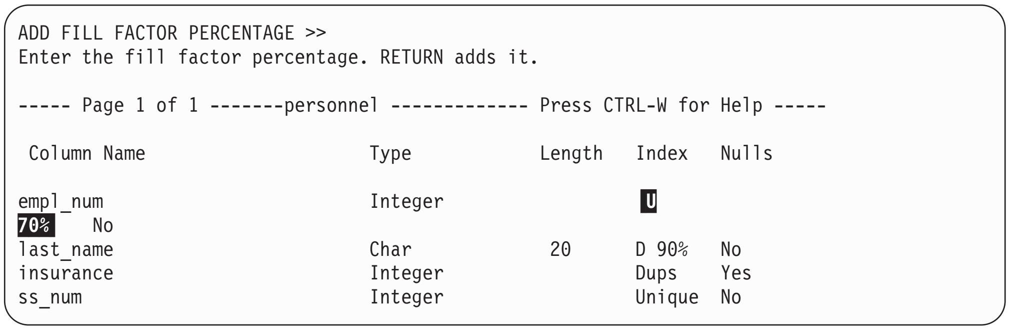
DB-Access displays an ADD DUPLICATES screen. Press **RETURN** or the **Y** key to allow duplicate values, and the word Dups shows in the Index field. Press the **N** key to prevent duplicate values. The word Unique shows in the Index field.

DB-Access displays the ADD FILL FACTOR PERCENTAGE screen, as Figure 5-10 shows.

If you do not want to index the values in this column or if you want any other type of index, such as an R-tree index, select the No option. You must create an R-tree index directly with SQL.

Column index fill factor

Use the ADD FILL FACTOR PERCENTAGE screen, as the following figure shows, to set the fill-factor percentage when you create an index on a single column. The index column has fill factor and Unique or Dups abbreviated to U or D.



```
ADD FILL FACTOR PERCENTAGE >>
Enter the fill factor percentage. RETURN adds it.

----- Page 1 of 1 -------personnel ------------- Press CTRL-W for Help -----

 Column Name                     Type            Length   Index   Nulls

empl_num                         Integer                  U
70%    No
last_name                        Char             20      D 90%   No
insurance                        Integer                  Dups    Yes
ss_num                           Integer                  Unique  No
```

Figure 5-10. The ADD FILL FACTOR PERCENTAGE screen

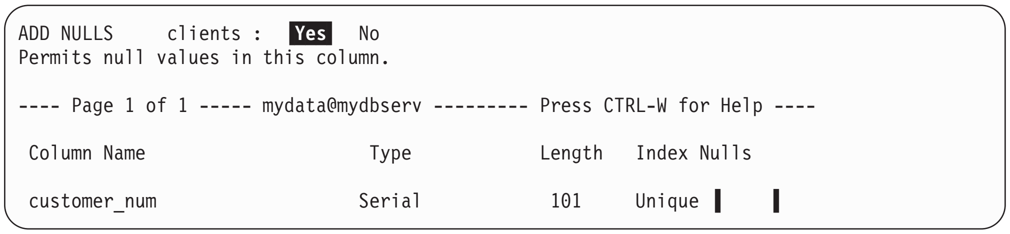
Restriction: You can only set a fill-factor value when you *create* a new index. You can modify the fill factor through the Modify option on the CREATE TABLE menu. However, you cannot alter it through the ALTER TABLE menu after the table for the index is created.

Enter any positive value to a maximum of 100. A value less than 1 or greater than 100 results in an error.

If you press **RETURN** without entering a value, the index will have the fill-factor percentage set in the database server `onconfig` file. If `onconfig` has no fill-factor setting, the index will have the default fill-factor value of 90 percent.

Null value permission

Specify whether the column allows null values on the ADD NULLS menu, as the following figure shows.



```
ADD NULLS     clients :   Yes   No
Permits null values in this column.

---- Page 1 of 1 ----- mydata@mydbserv --------- Press CTRL-W for Help ----

 Column Name                     Type            Length   Index Nulls

 customer_num                   Serial            101     Unique ▌    ▌
```

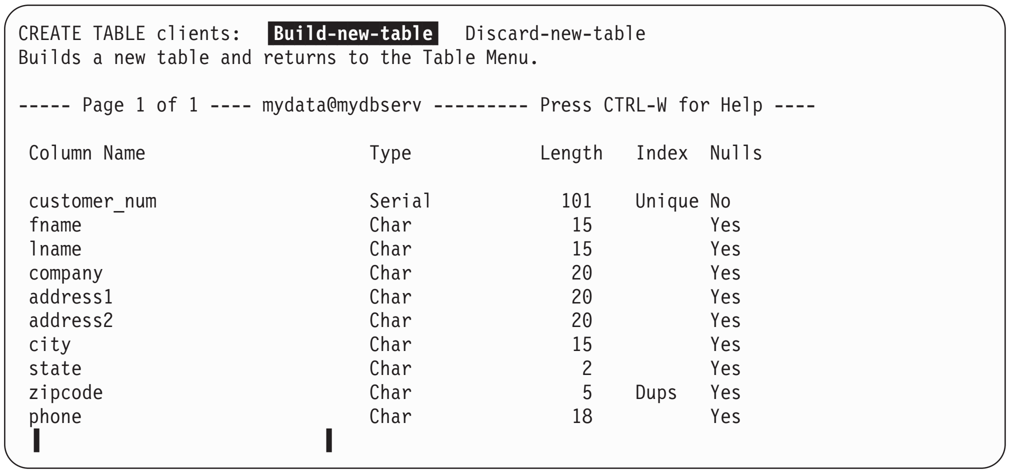
Figure 5-11. The ADD NULLS menu

Select Yes to allow null values in the column or No to force the column to always have a non-null value.

To add another column definition to the table or return to the CREATE TABLE menu, press Interrupt, an arrow key, or **RETURN**.

Build the table (Exit option)

When you complete the schema for the new (or modified) table, select the Exit option on the CREATE TABLE (or ALTER TABLE) menu. DB-Access displays the menu, as the following figure shows.



```
CREATE TABLE clients:   Build-new-table   Discard-new-table
Builds a new table and returns to the Table Menu.

----- Page 1 of 1 ---- mydata@mydbserv --------- Press CTRL-W for Help ----

 Column Name                     Type            Length   Index  Nulls

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

Figure 5-12. The CREATE TABLE menu

To create the table that contains the displayed columns and return to the TABLE menu, select Build-new-table. To return to the TABLE menu without saving the new or modified table definition, select Discard-new-table.

Modifying columns (Modify option)

To modify an existing column, perform the following steps:

1. Use the arrow keys to highlight the column definition that you want to modify.

2. If necessary, select Screen from the CREATE TABLE menu to display the next screen of column definitions in the Schema Editor.
3. Select the Modify option on the CREATE TABLE (or ALTER TABLE) menu and press **RETURN**.
4. Move the highlight to the field you want to modify.

   DB-Access prompts appropriately for the field where the highlight is located. Each of these prompt screens works like the corresponding ADD screen.

   For instructions on specific prompts, see the following table.

Prompt	**Instructions**
MODIFY COLUMN NAME	"Column name" on page 5-4
MODIFY TYPE	"Column data type" on page 5-5
MODIFY LENGTH	"Data length or range" on page 5-6
MODIFY INDEX	"Column index" on page 5-7
MODIFY NULLS	"Null value permission" on page 5-8

5. Move the highlight to the next field and repeat the process.
6. Select Exit to leave the screen after you have elected to build the table or discard the schema, as Figure 5-12 on page 5-8 shows.

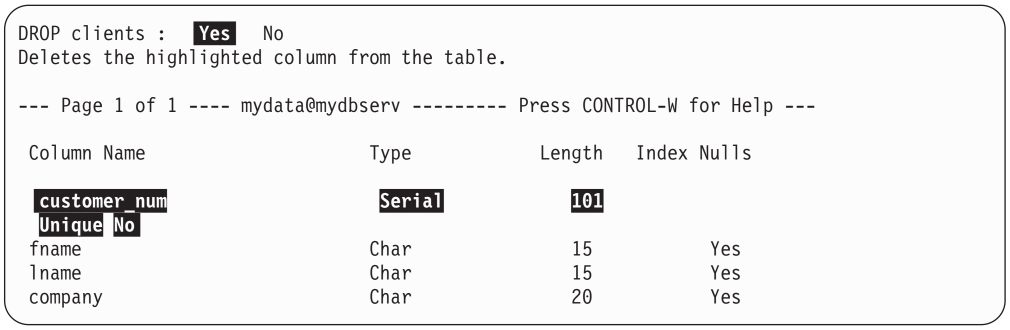
To leave a Modify screen or menu without making any changes, press the Interrupt key at any time.

Deleting columns (Drop option)

To delete a column from a table schema, perform the following steps:
1. Position the highlight anywhere on the column that you want to delete.
2. Select the Drop option on the CREATE TABLE (or ALTER TABLE) menu.

   The column line is then partially or completely highlighted on the screen.

DB-Access displays the DROP menu that prompts you to verify your decision, as the following figure shows.



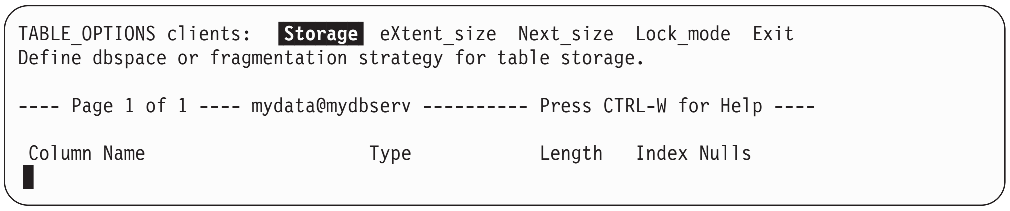
```
DROP clients :  Yes   No
Deletes the highlighted column from the table.

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

 Column Name                     Type            Length   Index Nulls

  customer_num                    Serial           101
  Unique No
 fname                           Char              15           Yes
 lname                           Char              15           Yes
 company                         Char              20           Yes
```

Figure 5-13. The DROP menu

Select **Yes** from the DROP menu to delete the line currently highlighted in the Schema Editor; select **No** to keep the line.

Arrange storage and locking (Table_options)

To display the TABLE_OPTIONS menu, as the following figure shows, select Table_options from the CREATE TABLE menu (or ALTER TABLE menu). You can then specify storage-management parameters, such as location and distribution of data on the storage media.



```
TABLE_OPTIONS clients:  Storage  eXtent_size  Next_size  Lock_mode  Exit
Define dbspace or fragmentation strategy for table storage.

---- Page 1 of 1 ---- mydata@mydbserv ---------- Press CTRL-W for Help ----

 Column Name                     Type            Length   Index Nulls
█
```

Figure 5-14. The TABLE_OPTIONS menu

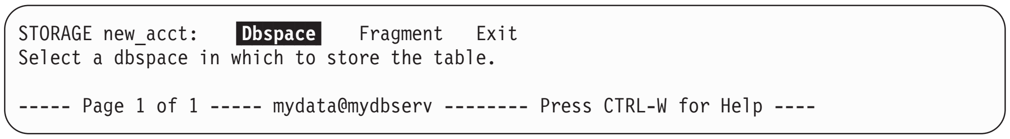
The TABLE_OPTIONS menu contains the following options.

Option	Purpose	Instructions
Storage	Displays dbspaces and enables you to assign a dbspace to the current table Enables you to define a fragmentation strategy for the current table	"Select dbspaces" "Fragmenting a new table" on page 5-11
eXtent_size	Enables you to specify the initial extent size of the table	"Set the extent size" on page 5-17
Next_size	Enables you to specify the next extent size	"Set the extent size" on page 5-17
Lock_mode	Enables you to select either Page or Row as the lock mode	"Determine the lock mode" on page 5-18
Exit	Returns to the previous menu	none

For assistance in setting Table_options values such as dbspaces, fragmentation strategy, extent sizes, and lock mode, see your *IBM Informix Administrator's Guide* and *IBM Informix Performance Guide*.

Select dbspaces

To display the STORAGE menu, as the following figure shows, select the Storage option from the TABLE_OPTIONS menu.



```
STORAGE new_acct:    Dbspace    Fragment   Exit
Select a dbspace in which to store the table.

----- Page 1 of 1 ----- mydata@mydbserv -------- Press CTRL-W for Help ----
```

Figure 5-15. The STORAGE menu for storing table data and defining fragmentation strategy

To display the SELECT DBSPACE screen, as the following figure shows, select Dbspace from the STORAGE menu. Use the arrow keys to highlight a dbspace from the list of dbspaces in the current database and then press **RETURN**.

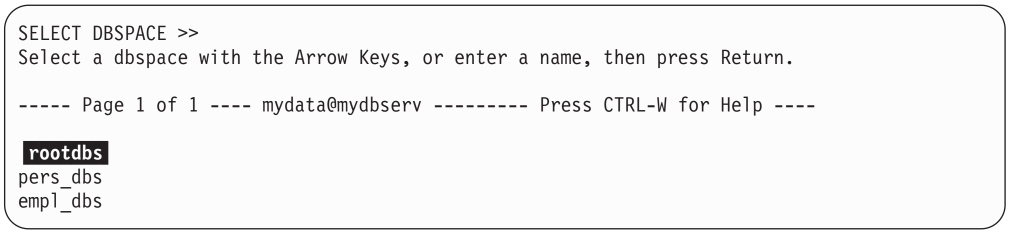

```
SELECT DBSPACE >>
Select a dbspace with the Arrow Keys, or enter a name, then press Return.

----- Page 1 of 1 ---- mydata@mydbserv --------- Press CTRL-W for Help ----

 rootdbs
pers_dbs
empl_dbs
```

Figure 5-16. The SELECT DBSPACE screen for specifying table storage

Select Fragment to set up fragmentation strategy in a series of additional menus. For instructions, see “Fragmenting a new table” or “Alter fragmentation for an existing table” on page 5-14.

Fragmenting a new table

To arrange fragmentation for a new table, display the FRAGMENT menu, as the following figure shows.

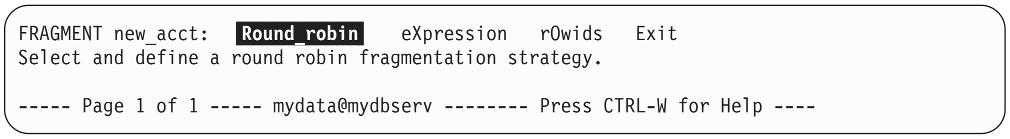

```
FRAGMENT new_acct:  Round_robin   eXpression   rOwids   Exit
Select and define a round robin fragmentation strategy.

----- Page 1 of 1 ----- mydata@mydbserv -------- Press CTRL-W for Help ----
```

Figure 5-17. The FRAGMENT menu for defining fragmentation strategy

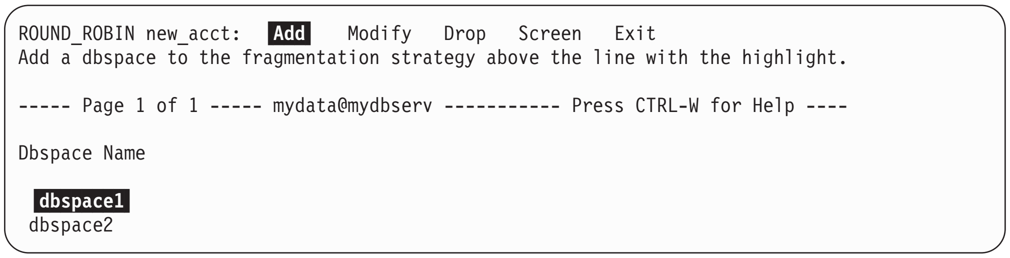
You reach this menu through the following steps:
1. On the main menu, select Table.
2. On the TABLE menu, select Create.
3. On the CREATE TABLE, select Table_options.
4. On the TABLE_OPTIONS menu, select Storage.
5. On the STORAGE menu, select Fragment.

Select the strategy that you want from the FRAGMENT menu, as the following table shows.

Option	Purpose	Instructions
Round_robin	Selects a round-robin strategy for fragmentation	Figure 5-30 on page 5-17
eXpression	Selects an expression strategy for fragmentation	“Expression strategy setup” on page 5-12
rOwids	Adds a column that contains rowids to a fragmented table (The database server does not automatically assign rowids when you insert rows in a fragmented table.)	Press the **o** key to explicitly enable access by rowid
Exit	Exits the FRAGMENT menu and returns to the STORAGE menu	Figure 5-15 on page 5-10

DB-Access has no FRAGMENT menu option for hash or hybrid fragmentation. If you want this type of strategy, use the SQL menu to enter and run the CREATE TABLE or ALTER TABLE statement.

Round-robin setup: To display the ROUND_ROBIN menu, as the following figure shows, select the Round_robin option on the FRAGMENT menu.



```
ROUND_ROBIN new_acct:  Add   Modify   Drop   Screen   Exit
Add a dbspace to the fragmentation strategy above the line with the highlight.

----- Page 1 of 1 ----- mydata@mydbserv ---------- Press CTRL-W for Help ----

Dbspace Name

 dbspace1
 dbspace2
```

Figure 5-18. The ROUND_ROBIN menu for selecting fragment storage spaces

The ROUND_ROBIN menu has the following options.

Add Displays dbspaces so that you can add a new dbspace to the round-robin fragment space assigned to the current table

Modify
Enables you to redefine the fragmentation strategy for the highlighted dbspace

Drop Deletes the highlighted dbspace from the existing strategy, but does *not* delete the dbspace from the database server

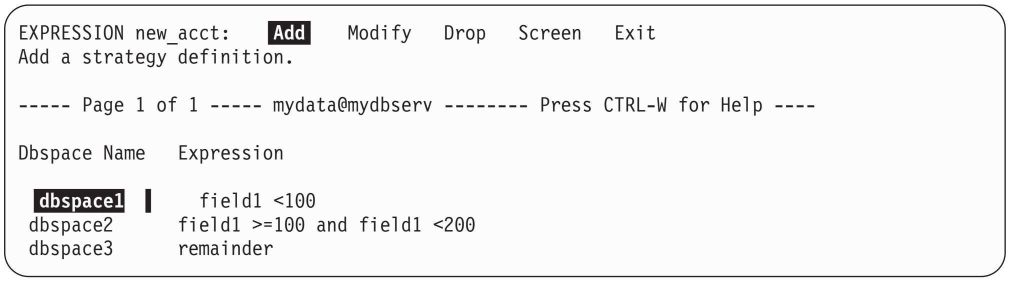
Screen
Scrolls the screen to display more of the available dbspaces

Exit Returns to the FRAGMENT menu

Use the arrow keys to highlight a dbspace from the list on the SELECT DBSPACE screen, as Figure 5-16 on page 5-11 shows, and press **RETURN**. If you try to add a dbspace that is already part of another strategy, an error message is displayed.

When you return to the ROUND_ROBIN menu, the screen displays all dbspaces currently chosen for the strategy.

Expression strategy setup: To display the EXPRESSION menu, as the following figure shows, select the eXpression option on the FRAGMENT menu.



```
EXPRESSION new_acct:   Add   Modify   Drop   Screen   Exit
Add a strategy definition.

----- Page 1 of 1 ----- mydata@mydbserv -------- Press CTRL-W for Help ----

Dbspace Name   Expression

 dbspace1  |    field1 <100
 dbspace2      field1 >=100 and field1 <200
 dbspace3      remainder
```

Figure 5-19. The EXPRESSION menu for defining expression fragmentation strategy

The EXPRESSION menu has the following options.

Add Adds a new dbspace to those that will contain fragments of this table, according to the expression fragmentation strategy

Modify
Modifies the dbspace or expression associated with that dbspace

Drop Deletes the highlighted dbspace and expression from the existing strategy, but does *not* delete the dbspace from the database server

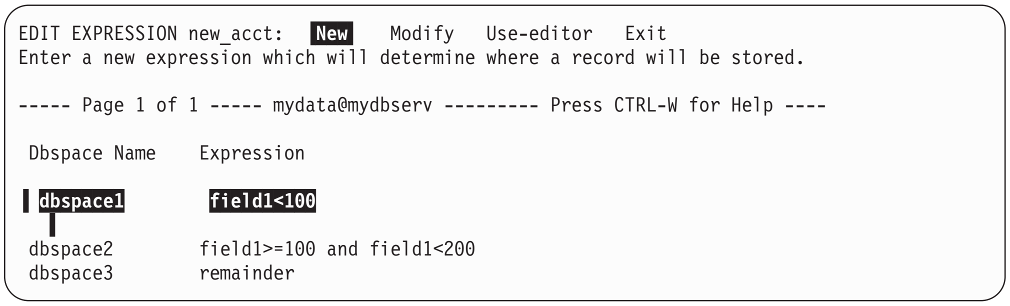
Screen
Scrolls the screen to display more of the available dbspaces

Exit Returns to the FRAGMENT menu

Restriction: DB-Access does not perform data validation on the expression of the strategy.

The Add option on the EXPRESSION menu displays the SELECT DBSPACE screen, as Figure 5-16 on page 5-11 shows. Use the arrow keys to highlight a dbspace from the list and press **RETURN**. If you try to add a dbspace that is already part of another strategy, an error message is displayed.

After you select a dbspace, the EDIT EXPRESSION menu opens, as the following figure shows. From this menu you can enter an expression that determines whether a particular record belongs in the highlighted dbspace.



```
EDIT EXPRESSION new_acct:  New   Modify  Use-editor  Exit
Enter a new expression which will determine where a record will be stored.

----- Page 1 of 1 ----- mydata@mydbserv --------- Press CTRL-W for Help ----

 Dbspace Name    Expression

 dbspace1        field1<100

 dbspace2        field1>=100 and field1<200
 dbspace3        remainder
```

Figure 5-20. The EDIT EXPRESSION menu for defining and editing expressions

The EDIT EXPRESSION menu has the following options.

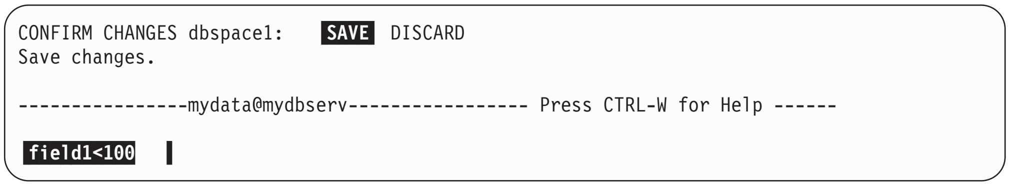
New Displays the blank SQL editor screen so that you can enter a new expression

Modify
Displays the current expression on the SQL editor screen so that you can modify the expression

Use-editor
Displays the current expression in the system editor so that you can modify the expression

Exit Returns to the EXPRESSION menu

After you exit the editor, DB-Access displays the CONFIRM CHANGES menu, as the following figure shows.



```
CONFIRM CHANGES dbspace1:   SAVE  DISCARD
Save changes.

---------------mydata@mydbserv---------------- Press CTRL-W for Help ------

 field1<100
```

Figure 5-21. The CONFIRM CHANGES menu

To save the edits to the expression, press **RETURN**. To discard the edits to the expression, select DISCARD. You return to the EXPRESSION menu.

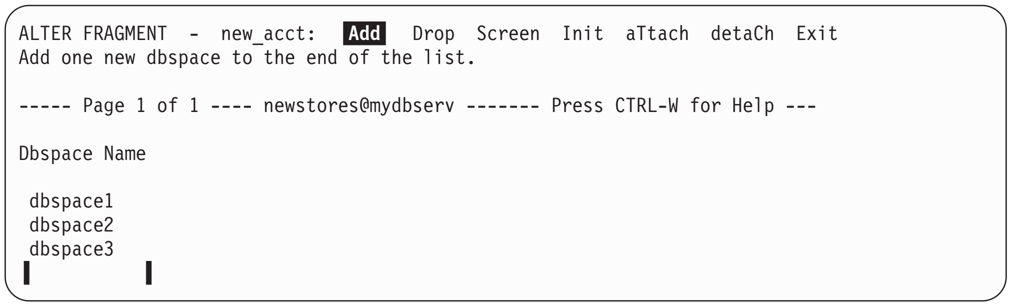
Alter fragmentation for an existing table

If you are altering a table, you arrange or remove fragmentation with the ALTER FRAGMENT menu.

To reach the ALTER FRAGMENT menu:

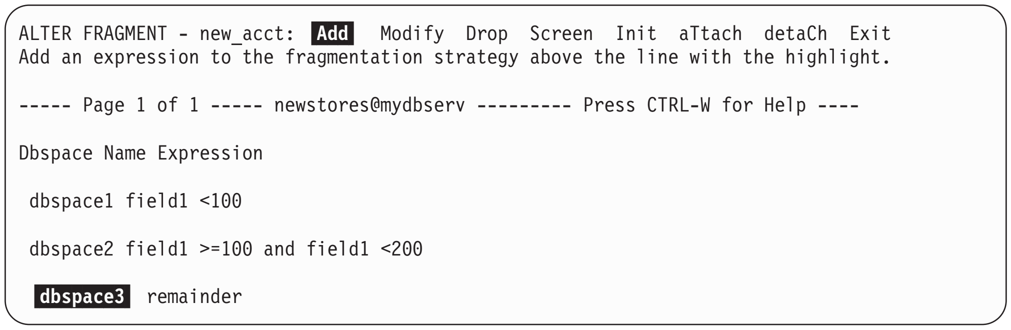
1. On the main menu, select Table.
2. On the TABLE menu, select Alter.
3. Select the Table_options option from the ALTER TABLE menu.
4. Select the Storage option from the TABLE_OPTIONS menu.
5. Select the Fragment option from the STORAGE menu.

The first figure shows how the ALTER FRAGMENT menu displays a table with a round-robin fragmentation strategy. The second figure shows how the ALTER FRAGMENT menu displays a table with an expression-based fragmentation strategy.



```
ALTER FRAGMENT  -  new_acct:  Add  Drop  Screen  Init  aTtach  detaCh  Exit
Add one new dbspace to the end of the list.

----- Page 1 of 1 ---- newstores@mydbserv ------- Press CTRL-W for Help ---

Dbspace Name

 dbspace1
 dbspace2
 dbspace3
```

Figure 5-22. The ALTER FRAGMENT menu for round-robin fragmentation strategy



```
ALTER FRAGMENT - new_acct: Add  Modify  Drop  Screen  Init  aTtach  detaCh  Exit
Add an expression to the fragmentation strategy above the line with the highlight.

----- Page 1 of 1 ----- newstores@mydbserv --------- Press CTRL-W for Help ----

Dbspace Name Expression

 dbspace1 field1 <100

 dbspace2 field1 >=100 and field1 <200

  dbspace3  remainder
```

Figure 5-23. The ALTER FRAGMENT menu for expression fragmentation strategy

The ALTER FRAGMENT menu has the following options.

Option	Purpose	Instructions
Add	Adds a dbspace to the round-robin scheme	"Round-robin setup" on page 5-12 or "Expression strategy setup" on page 5-12, depending on the fragmentation type of the current table
Drop	Drops one dbspace from those used for fragments of the current table	
Screen	Scrolls more dbspaces onto the screen	none

Option	Purpose	Instructions
Init	Describes the fragmentation scheme (if any) of an existing table and enables you to change the strategy, attach fragments, detach fragments, or remove fragmentation	"Fragment an existing table" on page 5-17
aTtach	Attaches one or more tables to the current table, modifies or deletes a selected attachment	"Attach a dbspace"
detaCh	Detaches records from the current table from a specific dbspace and creates a new table with those records	"Detach a dbspace" on page 5-16
Exit	Returns to the TABLE_OPTIONS menu	none

Important: You can execute only one menu option in an ALTER FRAGMENT menu, and it can be applied to the current strategy only once. For example, you can add only one dbspace to a round-robin strategy, and you cannot delete a dbspace during the same ALTER TABLE session.

Attach a dbspace

Select the aTtach option on the ALTER FRAGMENT expression strategy menu, and the ATTACH TABLES menu opens. The following figure shows the ATTACH TABLES menu for a table that has expression fragmentation. The round-robin version of this screen shows only the table name for each fragment.

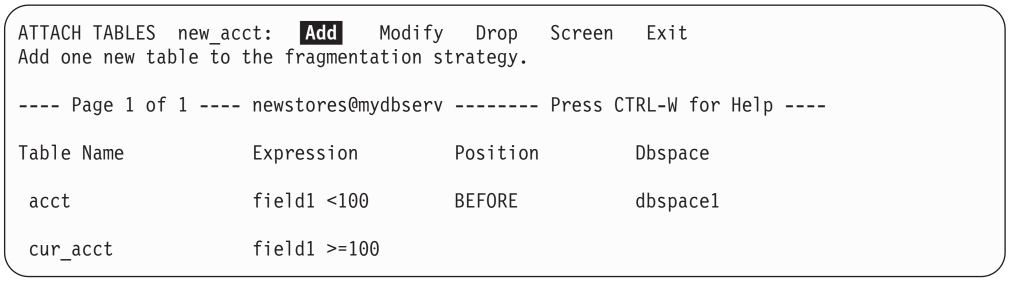

```
ATTACH TABLES  new_acct:  Add   Modify  Drop  Screen  Exit
Add one new table to the fragmentation strategy.

---- Page 1 of 1 ---- newstores@mydbserv -------- Press CTRL-W for Help ----

Table Name          Expression        Position        Dbspace

 acct               field1 <100       BEFORE          dbspace1

 cur_acct           field1 >=100
```

Figure 5-24. The ATTACH TABLES menu for expression fragmentation strategy

Select Add on the ATTACH TABLES menu to begin attaching a fragment. On the SELECT ATTACHING TABLE screen, highlight or type in the table name you want and press **RETURN**, as the following figure shows.

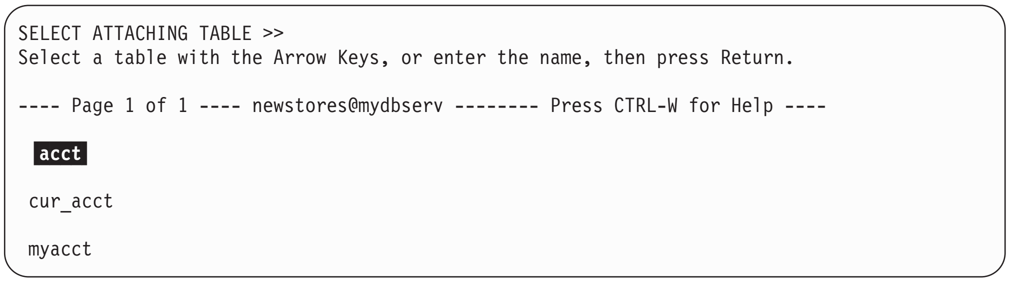
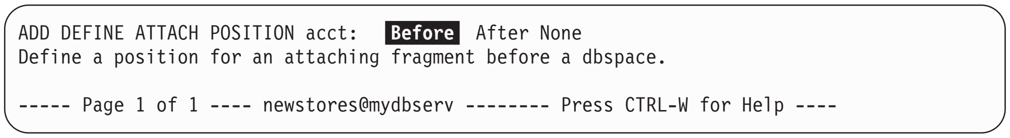

```
SELECT ATTACHING TABLE >>
Select a table with the Arrow Keys, or enter the name, then press Return.

---- Page 1 of 1 ---- newstores@mydbserv -------- Press CTRL-W for Help ----

 acct

 cur_acct

 myacct
```

Figure 5-25. The SELECT ATTACHING TABLE screen

If the table uses round-robin strategy, the resulting new fragment or fragments will be positioned at the end of the fragmentation strategy and the ALTER FRAGMENT menu returns.

If the table uses expression strategy, complete the following steps:

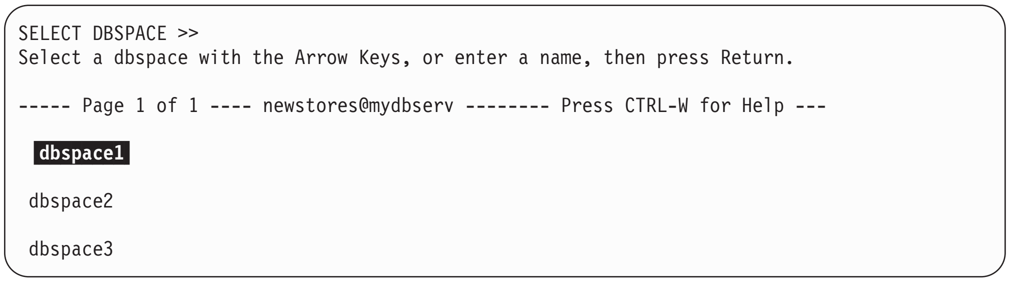
1. After you select a table, the EDIT EXPRESSION menu opens. See Figure 5-20 on page 5-13.
2. When you exit the EDIT EXPRESSION menu, the ADD DEFINE ATTACH POSITION menu opens, as the following figure shows.



```
ADD DEFINE ATTACH POSITION acct:  Before  After None
Define a position for an attaching fragment before a dbspace.

----- Page 1 of 1 ---- newstores@mydbserv -------- Press CTRL-W for Help ----
```

Figure 5-26. The ADD DEFINE ATTACH POSITION menu

   - Select Before to attach the new fragment *before* a dbspace that you select in the next step.
   - Select After to attach the new fragment *after* the dbspace that you select in the next step.
   - Select None to attach the new fragment in the default position.
3. If you select Before or After as the attach position, the SELECT DBSPACE screen opens, as Figure 5-27 shows, listing the dbspaces that the strategy encompasses before attaching the new one.



```
SELECT DBSPACE >>
Select a dbspace with the Arrow Keys, or enter a name, then press Return.

----- Page 1 of 1 ---- newstores@mydbserv -------- Press CTRL-W for Help ---

 dbspace1

 dbspace2

 dbspace3
```

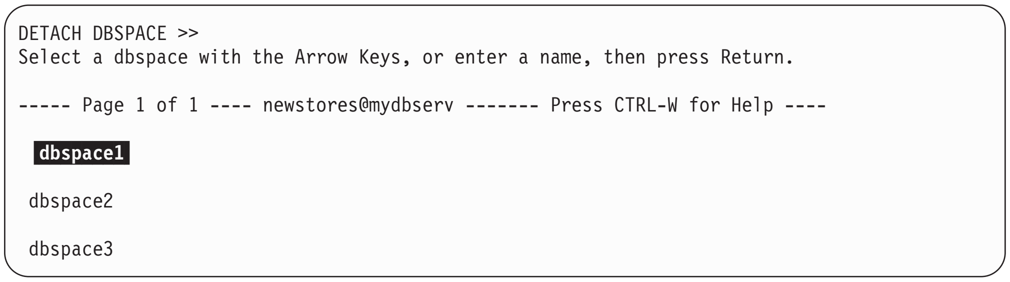
Figure 5-27. The SELECT DBSPACE Screen Listing Fragmented Dbspaces

   Select the dbspace before or after which you want to attach the added fragment.

   The ATTACH TABLES menu reopens, as Figure 5-24 on page 5-15 shows, and shows values for the Position and Dbspace fields.

Detach a dbspace

The detaCh option from the ALTER FRAGMENT menu opens the DETACH DBSPACE screen, as the following figure shows.

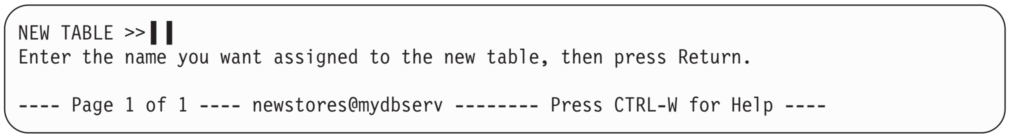


```
DETACH DBSPACE >>
Select a dbspace with the Arrow Keys, or enter a name, then press Return.

----- Page 1 of 1 ---- newstores@mydbserv ------- Press CTRL-W for Help ----

 dbspace1

 dbspace2

 dbspace3
```

Figure 5-28. The DETACH DBSPACE screen for removing fragmentation

Select the dbspace from which you want the records copied into a new, unfragmented table. You can select a dbspace from the list or type in a dbspace name. If you enter an invalid dbspace name, an error message is displayed.

After you correctly enter a dbspace on the DETACH DBSPACE screen, the NEW TABLE screen opens, as the following figure shows.



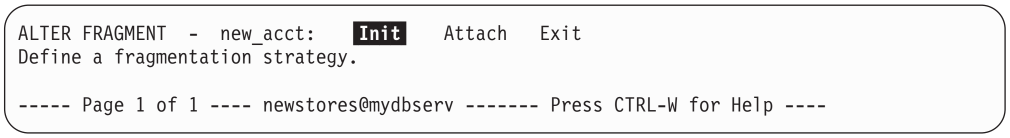
```
NEW TABLE >>
Enter the name you want assigned to the new table, then press Return.

---- Page 1 of 1 ---- newstores@mydbserv -------- Press CTRL-W for Help ----
```

Figure 5-29. The NEW TABLE screen for naming a detached dbspace

Enter the name you want to assign to the new, unfragmented table. This table stores the records from the dbspace you previously selected through the DETACH DBSPACE screen. The display returns to the ALTER FRAGMENT menu.

Fragment an existing table

If a table has no fragmentation strategy when you select the Fragment option on the STORAGE menu, the ALTER FRAGMENT menu opens, as the following figure shows.



```
ALTER FRAGMENT  -  new_acct:    Init    Attach   Exit
Define a fragmentation strategy.

----- Page 1 of 1 ---- newstores@mydbserv ------- Press CTRL-W for Help ----
```

Figure 5-30. The ALTER FRAGMENT menu for no fragmentation strategy

The ALTER FRAGMENT menu has the following options.

Option	Purpose	Instructions
Init	Provides the following options: • fragments a previously unfragmented table • removes fragmentation from a table • changes the fragmentation strategy for a table	The ALTER FRAGMENT-INIT menu has the same options as the FRAGMENT menu under CREATE TABLE. For instructions, see “Fragmenting a new table” on page 5-11.

Attach	Enables you to define a fragmentation strategy and select the tables to fragment with the new strategy. This option has the same effect as the following statement:
	`ALTER FRAGMENT ON TABLE table1...`
	`ATTACH table1, table2`
		The ALTER FRAGMENT-ATTACH menu offers round-robin and expression fragmentation. For instructions on both types of strategy, see “Fragmenting a new table” on page 5-11.

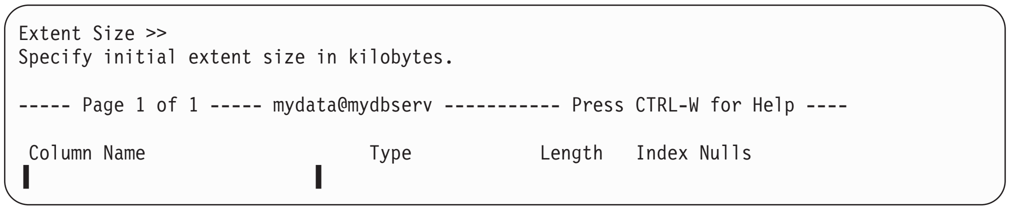
Exit	Returns to the TABLE_OPTIONS menu	none

Restriction: You can perform only one operation during an ALTER FRAGMENT session.

Set the extent size

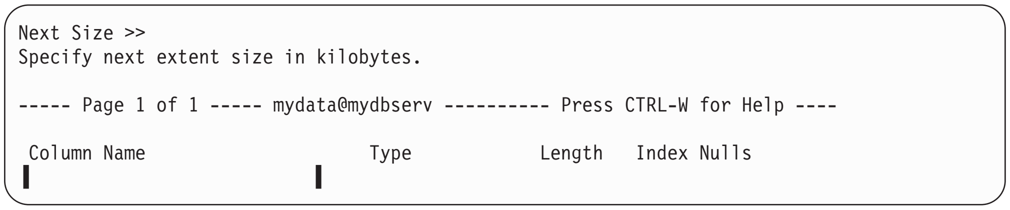
When you create a table, you can specify how much initial disk space, or initial extent size, to reserve for the table. You can also specify the size of additional extents, or next extent spaces, the database server adds if the initial extent becomes full.

To specify an initial extent size, select the eXtent_size option on the TABLE_OPTIONS menu. DB-Access displays the Extent Size screen, as the following figure shows. After you set initial extent size, select the Next_size option from the TABLE_OPTIONS menu to display the Next Size screen, as the second figure shows.



```
Extent Size >>
Specify initial extent size in kilobytes.

----- Page 1 of 1 ----- mydata@mydbserv ----------- Press CTRL-W for Help ----

 Column Name                    Type            Length   Index Nulls
```

Figure 5-31. The (Initial) Extent Size screen



```
Next Size >>
Specify next extent size in kilobytes.

----- Page 1 of 1 ----- mydata@mydbserv ---------- Press CTRL-W for Help ----

 Column Name                    Type            Length   Index Nulls
```

Figure 5-32. The Next (Extent) Size screen

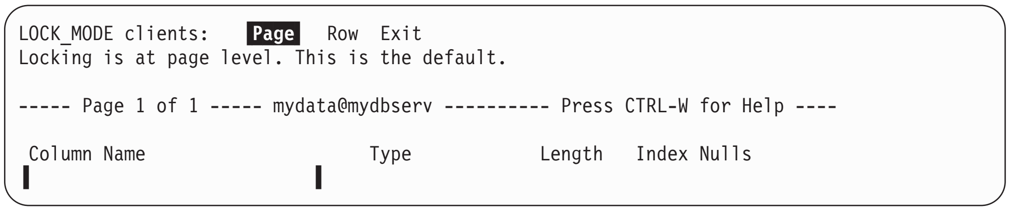
To select extent size on either screen, perform one of the following actions:

- Press **RETURN** to accept the default size of 8 KB.
- Type a number (representing kilobyte units) and press **RETURN**.

  The minimum extent size is 4 KB.

Determine the lock mode

When you select the Lock_mode option on the TABLE_OPTIONS menu, DB-Access displays the LOCK_MODE menu, as the following figure shows.



```
LOCK_MODE clients:    Page  Row  Exit
Locking is at page level. This is the default.

----- Page 1 of 1 ----- mydata@mydbserv ---------- Press CTRL-W for Help ----

 Column Name                    Type            Length   Index Nulls
```

Figure 5-33. The LOCK_MODE menu

The LOCK_MODE menu lets you choose the mode to use when the database locks the rows in a table. The LOCK_MODE menu has the following options.

Page	Locks the entire page on which a row is located
Row	Locks a selected row individually
Exit	Exits to the TABLE_OPTIONS menu

One row of a table is the smallest object that you can lock. A disk page contains one or more rows of a table. To determine if you will enhance performance by locking a disk page rather than individual rows on the page, see your *IBM Informix Performance Guide*. Unless you specify row-level locking before you exit, DB-Access uses the default (Page).

Add or drop rowids

You can add or delete rowids only when you alter an existing table. To reach the ALTER ROWID menu:

- Select Table_options from the ALTER TABLE menu.
- Select Rowids from the TABLE OPTIONS menu.

The ALTER ROWID menu has the following options.

Add Adds a column with rowids to the fragmented table

Drop Discards the rowid column previously added

None Cancels the selection you made on this screen so that you can exit without altering the table

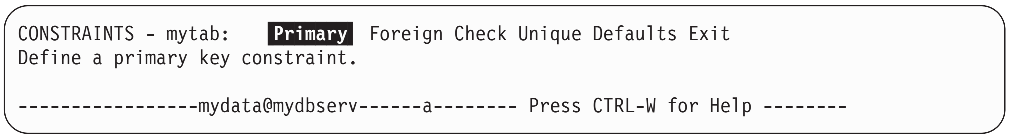
Exit Exits to the TABLE_OPTIONS menu

If you select Add or Drop, another menu prompts you to verify your selection. Select Yes to execute the Add or Drop; select No to cancel the Add or Drop.

Define constraints

You can use the DB-Access Schema Editor to define constraints for columns in a specified table. You can define primary-key or foreign-key constraints, column-level or table-level check constraints, and unique constraints. You can also add or modify a default value for the column when no value is provided when a new row is inserted.

If you select the Constraints option from the CREATE TABLE menu, the CONSTRAINTS menu opens, as the following figure shows.



```
CONSTRAINTS - mytab:    Primary  Foreign Check Unique Defaults Exit
Define a primary key constraint.

-----------------mydata@mydbserv------a-------- Press CTRL-W for Help --------
```

Figure 5-34. The CONSTRAINTS menu

The CONSTRAINTS menu has the following options.

Primary
Lists the columns that make up the table so that you can choose the column or columns that make up the primary key

Foreign
Asserts a foreign-key relationship for a column

Check Enables you to specify valid values for a column and forces the validation of data entry in that column

Unique
Declares that a column must contain a unique value

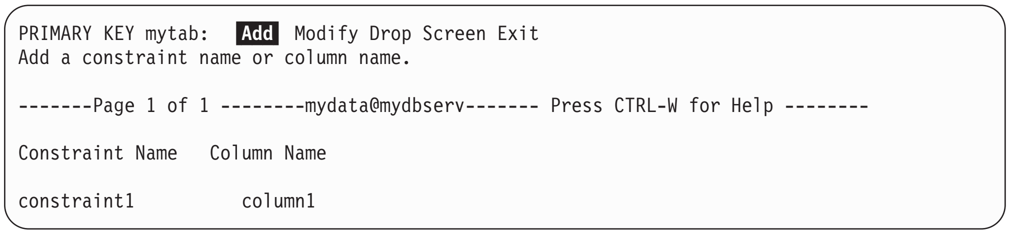
Defaults
Enables you to set a default value for a column

Exit Returns you to the CREATE TABLE menu

Referential integrity is the logical dependency of a foreign key on a primary key. The integrity of a row that contains a foreign key depends on the integrity of the row that it references, which contains the matching primary key. Foreign key and primary key constraints are also called *referential* constraints.

Define primary-key constraints

To add, modify, or delete a primary-key referential constraint for the current table, select the Primary option on the CONSTRAINTS menu. The PRIMARY KEY menu opens, as the following figure shows.



```
PRIMARY KEY mytab:  Add  Modify Drop Screen Exit
Add a constraint name or column name.

-------Page 1 of 1 --------mydata@mydbserv------- Press CTRL-W for Help --------

Constraint Name   Column Name

constraint1          column1
```

Figure 5-35. The PRIMARY KEY menu

When you define a primary key constraint, DB-Access validates your entry by verifying the following information:

- The column name is not repeated.
- No more than 16 column names are displayed in one constraint.
- The column is not assigned the BLOB, CLOB, BYTE or TEXT data type.

Use the menu options as follows:

- Use the Screen option to scroll to any additional constraint descriptions so that you can select one to modify or delete.
- If you select the Add option to add a primary-key referential constraint, the Schema Editor inserts and highlights a new line at the top of the constraint list. If you move your cursor to an existing constraint line and select the Modify option, you can change the contents of the line.
  1. At the ADD (or MODIFY) CONSTRAINT NAME prompt, enter a constraint name.

     If you press **RETURN** in this field without typing a constraint name, the database server assigns a temporary constraint name, such as **unassigned1**, **unassigned2**, and so on. This temporary constraint name exists until you modify it or the table is built or discarded. The database server assigns a permanent constraint name at the time that you create the table.
  2. At the ADD (or MODIFY) COLUMN NAME prompt, enter a column name.
- Select Drop to eliminate the constraint that the highlight indicates. If the highlighted field is Constraint Name, then all columns associated with that constraint are deleted. The lines are removed and any gaps are closed in the display.

Define foreign-key constraints

To create, modify, or delete foreign-key referential constraints on the current table, select the Foreign option on the CONSTRAINTS menu. The FOREIGN KEY menu opens, as the following figure shows.

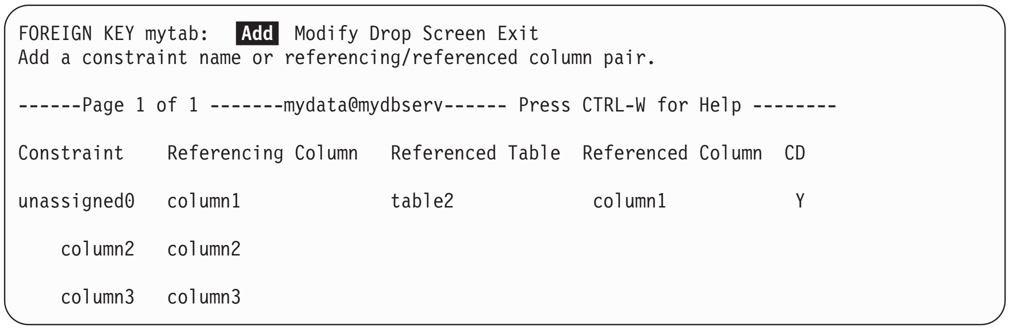

```
FOREIGN KEY mytab:  Add  Modify Drop Screen Exit
Add a constraint name or referencing/referenced column pair.

------Page 1 of 1 -------mydata@mydbserv------ Press CTRL-W for Help --------

Constraint    Referencing Column   Referenced Table  Referenced Column  CD

unassigned0   column1              table2             column1           Y

    column2   column2

    column3   column3
```

Figure 5-36. The FOREIGN KEY menu

Use the menu options as follows:

- Use the Screen option to scroll any additional constraint descriptions onto the screen so that you can select one to modify or delete.
- Select the Add option to specify the constraint name, referenced table, and cascading deletes, or to add one or more referencing and referenced column pairs to the constraint.
  1. At the ADD CONSTRAINT NAME prompt, enter a constraint name. If you press **RETURN** in this field without typing a constraint name, DB-Access assigns a temporary constraint name. The database server assigns a permanent name to the constraint when the constraint is created.
  2. At the ADD COLUMN NAME prompt, enter the name of the *referencing* column. The referencing table is sometimes called the *child* table.
  3. At the ADD REFERENCED TABLE prompt, enter the name of the *referenced table*. The referenced table is sometimes called the *parent* table.
  4. At the second ADD COLUMN NAME prompt, enter the name of the *referenced column*.
  5. When the Cascading Deletes (CD) field is the current field, the ADD ENABLE CASCADING DELETES menu opens, as the following figure shows.

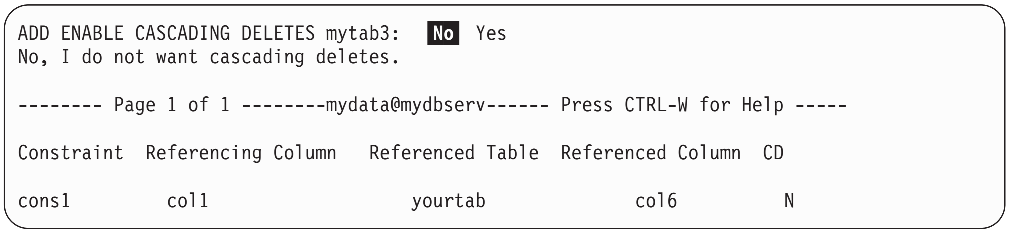

```
ADD ENABLE CASCADING DELETES mytab3:  No  Yes
No, I do not want cascading deletes.

-------- Page 1 of 1 --------mydata@mydbserv------ Press CTRL-W for Help -----

Constraint  Referencing Column   Referenced Table  Referenced Column  CD

cons1        col1                  yourtab            col6          N
```

Figure 5-37. The ADD ENABLE CASCADING DELETES menu

     Select Yes to enable cascading deletes. When you delete a referenced (parent) record, you also delete all corresponding referencing (child) records. (Option Yes is equivalent to the ON DELETE CASCADE option of the REFERENCES clause in the CREATE TABLE statement.)

     Select No to prevent cascading deletes. A referenced (parent) column cannot be deleted if referencing (child) records exist.

     For a detailed description of referential integrity and cascading deletes, see the CREATE TABLE statement in the *IBM Informix Guide to SQL: Syntax*.

  When you complete the Cascading Deletes entry, the cursor returns to the Constraint field. Enter another constraint or press the Interrupt key to return to the top line of the FOREIGN KEY menu.

- Move your cursor to an existing constraint line and select the Modify option to change the contents of the line. You can modify Constraint, Referencing Column, and Referenced Table information.

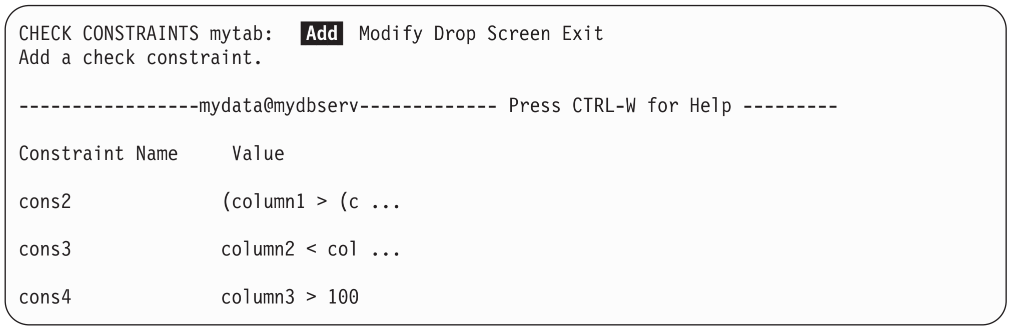
  Change the entry for the field and press **RETURN** to modify it, or press the Interrupt key if you do not want to modify a foreign constraint.
- The Drop option displays a list of foreign-key constraints and prompts you to select one to delete.

  If the current (highlighted) field is Constraint, then the *entire* constraint is deleted. If any other field is highlighted, then only that referenced and referencing pair is deleted.

  The default is Yes. Press **RETURN** to delete the highlighted constraint. Move the cursor to highlight No if you do not want to delete that constraint. You return to the FOREIGN KEY menu.

Define check constraints

The CHECK CONSTRAINTS menu lets you add, modify, or delete a check constraint for the current table. Select the Check option on the CONSTRAINTS menu to access the CHECK CONSTRAINTS menu, as the following figure shows.



```
CHECK CONSTRAINTS mytab:  Add  Modify Drop Screen Exit
Add a check constraint.

-----------------mydata@mydbserv------------- Press CTRL-W for Help ---------

Constraint Name     Value

cons2              (column1 > (c ...

cons3              column2 < col ...

cons4              column3 > 100
```

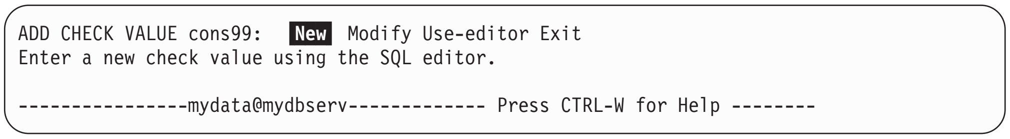
Figure 5-38. The CHECK CONSTRAINTS menu

The CHECK CONSTRAINTS menu displays any previously added check constraints. The first 36 characters of the check value are displayed on the CHECK CONSTRAINTS menu.

Use the CHECK CONSTRAINTS menu options as follows:

- The Drop option lets you delete the current check constraint.

  The default is Yes. Press **RETURN** to delete the highlighted constraint. Move the cursor to highlight No and then press **RETURN** if you do not want to delete that constraint. You return to the CHECK CONSTRAINTS menu.
- Select the Add option to add a check constraint for the table and enter the name and value of the constraint.
  1. At the ADD (or MODIFY) CONSTRAINT NAME prompt, enter a name. If you press RETURN in this field without specifying a value, a temporary constraint name is assigned, and it exists until the table is built or discarded.
  2. From the ADD (or MODIFY) CHECK VALUE menu, as the following figure shows, select the SQL editor or specify another editor to enter the check-constraint value.



```
ADD CHECK VALUE cons99:  New  Modify Use-editor Exit
Enter a new check value using the SQL editor.

----------------mydata@mydbserv------------- Press CTRL-W for Help --------
```

Figure 5-39. The ADD CHECK VALUE menu

The ADD CHECK VALUE menu has the following options.

New Displays the blank SQL editor screen so that you can enter a new check value

Modify
Displays the current check value on the SQL editor screen so that you can modify the value

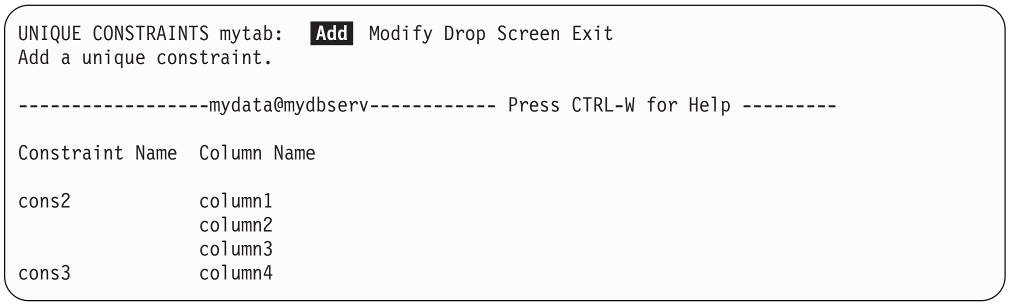
Use-editor
Displays the current check value in the system editor so that you can modify the value

Exit Returns to the CHECK CONSTRAINTS menu

If you break from the ADD CHECK VALUE menu or exit without defining the check value, you return to the CHECK CONSTRAINTS menu. If you defined the check value, you remain in add mode, a new line is inserted, the Constraint Name is the current field, and the ADD CONSTRAINT NAME prompt are displayed.

Define unique constraints

The UNIQUE CONSTRAINTS menu lets you add, modify, or delete a unique constraint for the current table. To access the UNIQUE CONSTRAINTS menu, select the Unique option on the CONSTRAINTS menu, as the following figure shows.



```
UNIQUE CONSTRAINTS mytab:  Add  Modify Drop Screen Exit
Add a unique constraint.

------------------mydata@mydbserv------------ Press CTRL-W for Help ---------

Constraint Name  Column Name

cons2            column1
                 column2
                 column3
cons3            column4
```

Figure 5-40. The UNIQUE CONSTRAINTS menu

Use the UNIQUE CONSTRAINTS menu options as follows:

- Select Screen to display the next screen of unique constraints.
- If you select the Add option, the Schema Editor adds a new line.
  1. At the ADD CONSTRAINT NAME prompt, enter a constraint name. If you press **RETURN** in this field without typing a name, a temporary constraint name is assigned, which exists until the table is built or discarded.
  2. At the ADD COLUMN NAME prompt, enter the name of the column that should have a unique value. A new line is added, and the Column Name is still the current field.
- The Modify option lets you modify either the Constraint Name or Column Name field. If you modify the Constraint Name field, the MODIFY CONSTRAINT NAME prompt is displayed. If you modify the Column Name field, the MODIFY COLUMN NAME prompt is displayed.

- Select the Drop option to delete the constraint or column name where the highlight is located.

  If you delete a constraint name, all column names associated with that constraint name are also deleted.

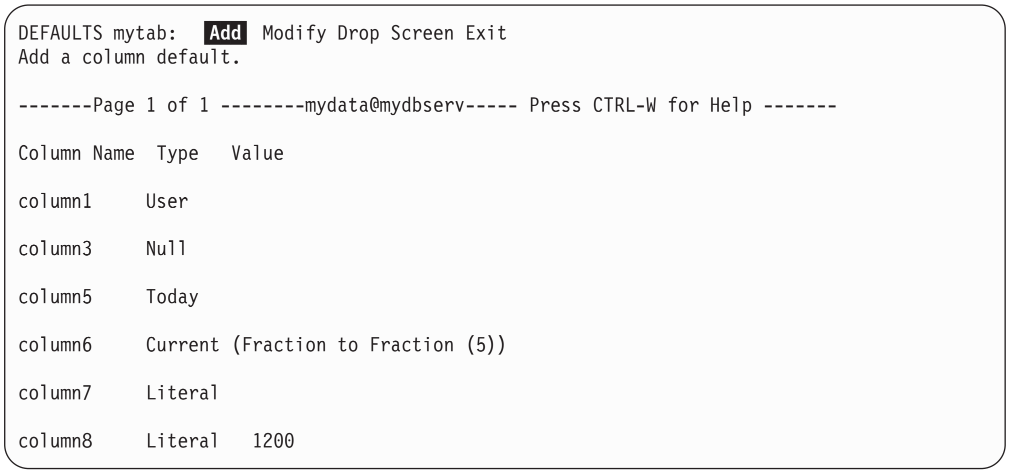
Restriction: You cannot modify unique constraints after you create them. To identify the unique constraints listed on the UNIQUE CONSTRAINTS menu, use an asterisk (*) before the constraint name. If you try to modify a unique constraint using the Modify option in the UNIQUE CONSTRAINTS menu, an error message is displayed.

Data validation: When you enter a unique constraint, DB-Access validates your entry by verifying the following information:

- The column name exists.
- The column name is not repeated.
- No more than 16 column names are present in one constraint.
- The column is not assigned the BYTE or TEXT data type.

Define default values

Use the DEFAULTS menu to define default values for columns in a table, as the following figure shows. To access the DEFAULTS menu, select the Defaults option on the CONSTRAINTS menu.



```
DEFAULTS mytab:  Add  Modify Drop Screen Exit
Add a column default.

-------Page 1 of 1 --------mydata@mydbserv----- Press CTRL-W for Help -------

Column Name  Type   Value

column1     User

column3     Null

column5     Today

column6     Current (Fraction to Fraction (5))

column7     Literal

column8     Literal   1200
```

Figure 5-41. The DEFAULTS menu

The DEFAULTS menu has the following options.

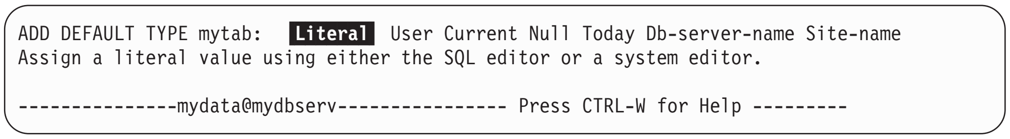
Option	Purpose
Add	Adds a column default value of the appropriate data type
Modify	Lists an existing default name, data type, and value so that you can change the default attributes
Drop	Deletes a column default
Screen	Displays the next screen of defaults
Exit	Returns to the CONSTRAINTS menu

You can see the first 28 characters of the value.

To add a column default

1. Select the Add option.

   The Schema Editor inserts a new line at the top of the list and makes the Column Name the current field.
2. At the ADD COLUMN NAME prompt, enter a value for the column name.
3. From the ADD DEFAULT TYPE menu that the following figure shows, define the default value of a column in the current table.



```
ADD DEFAULT TYPE mytab:  Literal  User Current Null Today Db-server-name Site-name
Assign a literal value using either the SQL editor or a system editor.

---------------mydata@mydbserv---------------- Press CTRL-W for Help ---------
```

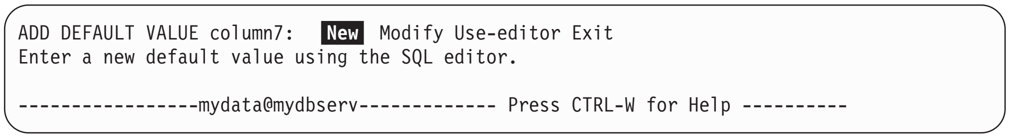
Figure 5-42. The ADD DEFAULT TYPE menu

The ADD DEFAULT TYPE menu has the following options that let you assign default values to the column.

Literal	A literal default value entered either in the SQL editor or a user-specified system editor
User	The login name of the current user
Current	The current system clock time of day
Null	Null
Today	The current system date
Db-server-name	The current database server name
Site-name	The current site name

- If you select User, Null, Today, Db-server-name, or Site-name, the system assigns that value to the default type and returns you to the DEFAULTS menu in add mode.
- If you select Current as the default value, the qualifier is taken from the column definition.
- If you enter a default value and the type is DATETIME or INTERVAL, enter only the value. The qualifier comes from the column definition.
- If you select Literal as the default type, the ADD DEFAULT VALUE menu opens, which lets you assign a literal as the default value.

The ADD DEFAULT VALUE menu lets you add or modify the default value for a column in the current table with either the SQL editor or a system editor, as the following figure shows.



```
ADD DEFAULT VALUE column7:  New  Modify Use-editor Exit
Enter a new default value using the SQL editor.

-----------------mydata@mydbserv------------- Press CTRL-W for Help ----------
```

Figure 5-43. The ADD DEFAULT VALUE menu

The ADD DEFAULT VALUE menu displays the following options.

New	Displays the blank SQL editor screen so that you can enter a new value

Modify
Displays the current default on the SQL editor screen so that you can modify the default value

Use-editor
Displays the current default in the system editor so that you can modify the current value

Exit Returns to the ADD DEFAULT VALUE menu

To modify the column name, type, or value field

1. Select the Modify option to modify the Column Name, Type, or Value field where the highlight is located.
2. If you highlight a value, the MODIFY DEFAULT VALUE menu prompts you to start the SQL editor or the system editor so that you can type over or modify the existing default value.

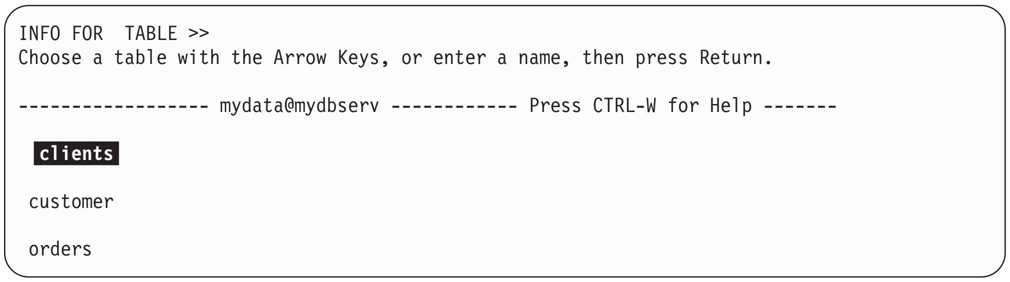
Data validation: When you enter a default value, DB-Access validates your entry. The database server validates the literal value and checks the following information:

- The column name must exist.
- The column type cannot be SERIAL.
- If the column does not allow nulls, you cannot specify the default value as `Null`.
- You can use the default value `Current` only with a DATETIME column type.
- You can use the default value `Db-server-name` only with a column type of CHAR, NCHAR, VARCHAR, or NVARCHAR.
- You can use the default value `Site-name` only with a column type of CHAR, NCHAR, VARCHAR, or NVARCHAR.
- You can use the default value `Today` only with a column type of DATE.
- You can use the default value `User` only with a column type of CHAR, NCHAR, VARCHAR, or NVARCHAR.

Display table information

Use the Info option on the TABLE menu to display information about columns, indexes, access privileges, reference privileges, constraints (referential, check, or unique), column default values, triggers, status, and fragmentation strategy of a table. No options exist to display table owners or information about views.

When you select the Info option on the TABLE menu, the INFO FOR TABLE screen opens, as the following figure shows.



```
INFO FOR  TABLE >>
Choose a table with the Arrow Keys, or enter a name, then press Return.

------------------ mydata@mydbserv ------------ Press CTRL-W for Help -------

 clients

 customer

 orders
```

Figure 5-44. The INFO FOR TABLE screen

This screen lists the names of tables that exist in the current database. Note the following items:

- If you are not the table owner, the table name is prefixed by the owner name, as in **june.clients**.
- If the list of tables does not fit on one screen, the last entry is an ellipsis (...). Use the arrow keys to highlight the ellipsis, and the next page of table names are displayed.
- If Global Language Support is enabled, the list of table names is sorted according to the database collation rules defined when the database was created. Thus, different users using different collating sequences for DB-Access see the table names in the database listed in the same order.

To request information about tables on a different database server, use the format *database@server:table* or *database@server:owner.table* at the prompt. The following example requests information about the **customer** table that **dba** created in the **accounts** database on the database server **topend**:

```
INFO FOR TABLE >> accounts@topend:dba.customer
```

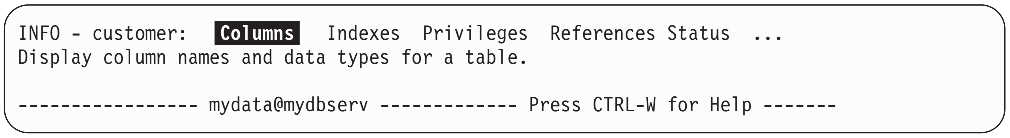
To leave the INFO FOR TABLE screen without requesting table information, press the Interrupt key. You return to the TABLE menu.

You can select a table in one of the following ways:

- Type the table name and press **RETURN**.

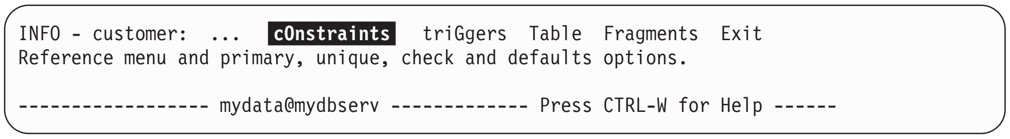
  You must use this method and include the full path name if you want information about a table that is not in the current database.
- Use the arrow keys to highlight the table name that you want and press **RETURN**.

For example, for the **customer** table, type **customer** or use the arrow keys to highlight it and press **RETURN**. The INFO menu opens, with **customer** in the top line, as the following figures show.



```
INFO - customer:  Columns  Indexes  Privileges  References Status  ...
Display column names and data types for a table.

---------------- mydata@mydbserv ------------- Press CTRL-W for Help -------
```

Figure 5-45. The INFO menu for displaying table information (first screen)



```
INFO - customer:  ...  cOnstraints  triGgers  Table  Fragments  Exit
Reference menu and primary, unique, check and defaults options.

----------------- mydata@mydbserv ------------- Press CTRL-W for Help ------
```

Figure 5-46. The INFO menu for displaying table information (second screen)

The INFO menu has the following options.

Option	Purpose	Instructions
Columns	Lists data type by column name and indicates which columns can contain a null value	"Display column information" on page 5-28
Indexes	Describes each index defined for a specified table	"Display index information" on page 5-31
Privileges	Lists the users who have Select, Update, Insert, Delete, Index, or Alter privileges for the specified table	"Display table-level privileges" on page 5-32

Option	Purpose	Instructions
References	Lists the users who have the table-level References privilege for the specified table and the names of the columns they can reference	"Display references privileges" on page 5-32
Status	Lists the table name, owner, row size, number of rows and columns, and creation date of the current table	none
cOnstraints	Displays the referential, primary, unique, and check constraints, and the default values for the columns in the specified table	"Display column constraints and defaults" on page 5-32
triGgers	Displays header and body information for a specified trigger	"Display triggers" on page 5-34
Table	Redisplays the INFO FOR TABLE menu so that you can select a different table for examination	none
Fragments	Lists fragmented dbspaces assigned to the table and, for expression-based fragmentation, displays the expression assigned to each dbspace	"Display fragmentation information" on page 5-35
Exit	Returns to the TABLE menu	none

Tip: From the CREATE TABLE menu, use Table-options to view extent and lock mode information, or issue a SELECT statement to list the table description in the **systables** system catalog table.

Display column information

Use the Columns option on the INFO menu to display the following information for each column of the specified table: the name of the column, the data type of the column, and whether null values are allowed in the column.

The following figure shows the kind of information that you see when you select the Columns option for the **cust_calls** table.

```
Column name        Type                              Nulls

customer_num       INTEGER                           no
call_dtime         DATETIME YEAR TO MINUTE           yes
user_id            CHAR(32)                          yes
call_code          CHAR(1)                           yes
call_descr         CHAR(240)                         yes
res_dtime          DATETIME YEAR TO MINUTE           yes
res_descr          CHAR(240)                         yes
```

Figure 5-47. Display column information for the cust_calls table

DB-Access data types

The columns in Figure 5-47 show that the **cust_calls** table consists of all built-in (standard) data types. The data types available through the CREATE TABLE menu hierarchy are built-in types.

The built-in types that the Columns option can display are

- BIGINT

- BIGSERIAL
- BOOLEAN
- BYTE
- CHAR
- CHARACTER VARYING
- DATE
- DATETIME
- DECIMAL
- FLOAT
- INT8
- INTEGER
- INTERVAL
- MONEY
- NCHAR
- NVARCHAR
- SERIAL
- SERIAL8
- SMALLFLOAT
- SMALLINT
- TEXT
- VARCHAR
- user-defined types.

For descriptions of these data types, see the *IBM Informix Guide to SQL: Reference*.

The following figure shows the display of column information for a table that has BOOLEAN, INT8, and SERIAL8 columns and other built-in data types.

```
Column name          Type                          Nulls

id_num              integer                       yes
yes_or_no           boolean                       yes
int8col             int8                          yes
serial8col          serial8                       yes
text_descr          char(20)                      yes
```

Figure 5-48. Display column information for a table with several built-in types

Large objects: Large objects are built-in data types that store a large amount of data in a single column. Within a table, large-object data type columns actually contain pointers to the *physical* storage spaces where the database server places the large data objects.

DB-Access can display any of the following large-object data types:
- TEXT, BYTE
- CLOB, BLOB

The Columns option displays the specific data type for any column that contains pointers to large objects. The following figure shows the display of column information for a table that has a BYTE column.

```
Column name         Type                          Nulls

id                  integer                       yes
binary_col          byte                          yes
```

Figure 5-49. Display column information for a BLOB column

Extended data types in Informix

These topics show how DB-Access displays user-defined and complex data types for IBM Informix.

Opaque data types: An opaque data type characterizes data that cannot be represented by any of the built-in types that belong to the database server. DB-Access can identify and display opaque data types.

For example, suppose you assign an opaque data type called **circle_t** to a column named **circle_col**. The Columns option displays the opaque data type name in the **Type** column, as the following figure shows.

```
Column name       Type                              Nulls

id                integer                           yes
circle_col        circle_t                          yes
```

Figure 5-50. Display information for a column with an opaque data type

Collection types: A collection type contains zero or more elements and is more specifically defined with one of the following data type names.

SET An unordered collection of elements in which duplicates are not allowed

MULTISET
An unordered collection of elements in which duplicates are allowed

LIST An ordered collection of elements in which duplicates are allowed

DB-Access displays the specific kind of collection type in the **Type** column. For example, the following figure shows the display of a SET data type column named **siblings**.

```
Column name         Type                          Nulls

id                  integer                       yes
siblings            set                           yes
```

Figure 5-51. Display information for a column with a collection data type

Row types: The Columns option for a table that includes a column with a row type displays the string **Row** in the **Type** column. DB-Access displays this string whether the column has a named or unnamed row type. Assume you define row type **rectangle_t** and assign it to column **rect**. The following figure shows that the Columns display for **rect** returns row rather than the specific row-type name **rectangle_t** as the data type.

```
Column name         Type                          Nulls

id                  integer                       yes
rect                row                           yes
```

Figure 5-52. Column information for a row data type

Display index information

The following figure shows the kind of information that you see when you select the Indexes option for the **cust_calls** table. The **c_num_dt_ix** index is a B-tree index defined on two columns of the **cust_calls** table. The **c_num_cus_ix** index is a B-tree index defined on a single column of the **cust_calls** table.

```
Index name        Owner     Type/Clstr    Access_Method       Columns

c_num_dt_ix       velma     unique/No     B-Tree              customer_num
                                                              call_dtime
c_num_cus_ix      velma     dupls/No      B-Tree              customer_num
```

Figure 5-53. Display of index information

The following table shows the meaning of each column in the display.

Index Name
The name of the index

Owner
The owner of the index

Type The index type (unique or duplicate)

Clstr Indicates whether the index is clustered. (A clustered index causes the table to be physically reordered in the same sequence as the index.)

Access Method
The index access method (such as B-tree or functional)

Columns
The column or columns on which the index is defined

For further information about the types of indexes available on your database server, see your *IBM Informix Performance Guide*.

You can display information for non-B-tree indexes, including indexes based on user-defined secondary access methods that IBM Informix permits. For example, the index shown in the following figure is based on a **Fulltext** access method that a DataBlade module provides.

```
Index name       Owner    Type/Clstr    Access_Method        Columns
text_idx         wilma     dupls/No      Fulltext            zone_descr
```

Figure 5-54. Display of information for a DataBlade module index

For information about user-defined access methods, see your DataBlade documentation or the documentation for the access method.

Display table-level privileges

The following figure shows the kind of table-level access-privileges information that you see when you select the Privileges option for the **cust_calls** table.

```
User    Select   Update   Insert   Delete   Index   Alter

public  All      All      Yes      Yes      Yes     No
```

Figure 5-55. Display of Privileges information

Unless your login is listed separately, you have the privileges given for **public** (a general category for all users).

If you want information about database-level privileges, use a SELECT statement to access the **sysusers** system catalog table.

Display references privileges

The following figure shows the kind of information that you see when you select the References option for a table that has referential integrity.

```
User          Column References

betty         col1
              col2
              col3
wilma         All
public        None
```

Figure 5-56. Display of References information

This display indicates the following table-level references privileges:

- The user `betty` can reference columns 1, 2, and 3 of the specified table.
- The user `wilma` can reference all the columns in the table.
- Users who only have privileges granted to `public` cannot access any columns in the table.

Display column constraints and defaults

When you select the cOnstraints option from the INFO menu, the CONSTRAINTS menu opens, as the following figure shows.

```
CONSTRAINTS - mytab:   Reference  Primary Check Unique Defaults Exit
Referenced and referencing options.

------------------mydata@mydbserv-------------- Press CTRL-W for Help -------
```

Figure 5-57. The CONSTRAINTS menu

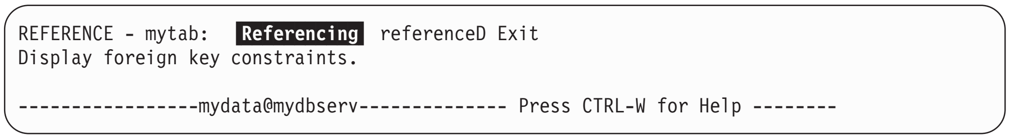
You can select from the following options on the CONSTRAINTS menu.

Option	Purpose	Instructions
Reference	Shows referential (foreign-key) constraints associated with the specified table	"Display referential constraints" on page 5-33
Primary	Shows primary-key columns in the specified table	Figure 5-61 on page 5-34

Option	Purpose	Instructions
Check	Shows check constraints on columns in the specified table	Figure 5-62 on page 5-34
Unique	Lists columns that must contain unique data	none
Defaults	Lists columns for which default values are defined and the default value	none
Exit	Returns to the INFO menu	none

Display referential constraints

When you select the Reference option from the CONSTRAINTS menu, the REFERENCE menu opens, as the following figure shows.



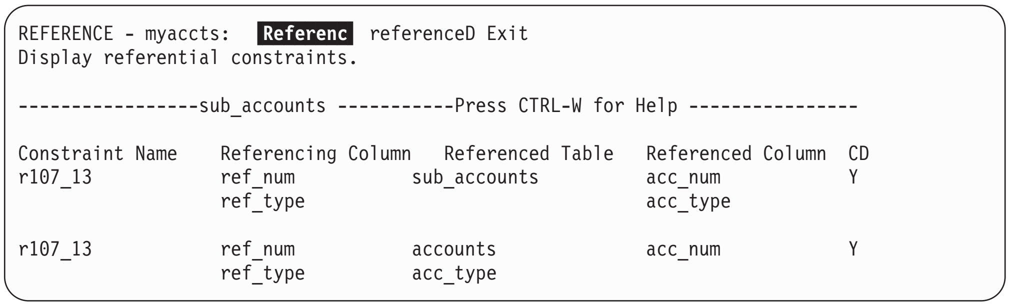
```
REFERENCE - mytab:  Referencing  referenceD Exit
Display foreign key constraints.

-----------------mydata@mydbserv-------------- Press CTRL-W for Help --------
```

Figure 5-58. The REFERENCE menu

The REFERENCE menu lets you display the following information:

- Foreign-key (referencing) constraints
- Columns of the current table
- Cascading deletes enabling
- Referenced columns (columns referenced as foreign keys by columns from another table)

Select the Referencing option on the REFERENCE menu to display the foreign-key constraints in the current table, as the following figure shows.



```
REFERENCE - myaccts:  Referenc  referenceD Exit
Display referential constraints.

-----------------sub_accounts -----------Press CTRL-W for Help ----------------

Constraint Name    Referencing Column   Referenced Table   Referenced Column  CD
r107_13            ref_num          sub_accounts         acc_num            Y
                   ref_type                              acc_type

r107_13            ref_num          accounts             acc_num            Y
                   ref_type         acc_type
```

Figure 5-59. The REFERENCE menu with referencing information

Select the referenceD option on the REFERENCE menu to display other tables and columns that reference your current columns as foreign keys, as the following figure shows.

```
REFERENCE - myaccts:  Referenc  referenceD  Exit
Display columns which have foreign keys which reference this table.

------------------accounts ------------Press CTRL-W for Help ------------------

Constraint Name    Referenced Column   Referencing Table   Referencing Column  CD
r107_13            acc_num            sub_accounts         ref_acc             Y
                   acc_type                                ref_type
```

Figure 5-60. The REFERENCE menu with referenced information

Select Exit to return to the CONSTRAINTS menu.

Sample primary and check constraint displays

The following figure shows the columns that form a primary key. This display results from selecting Primary from the CONSTRAINTS menu.

```
CONSTRAINTS - mytab:  References  Primary  Check Unique Defaults Exit
Primary key constraints

-----------------mydata@mydbserv------------------ Press CTRL-W for Help ------

Constraint Name     Column Name

constraint1     assembly
                  partnum
```

Figure 5-61. The CONSTRAINTS menu with primary-key constraint information

Select the Check option on the CONSTRAINTS menu to display the check constraints placed on columns of the current table, as the following figure shows.

```
CONSTRAINTS - mytab:  Reference Primary  Check  Unique Defaults Exit
Display check constraints.

--------------------mydata@mydbserv------------- Press CTRL-W for Help -------

Constraint name           Value

cons2                     (column1 > (column2 * 100 - 1000 + column1 / 2 -
                          column2/20 + 40 * 3 - 55 * column2 + 77 * column1))

cons3                     column2 > column3

cons4                     column3 > 100
```

Figure 5-62. The CONSTRAINTS menu with check-constraint information

Display triggers

When you select the triGgers option from the INFO menu, the INFO FOR TRIGGER screen opens, as the following figure shows.

```
INFO FOR TRIGGER>>
Choose a trigger with the Arrow Keys, or enter a name, then press Return.

------------------mydata@mydbserv--------------- Press CTRL-W for Help ------

 updrec_t
```

Figure 5-63. The INFO FOR TRIGGER screen

Select a trigger from the list of trigger names in the current table. If you do not want to select a trigger, press the Interrupt key, and you return to the TABLE INFO menu.

If the header and body information for the selected trigger fit on one screen, the INFO menu reopens, displaying the trigger information, as the following figure shows.

```
INFO - updrec_t:  ....  triGgers   Status  Table  Exit
Display header and body information for a trigger.

---------------- mystores@dbserver1 ----------- Press CTRL-W for Help ------

create trigger updrec_t
                unit_price on stock
                        referencing old as pre_upd
                                new as post_upd

        (insert into log_record values (stock_num, CURRENT,
        pre_upd.unit_price, post_upd.unit_price)) for each row;
```

Figure 5-64. Display of trigger information from the INFO menu

The previous figure shows the header information for a trigger, which consists of the CREATE TRIGGER statement and trigger name, the SQL statement that triggers an event, and the referencing clause. The body of a trigger is the triggered action. In Figure 5-64, the triggered action is the INSERT statement.

If the trigger does not fit on a single INFO menu screen, use the menu at the top of the screen as follows:
- Press **N**, or if the Next option is highlighted, press **RETURN** to advance to the next screen of trigger information. Continue to press **N** or **RETURN** as needed to page through the information.
- Select Restart at any time to display the trigger information from the beginning.
- Select Exit to return to the TABLE menu.

Display fragmentation information

The following figure shows the kind of information that you see when you select the Fragments option for an indexed table created with a round-robin fragmentation strategy.

```
Idx/Tbl Name     Dbspace         Type        Expression

cust            dbspace1        T
cust            dbspace2        T
cust            rootdbs         T
cust            dbspace1        I
cust            dbspace2        I
cust            rootdbs         I
```

Figure 5-65. Fragmentation information where round-robin strategy applies to both table and index

`Idx/Tbl Name` shows the object that was explicitly fragmented.

If you use Informix, the display includes a **Type** column to indicate whether the fragment on the line is part of an index or the table data. In the previous figure, the **cuts** table was created with round-robin strategy, but the index was created without specifying a strategy. In this case, the indexes are located in default dbspaces.

Suppose, when creating an index, you use the following statement to apply a fragmentation strategy:

```
create index idx on cust(custnum) fragment by expression
   custnum < 200 in dbspace1,
   custnum > 200 in dbspace2,
   remainder in rootdbs;
```

In this case, the fragmentation display looks like the following figure.

```
Idx/Tbl Name     Dbspace        Type        Expression

cust            dbspace1        T
cust            dbspace2        T
cust            rootdbs         T
idx             dbspace1        I           (custnum < 200)
idx             dbspace2        I           (custnum > 200)
idx             rootdbs         I           remainder
```

Figure 5-66. Fragmentation information where table and index have different strategies

`Idx/Tbl Name` shows the index name because the fragmentation was explicitly applied to the index.

Drop a table

Use the Drop option on the TABLE menu to delete an existing table schema from the database. Press the **D** key, or highlight Drop and press **RETURN**. The DROP TABLE screen opens, as the following figure shows.

```
DROP TABLE >>
Enter the table name you wish to drop from the database.

----------------- mydata@mydbserv --------------- Press CTRL-W for Help -----

  clients

 customer

 orders
```

Figure 5-67. The DROP TABLE screen

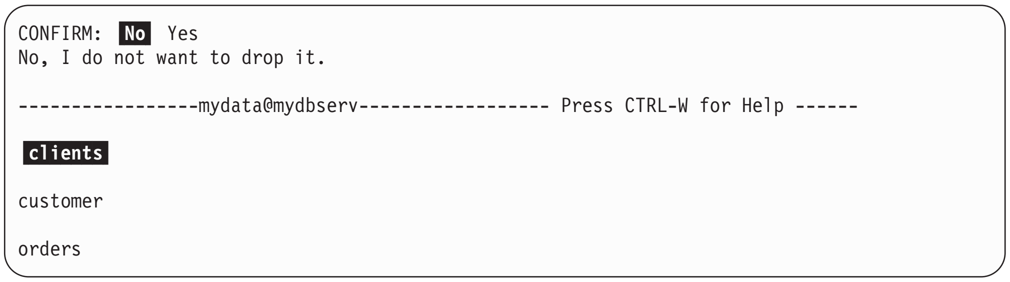
This screen lists the names of tables that exist in the current database. You can delete a table in one of the following ways:

- Type the table name and press **RETURN**. You must use this method and include the full path name if you want to delete a table that is not in the current database.
- Use the arrow keys to highlight the name of the table that you want to delete from the database and press **RETURN**.

To leave the DROP TABLE screen without deleting a table, press the Interrupt key. You return to the TABLE menu.

Warning: When you delete a table, you delete both the table and all the data it contains.

After you select a table to delete, DB-Access displays the CONFIRM menu, which asks for confirmation before it deletes the table, as the following figure shows.



```
CONFIRM:  No  Yes
No, I do not want to drop it.

-----------------mydata@mydbserv------------------ Press CTRL-W for Help ------

 clients

customer

orders
```

Figure 5-68. The CONFIRM menu

The default is No to prevent you from deleting a table. You must explicitly delete a table. Thus, if you want to delete the highlighted table, press the **Y** key or use the right arrow key to highlight Yes and press **RETURN**. DB-Access deletes the table.

Chapter 6. The Connection and Session options

These topics describe the Connection and Session options on the main menu.

Overview of Connection and Session options

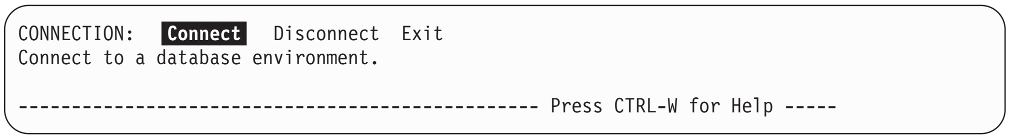
Use the Connection option if you want to connect to a specific database server and database or explicitly disconnect from the current database environment. Use the Session option to display information about the current DB-Access session.

For the GLS considerations that apply to establishing a connection between a client application, such as DB-Access, and a database, see the *IBM Informix GLS User's Guide*. The database server examines the client locale information passed by the client, verifies the database locale, and determines the server-processing locale for transferring data between the client and the database.

You can use the Secure Sockets Layer (SSL) protocol, a communication protocol that ensures privacy and integrity of data transmitted over the network, for DB-Access connections with IBM Informix. For information about the SSL protocol, see the "Secure Sockets Layer Communication Protocol" section of the *IBM Informix Security Guide*.

Choose the Connection option

From the main menu, press the **C** key (or highlight the Connection option and press **RETURN**) to display the CONNECTION menu, as the following figure shows.



```
CONNECTION:  Connect  Disconnect  Exit
Connect to a database environment.

------------------------------------------------ Press CTRL-W for Help -----
```

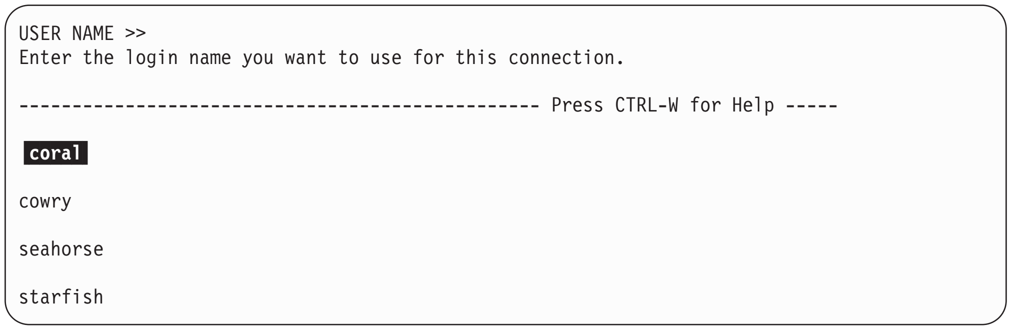
Figure 6-1. The CONNECTION menu

The CONNECTION menu displays the following options.

Option	Purpose	Instructions
Connect	Connects to a database environment	"Connect to a database environment"
Disconnect	Disconnects from the current database environment	"Disconnect from a database environment" on page 6-4
Exit	Returns to the DB-Access main menu	none

Connect to a database environment

To connect to an existing database server, choose the Connect option from the CONNECTION menu. DB-Access displays a list of available database servers and prompts you to make a selection. Select a database server and DB-Access prompts you to enter a user name, as the following figure shows.



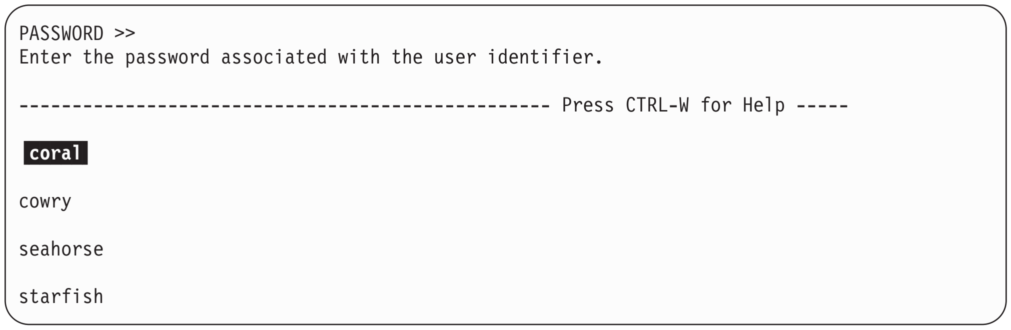
```
USER NAME >>
Enter the login name you want to use for this connection.

------------------------------------------------- Press CTRL-W for Help -----

coral

cowry

seahorse

starfish
```

Figure 6-2. The USER NAME prompt screen

On Windows, if you specify a user identifier but no domain name for a connection to a machine that expects both a domain name and a user name (*domain\user*), DB-Access checks only the local machine and the primary domain for the user account. If you explicitly specify a domain name, that domain is used to search for the user account. The attempted connection fails with error -951 if no matching *domain\user* account is found on the local machine.

If you do not specify a user identifier on the USER NAME screen and press **RETURN**, you see the standard SELECT DATABASE screen listing databases on the chosen database server.

If you enter the login name that you want DB-Access to use when connecting to the target database server, DB-Access displays the PASSWORD screen, as the following figure shows.



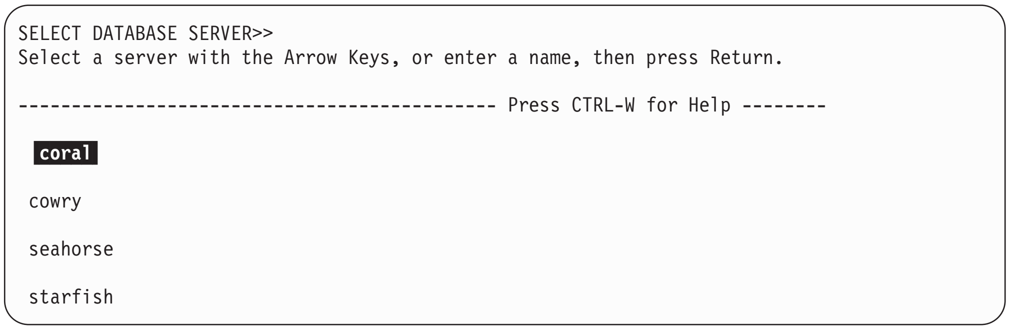
```
PASSWORD >>
Enter the password associated with the user identifier.

-------------------------------------------------- Press CTRL-W for Help -----

coral

cowry

seahorse

starfish
```

Figure 6-3. The PASSWORD screen

From the PASSWORD screen, enter a password associated with the user identifier or press **RETURN** if you do not want to enter a password. For security reasons, the password that you enter on the screen is not displayed.

Tip: If you press **CRTL-C** on the USER NAME screen, DB-Access might try to connect to the specified database server rather than interrupt the session. This situation occurs because pressing **CRTL-C** on this screen is the equivalent of using the current user name.

If the user identifier and password combination are valid, you connect to the target database server. You can then select a database from that database server. The SELECT DATABASE SERVER screen opens, as the following figure shows.



```
SELECT DATABASE SERVER>>
Select a server with the Arrow Keys, or enter a name, then press Return.

--------------------------------------------- Press CTRL-W for Help --------

 coral

 cowry

 seahorse

 starfish
```

Figure 6-4. The SELECT DATABASE SERVER screen

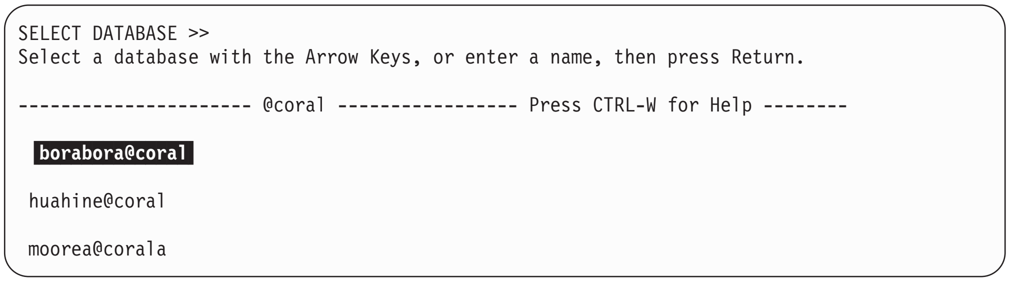
The SELECT DATABASE SERVER screen shows an alphabetical list of database servers from the `$INFORMIXDIR/etc/sqlhosts` file on UNIX or the `sqlhosts` entry in the registry on Windows NT. The first entry is highlighted. Press **RETURN** to select that database server or select a different database server. If entries span more than one page, use the arrow keys to move the cursor to highlight the ellipses (...), and the next page of database servers is displayed.

Permissions needed

To access a specific database, you must have permission. If you do not have permission to connect to the specified database server, an error message is displayed. Select a different database server for which you have permission or ask your database server administrator for permission to connect to the database server.

If you have the correct permissions for the specified database server, you are prompted to specify a database to use on that database server.

When the SELECT DATABASE screen opens, the name of the specified database server is highlighted, as the following figure shows.



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

----------------------- @coral ----------------- Press CTRL-W for Help --------

 borabora@coral

 huahine@coral

 moorea@corala
```

Figure 6-5. The SELECT DATABASE screen

The SELECT DATABASE screen alphabetically lists all available databases on the specified database server. The database list on the SELECT DATABASE screen depends on the current connection. For example:

- If no current connection exists or the current connection is an implicit default connection, all the databases listed in the **DBPATH** environment variable setting are displayed.
- If a current explicit connection exists, all the databases in the **DBPATH** that pertain to the current server are displayed.

For information about setting the **DBPATH** environment variable, see the *IBM Informix Guide to SQL: Reference*.

To select a database, type the database name or use the arrow keys to highlight the name of a database, then press **RETURN**.

If you enter the CONNECT menu with a current connection, and the new connection succeeds, DB-Access disconnects from the previous environment and closes any databases that belong to that environment. For more information, see "Implicit closures."

If you enter the name of a nonexistent database or a database that DB-Access cannot locate, an error message is displayed.

To leave the SELECT DATABASE screen without selecting a database, press the Interrupt key. You return to the CONNECTION menu, but your database server connection is not severed.

Implicit closures

DB-Access closes any open connections or databases when you connect to a new environment, in the following situations:

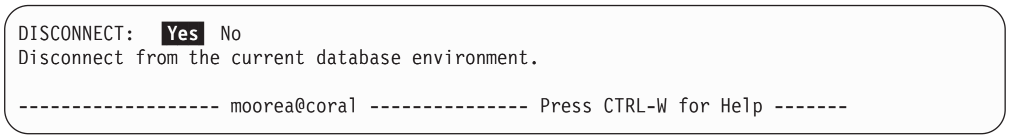
- When you connect to a new database environment without explicitly disconnecting from the current one, DB-Access performs an implicit disconnect and the database closes.
- When you connect to a database@server and then close the database, the database server remains connected.
- When you connect to a database server, open a database, and then close the database, the database server remains connected.
- If you open a database and then try to connect to a database server, DB-Access performs an implicit disconnect and closes the database.

  Only one connection is allowed. You must disconnect from the database server associated with the open database or close the database before you can connect to another database server.

If DB-Access must close a database that still has outstanding transactions, it prompts you to commit or rollback those transactions, as described in "Transaction processing" on page 6-5.

Disconnect from a database environment

To disconnect from the current database server and close the current database, choose the Disconnect option from the CONNECTION menu. The DISCONNECT confirmation menu opens, as the following figure shows.



```
DISCONNECT:  Yes  No
Disconnect from the current database environment.

------------------- moorea@coral --------------- Press CTRL-W for Help -------
```

Figure 6-6. The DISCONNECT confirmation menu

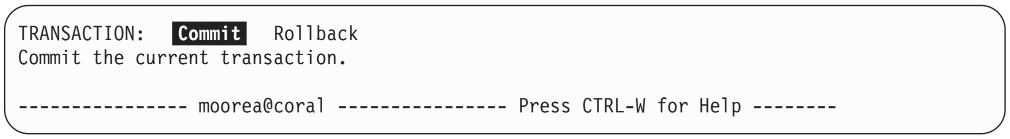
When you select the Disconnect option from the CONNECTION menu, you must confirm your decision on the DISCONNECT confirmation screen. The following two options are available:

- To confirm that you want to disconnect, press **RETURN** with the default Yes option highlighted. DB-Access disconnects from the database server and closes the database.

- If you do not want to disconnect, press the **N** key or use the right arrow key to highlight No, and press **RETURN**. DB-Access returns to the CONNECTION menu.

Transaction processing

A database that has transaction logging prompts you to confirm or roll back any transactions when you explicitly disconnect from the current database environment or when you connect to another environment, which forces DB-Access to close an open database. The TRANSACTION menu opens, as the following figure shows.



```
TRANSACTION:  Commit  Rollback
Commit the current transaction.

---------------- moorea@coral ---------------- Press CTRL-W for Help --------
```

Figure 6-7. The TRANSACTION menu for databases with transactions

The TRANSACTION menu ensures that you either commit or roll back an active transaction before you close the current database. You have the following menu options:

- The default is Commit. Press **RETURN**, and DB-Access commits the transaction and closes the database.
- If you want to roll back the transaction, use an arrow key to move the highlight to the Rollback option. Press **RETURN** and DB-Access rolls back the transaction and closes the database.

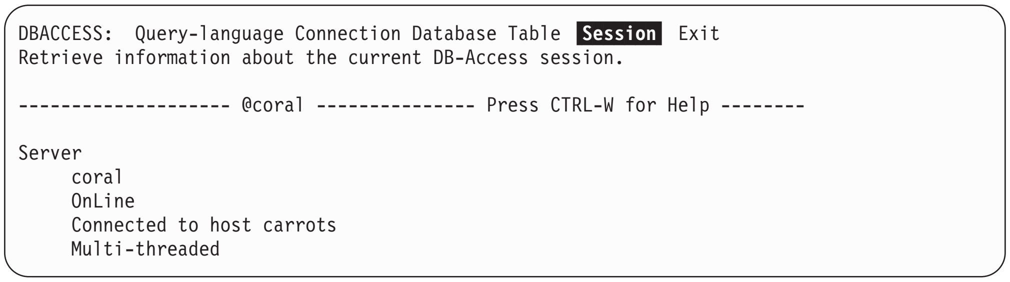
Important: Select an option carefully. You might commit transactions that you do not want if you select Commit. You will lose any new transactions if you select Rollback.

If you press the Interrupt key, DB-Access displays the DATABASE menu without committing or rolling back the transaction.

Choose the session option

From the main menu, press the **S** key or highlight the Session option and press **RETURN**. The DB-Access main menu remains on the screen and information about the current DB-Access session is displayed. For example, the following figure shows the following information:

- The name of the database server for this session is **coral**.
- The database server type is IBM Informix. This server type refers to all Informix database servers that are based on dynamic scalable architecture. For example, you would see Informix if you use Informix.
- The server is connected to a host named **carrots**.
- The server is a multithreaded server.



```
DBACCESS:  Query-language Connection Database Table  Session  Exit
Retrieve information about the current DB-Access session.

-------------------- @coral --------------- Press CTRL-W for Help --------

Server
     coral
     OnLine
     Connected to host carrots
     Multi-threaded
```

Figure 6-8. Main menu with sample session information for an Informix instance

If you are running a database that supports Native Language Support, the Session option shows the collating sequence and character type.

The Session option does not show Global Language Support attributes, but you can use the method shown in "Retrieve nondefault locale information" on page 4-6 to obtain these settings.

To exit from the Session information screen, select another option on the main menu.

Appendix A. How to read online help for SQL statements

This topic shows the conventions that are used to represent the syntax of SQL statements in DB-Access online help screens. You can request online help for SQL statements in either of the following ways:

- Highlight the New, Modify, or Use-editor options on the SQL menu and press **CTRL-W**.
- Press **CTRL-W** while you are on the NEW or MODIFY screens of the SQL menu.

The form of the syntax diagrams that shows when you request online Help for SQL statements in DB-Access is different from the syntax diagrams in the *IBM Informix Guide to SQL: Syntax*.

The conventions and rules governing SQL statement syntax in DB-Access online help screens are described in the following list.

ABC
Any term in an SQL statement that is displayed in uppercase letters is a keyword. Type keywords exactly, disregarding case, as shown in the following example:

```
CREATE SYNONYM synonym-name
```

This syntax indicates you must type the keywords CREATE SYNONYM or create synonym without adding or deleting spaces or letters.

abc
Substitute a value for any term that is displayed in lowercase letters. In the previous example, you should substitute a value for *synonym-name*.

( )
Type any parentheses as shown. They are part of the syntax of an SQL statement and are not special symbols.

[ ]
Do not type brackets as part of a statement. They surround any part of a statement that is optional. For example:

```
CREATE [TEMP] TABLE
```

This syntax indicates that you can type either CREATE TABLE or CREATE TEMP TABLE.

|
The vertical bar indicates a choice among several options. For example:

```
[VANILLA | CHOCOLATE [MINT] | STRAWBERRY]
```

This syntax indicates that you can enter either VANILLA, CHOCOLATE, or STRAWBERRY and that, if you enter CHOCOLATE, you can also enter MINT.

{ }
When you must choose only one of several options, the options are enclosed in braces and are separated by vertical bars. For example:

```
{GUAVA | MANGO | PASSIONFRUIT}
```

This syntax indicates that you must enter either GUAVA, MANGO, or PASSIONFRUIT, but you cannot enter more than one choice.

...
An ellipsis indicates that you can enter an indefinite number of additional items, such as the one immediately preceding the ellipsis. For example:

```
old-column-name
...
```

This syntax indicates that you can enter a series of existing column names after the first one.

The *IBM Informix Guide to SQL: Syntax* contains more detailed syntax diagrams and instructions for interpreting the diagram format used in the publication.

For a general explanation of how to use online Help in DB-Access, see “The HELP screen” on page 2-4.

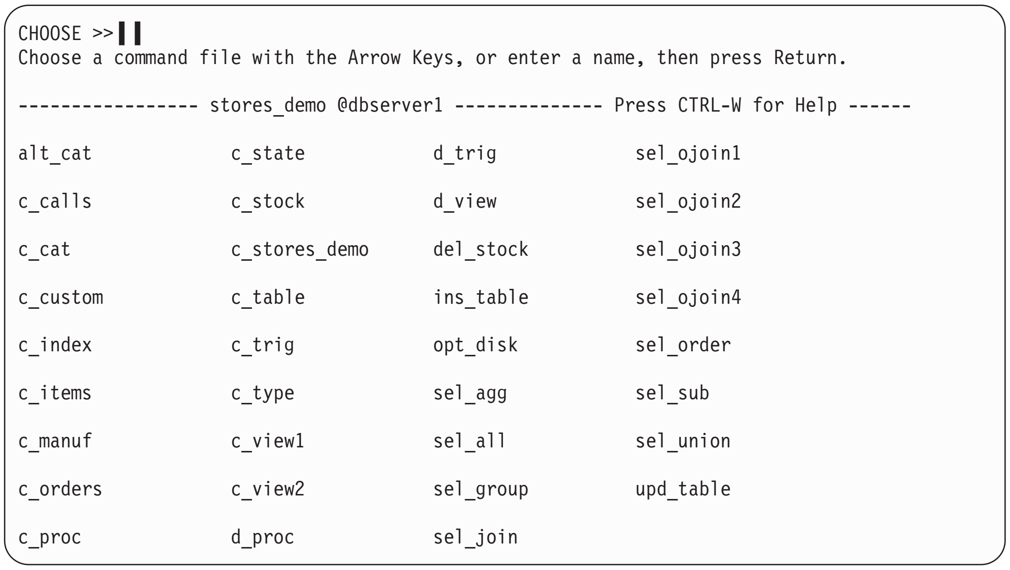
Appendix B. Demonstration SQL

These topics show the contents of the various command files that are available with DB-Access. These command files all have the extension `.sql` when displayed from the command line but are displayed without the extension on the SQL CHOOSE menu.

Keywords in these command files are shown in uppercase letters to make the SQL statements easier to read. Keywords in the actual command files are lowercase.

Important: Although the command files are listed alphabetically in this appendix, you cannot execute the command files that create tables in that order without causing errors. The order in which the tables are created is very important because of the referential constraints that link those tables.

When you select the Choose option on the SQL menu, the CHOOSE screen opens. The screen shows a list of the command files that you can access, similar to the display that the following figure shows. These files are included with the **stores_demo** database. Other `.sql` files are described later in this appendix.



```
CHOOSE >>
Choose a command file with the Arrow Keys, or enter a name, then press Return.

----------------- stores_demo @dbserver1 -------------- Press CTRL-W for Help ------

alt_cat            c_state            d_trig             sel_ojoin1

c_calls            c_stock            d_view             sel_ojoin2

c_cat              c_stores_demo      del_stock          sel_ojoin3

c_custom           c_table            ins_table          sel_ojoin4

c_index            c_trig             opt_disk           sel_order

c_items            c_type             sel_agg            sel_sub

c_manuf            c_view1            sel_all            sel_union

c_orders           c_view2            sel_group          upd_table

c_proc             d_proc             sel_join
```

Figure B-1. Command files listed on the CHOOSE screen

If you do not see the command files included with your demonstration database, check the following:

1. Did you copy the demonstration SQL command files to your current directory when you ran the demonstration database initialization script? If not, you can rerun the initialization script to copy them.
2. Did you start DB-Access from the directory in which you installed the demonstration SQL command files? If not, exit DB-Access, change to the appropriate directory, and start DB-Access again.

For instructions about running the initialization script, copying command files, and starting DB-Access, seeChapter 1, “Getting started with DB-Access,” on page 1-1.

Use these command files with DB-Access for practice with SQL and the demonstration database. You can rerun the demonstration database initialization script whenever you want to refresh the database tables and SQL files.

SQL files for the Relational database model

The SQL files described in these are included with the **stores_demo** demonstration database and conform to the **stores_demo** schema. For more information about the **stores_demo** schema, see the *IBM Informix Guide to SQL: Reference*.

The alt_cat.sql command file

The following command file alters the **catalog** table. It drops the existing constraint **aa** on the **catalog** table and adds a new constraint, **ab**, which specifies cascading deletes. You can use this command file and then the `del_stock.sql` command file for practice with cascading deletes on a database with logging.

```
ALTER TABLE catalog DROP CONSTRAINT aa;

ALTER TABLE catalog ADD CONSTRAINT
   (FOREIGN KEY (stock_num, manu_code) REFERENCES stock
    ON DELETE CASCADE CONSTRAINT ab);
```

The c_calls.sql command file

The following command file creates the **cust_calls** table:

```
CREATE TABLE cust_calls
   (
   customer_num         INTEGER,
   call_dtime           DATETIME YEAR TO MINUTE,
   user_id              CHAR(18) DEFAULT USER,
   call_code            CHAR(1),
   call_descr           CHAR(240),
   res_dtime            DATETIME YEAR TO MINUTE,
   res_descr            CHAR(240),
   PRIMARY KEY (customer_num, call_dtime),
   FOREIGN KEY (customer_num) REFERENCES customer (customer_num),
   FOREIGN KEY (call_code) REFERENCES call_type (call_code)
   );
```

The c_cat.sql command file

The following command file creates the **catalog** table. It contains a constraint, **aa**, which allows you to practice with cascading deletes by running the SQL statements in the `alt_cat.sql` and `del_stock.sql` command files on a database with logging.

```
CREATE TABLE catalog
   (
   catalog_num      SERIAL(10001),
   stock_num        SMALLINT NOT NULL,
   manu_code        CHAR(3) NOT NULL,
   cat_descr        TEXT,
   cat_picture      BYTE,
   cat_advert       VARCHAR(255, 65),
   PRIMARY KEY (catalog_num),
   FOREIGN KEY (stock_num, manu_code) REFERENCES stock
      CONSTRAINT aa
   );
```

The c_custom.sql command file

The following command file creates the **customer** table:

```
CREATE TABLE customer
   (
   customer_num       SERIAL(101),
   fname              CHAR(15),
   lname              CHAR(15),
   company            CHAR(20),
   address1           CHAR(20),
   address2           CHAR(20),
   city               CHAR(15),
   state              CHAR(2),
   zipcode            CHAR(5),
   phone              CHAR(18),
   PRIMARY KEY (customer_num)
   );
```

The c_index.sql command file

The following command file creates an index on the **zipcode** column of the **customer** table:

```
CREATE INDEX zip_ix ON customer (zipcode);
```

The c_items.sql command file

The following command file creates the **items** table:

```
CREATE TABLE items
   (
   item_num          SMALLINT,
   order_num         INTEGER,
   stock_num         SMALLINT NOT NULL,
   manu_code         CHAR(3) NOT NULL,
   quantity          SMALLINT CHECK (quantity >= 1),
   total_price       MONEY(8),
   PRIMARY KEY (item_num, order_num),
   FOREIGN KEY (order_num) REFERENCES orders (order_num),
   FOREIGN KEY (stock_num, manu_code) REFERENCES stock
      (stock_num, manu_code)
   );
```

The c_manuf.sql command file

The following command file creates the **manufact** table:

```
CREATE TABLE manufact
   (
   manu_code               CHAR(3),
   manu_name               CHAR(15),
   lead_time               INTERVAL DAY(3) TO DAY,
   PRIMARY KEY (manu_code)
   );
```

The c_orders.sql file

The following command file creates the **orders** table:

```
CREATE TABLE orders
   (
   order_num            SERIAL(1001),
   order_date           DATE,
   customer_num        INTEGER NOT NULL,
   ship_instruct       CHAR(40),
   backlog             CHAR(1),
   po_num              CHAR(10),
   ship_date           DATE,
   ship_weight         DECIMAL(8,2),
   ship_charge         MONEY(6),
   paid_date           DATE,
   PRIMARY KEY (order_num),
   FOREIGN KEY (customer_num) REFERENCES customer (customer_num)
   );
```

The c_proc.sql command file

The following command file creates an SPL routine. It reads the full name and address of a customer and takes a last name as its only argument.

This routine shows the legacy use of CREATE PROCEDURE.

To conform with the SQL standard preferred with IBM Informix, define a *function* if you want to return values from a routine.

```
CREATE PROCEDURE read_address (lastname CHAR(15))
   RETURNING CHAR(15), CHAR(15), CHAR(20), CHAR(15), CHAR(2), CHAR(5);
   DEFINE p_fname, p_city CHAR(15);
   DEFINE p_add CHAR(20);
   DEFINE p_state CHAR(2);
   DEFINE p_zip CHAR(5);
   SELECT fname, address1, city, state, zipcode
      INTO p_fname,  p_add, p_city, p_state, p_zip
      FROM customer
      WHERE lname = lastname;

   RETURN p_fname, lastname, p_add, p_city, p_state, p_zip;

END PROCEDURE;
```

The c_state command file

The following command file creates the **state** table:

```
CREATE TABLE state
   (
   code              CHAR(2),
   sname             CHAR(15),
   PRIMARY KEY (code)
   );
```

The c_stock.sql command file

The following command file creates the **stock** table:

```
CREATE TABLE stock
   (
   stock_num         SMALLINT,
   manu_code         CHAR(3),
   description       CHAR(15),
   unit_price        MONEY(6),
   unit              CHAR(4),
   unit_descr        CHAR(15),
   PRIMARY KEY (stock_num, manu_code),
   FOREIGN KEY (manu_code) REFERENCES manufact
   );
```

The c_stores.sql command file

The following command file creates the **stores_demo** database:

```
CREATE DATABASE stores_demo;
```

The c_table.sql command file

The following command file creates a database named **restock** and then creates a custom table named **sports** in that database:

```
CREATE DATABASE restock;

CREATE TABLE sports
   (
   catalog_no          SERIAL UNIQUE,
   stock_no            SMALLINT,
   mfg_code            CHAR(5),
   mfg_name            CHAR(20),
   phone               CHAR(18),
   descript            VARCHAR(255)
   );
```

The c_trig.sql command file

The following command file creates a table named **log_record** and then creates a trigger named **upqty_i**, which updates it:

```
CREATE TABLE log_record
   (item_num    SMALLINT,
   ord_num      INTEGER,
   username     CHARACTER(8),
   update_time  DATETIME YEAR TO MINUTE,
   old_qty      SMALLINT,
   new_qty      SMALLINT);

CREATE TRIGGER upqty_i
UPDATE OF quantity ON items
REFERENCING OLD AS pre_upd
        NEW AS post_upd
FOR EACH ROW(INSERT INTO log_record
   VALUES (pre_upd.item_num, pre_upd.order_num, USER, CURRENT,
        pre_upd.quantity, post_upd.quantity));
```

The c_type.sql command file

The following command file creates the **call_type** table:

```
CREATE TABLE call_type
   (
   call_code              CHAR(1),
   code_descr             CHAR(30),
   PRIMARY KEY (call_code)
   );
```

The c_view1.sql command file

The following command file creates a view called **custview** on a single table and grants privileges on the view to **public**. It includes the WITH CHECK OPTION keywords to verify that any changes made to underlying tables through the view do not violate the definition of the view.

```
CREATE VIEW custview (firstname, lastname, company, city) AS
   SELECT fname, lname, company, city
      FROM customer
      WHERE city = 'Redwood City'
      WITH CHECK OPTION;

GRANT DELETE, INSERT, SELECT, UPDATE
   ON custview
   TO public;
```

The c_view2.sql command file

The following command file creates a view on the **orders** and **items** tables:

```
CREATE VIEW someorders (custnum,ocustnum,newprice) AS
   SELECT orders.order_num, items.order_num,
         items.total_price*1.5
      FROM orders, items
      WHERE orders.order_num = items.order_num
      AND items.total_price > 100.00;
```

The d_proc.sql command file

The following command file drops the SPL routine that the `c_proc.sql` command file created:

```
DROP PROCEDURE read_address;
```

The d_trig.sql command file

The following command file drops the trigger that the `c_trig.sql` command file created:

```
DROP TRIGGER upqty_i;
```

The d_view.sql command file

The following command file drops the view named **custview** that the `c_view1.sql` command file created:

```
DROP VIEW custview;
```

The del_stock.sql command file

The following command file deletes rows from the **stock** table where the stock number is 102. This delete will cascade to the **catalog** table (although the related manufacturer codes will remain in the **manufact** table). The `del_stock.sql` command file can be used following the `alt_cat.sql` command file for practice with cascading deletes on a database with logging.

```
DELETE FROM stock WHERE stock_num = 102;
```

After running the SQL statements in the `alt_cat.sql` and `del_stock.sql` command files, issue the following query on the **catalog** table to verify that the rows were deleted:

```
SELECT * FROM catalog WHERE stock_num = 102;
```

The **stores_demo** database has been changed. You might want to rerun the **dbaccessdemo** script to rebuild the original database.

The ins_table.sql command file

The following command file inserts one row into the **sports** table that the `c_table.sql` command file created:

```
INSERT INTO sports
   VALUES (0,18,'PARKR', 'Parker Products', '503-555-1212',
   'Heavy-weight cotton canvas gi, designed for aikido or
   judo but suitable for karate. Quilted top with side ties,
   drawstring waist on pants. White with white belt.
   Pre-washed for minimum shrinkage. Sizes 3-6.');
```

The opt_disk.sql command file

The following command file provides an example of a SELECT statement on an optical-disc subsystem. It includes the read-only **family()** and **volume()** operators that support optical storage. (This is available only with the Optical Subsystem.)

The query generates a list of the volumes that contain pictures of bicycle helmets. One row of output (family, volume) is generated for each data row that contains a picture of a bicycle helmet. The **family()** operator returns the name of the optical family where an optical blob column is stored, and **volume()** returns the number of the volume where an optical blob column is stored. These functions are valid only for data stored on optical media.

```
SELECT family(cat_picture), volume(cat_picture)
   FROM catalog
   WHERE stock_num = 110;
```

The sel_agg.sql command file

The SELECT statement in the following command file queries on table data using aggregate functions. It combines the aggregate functions MAX and MIN in a single statement.

```
SELECT MAX (ship_charge), MIN (ship_charge)
   FROM orders;
```

The sel_all.sql command file

The following example command file contains all seven SELECT statement clauses that you can use in the IBM Informix implementation of interactive SQL. This SELECT statement joins the **orders** and **items** tables. It also uses display labels, table aliases, and integers as column indicators; groups and orders the data; and puts the results into a temporary table.

```
SELECT o.order_num, SUM (i.total_price) price,
     paid_date - order_date span
   FROM orders o, items i
   WHERE o.order_date > '01/01/90'
      AND o.customer_num > 110
      AND o.order_num = i.order_num
   GROUP BY 1, 3
   HAVING COUNT (*) < 5
   ORDER BY 3
   INTO TEMP temptab1;
```

The sel_group.sql command file

The following example command file includes the GROUP BY and HAVING clauses. The HAVING clause usually complements a GROUP BY clause by applying one or more qualifying conditions to groups after they are formed, which is similar to the way the WHERE clause qualifies individual rows. (One advantage to using a HAVING clause is that you can include aggregates in the search condition; you cannot include aggregates in the search condition of a WHERE clause.)

Each HAVING clause compares one column or aggregate expression of the group with another aggregate expression of the group or with a constant. You can use the HAVING clause to place conditions on both column values and aggregate values in the group list.

```
SELECT order_num, COUNT(*) number, AVG (total_price) average
   FROM items
   GROUP BY order_num
   HAVING COUNT(*) > 2;
```

The sel_join.sql command file

The following example command file uses a simple join on the **customer** and **cust_calls** tables. This query returns only those rows that show the customer has made a call to customer service.

```
SELECT c.customer_num, c.lname, c.company,
     c.phone, u.call_dtime, u.call_descr
   FROM customer c, cust_calls u
   WHERE c.customer_num = u.customer_num;
```

The sel_ojoin1.sql command file

The following example command file uses a simple outer join on two tables. The use of the keyword OUTER in front of the **cust_calls** table makes it the subservient table. An outer join causes the query to return information about all customers, even if they do not make calls to customer service. All rows from the dominant **customer** table are retrieved, and null values are assigned to corresponding rows from the subservient **cust_calls** table.

```
SELECT c.customer_num, c.lname, c.company,
     c.phone, u.call_dtime, u.call_descr
   FROM customer c, OUTER cust_calls u
   WHERE c.customer_num = u.customer_num;
```

The sel_ojoin2.sql command file

The following example command file creates an outer join, which is the result of a simple join to a third table. This second type of outer join is called a *nested simple join*.

This query first performs a simple join on the **orders** and **items** tables, retrieving information about all orders for items with a **manu_code** of KAR or SHM. It then performs an outer join, which combines this information with data from the dominant **customer** table. An optional ORDER BY clause reorganizes the data.

```
SELECT c.customer_num, c.lname, o.order_num,
     i.stock_num, i.manu_code, i.quantity
   FROM customer c, OUTER (orders o, items i)
   WHERE c.customer_num = o.customer_num
      AND o.order_num = i.order_num
      AND manu_code IN ('KAR', 'SHM')
   ORDER BY lname;
```

The sel_ojoin3.sql command file

The following example SELECT statement is the third type of outer join, known as a *nested outer join*. It queries on table data by creating an outer join, which is the result of an outer join to a third table.

This query first performs an outer join on the **orders** and **items** tables, retrieving information about all orders for items with a **manu_code** of KAR or SHM. It then performs an outer join, which combines this information with data from the dominant **customer** table. This query preserves order numbers that the previous example eliminated, returning rows for orders that do not contain items with either manufacturer code. An optional ORDER BY clause reorganizes the data.

```
SELECT c.customer_num, lname, o.order_num,
     stock_num, manu_code, quantity
   FROM customer c, OUTER (orders o, OUTER items i)
   WHERE c.customer_num = o.customer_num
      AND o.order_num = i.order_num
      AND manu_code IN ('KAR', 'SHM')
   ORDER BY lname;
```

The sel_ojoin4.sql command file

The following example queries on table data using the fourth type of outer join. This query shows an outer join, which is the result of an outer join of each of two tables to a third table. In this type of outer join, join relationships are possible *only* between the dominant table and subservient tables.

This query individually joins the subservient tables **orders** and **cust_calls** to the dominant **customer** table but does not join the two subservient tables. (An INTO TEMP clause selects the results into a temporary table.)

```
SELECT c.customer_num, lname, o.order_num,
     order_date, call_dtime
   FROM customer c, OUTER orders o, OUTER cust_calls x
   WHERE c.customer_num = o.customer_num
      AND c.customer_num = x.customer_num
   INTO temp service;
```

The sel_order.sql command file

The following example uses the ORDER BY and WHERE clauses to query. In this SELECT statement, the comparison 'bicycle%' (LIKE condition, or 'bicycle*' for a MATCHES condition) specifies the letters bicycle followed by any sequence of zero or more characters. It narrows the search further by adding another comparison condition that excludes a **manu_code** of PRC.

```
SELECT * FROM stock
   WHERE description LIKE 'bicycle%'
      AND manu_code NOT LIKE 'PRC'
   ORDER BY description, manu_code;
```

The sel_sub.sql command file

The following example uses a subquery to query. This self-join uses a correlated subquery to retrieve and list the 10 highest-priced items ordered.

```
SELECT order_num, total_price
   FROM items a
   WHERE 10 >
      (SELECT COUNT (*)
         FROM items b
         WHERE b.total_price < a.total_price)
   ORDER BY total_price;
```

The sel_union.sql command file

The following example uses the UNION clause to query on data in two tables. The compound query performs a union on the **stock_num** and **manu_code** columns in the **stock** and **items** tables. The statement selects items that have a unit price of less than $25.00 or that have been ordered in quantities greater than three, and it lists their **stock_num** and **manu_code**.

```
SELECT DISTINCT stock_num, manu_code
   FROM stock
   WHERE unit_price < 25.00

UNION

SELECT stock_num, manu_code
   FROM items
   WHERE quantity > 3;
```

The upd_table.sql command file

The following example updates the **sports** table that the `c_table.sql` command file created:

```
UPDATE sports
   SET phone = '808-555-1212'
   WHERE mfg_code = 'PARKR';
```

SQL files for the Dimensional Database Model

The **sales_demo** database illustrates a dimensional schema for data-warehousing applications. This database model alters the **stores_demo** schema and data. The success of the files in this section requires two prerequisites:

- You must first create a **stores_demo** database with the following command:

  ```
  dbaccessdemo -log
  ```
- The `createdw.sql` and `loaddw.sql` files must be in the same directory as the files with extension `.unl` that the `loaddw.sql` uses.

The createdw.sql file

This file creates the new **sales_demo** database with logging and then creates tables within that database. It contains the following statements:

```
create database sales_demo with log;

create table product (
   product_code integer,
   product_name char(31),
   vendor_code char(3),
```

```
   vendor_name char(15),
   product_line_code smallint,
   product_line_name char(15));

create table customer (
   customer_code integer,
   customer_name char(31),
   company_name char(20));


create table sales (
   customer_code integer,
   district_code smallint,
   time_code integer,
   product_code integer,
   units_sold smallint,
   revenue money (8,2),
   cost money (8,2),
   net_profit money(8,2));

create table time
   (
   time_code int,
   order_date date,
   month_code smallint,
   month_name char(10),
   quarter_code smallint,
   quarter_name char(10),
   year integer
   );

create table geography (
    district_code serial,
   district_name char(15),
   state_code char(2),
   state_name char(18),
   region smallint);
```

The loaddw.sql file

This file contains the commands necessary to load data from two sources:

- The files with the extension `.unl` in your demonstration directory
- Data selected from the **stores_demo** database

These SQL statements in `loaddw.sql` accomplish these actions:

```
connect to "stores_demo ";
load from "add_orders.unl"
   insert into stores_demo :orders;
load from 'add_items.unl'
   insert into stores_demo :items;

connect to "sales_demo";
load from 'costs.unl'
   insert into cost;
load from 'time.unl'
   insert into time;

insert into geography(district_name, state_code, state_name)
   select distinct c.city, s.code, s.sname
from stores_demo :customer c, stores_demo :state s
   where c.state = s.code;
update geography      -- converts state_code values to region values
   set region = 1
   where state_code = "CA";
```

```
update geography
   set region = 2
   where state_code <> "CA";

insert into customer (customer_code, customer_name, company_name)
   select c.customer_num, trim(c.fname) || " " || c.lname, c.company
   from stores_demo :customer c;

insert into product (product_code, product_name, vendor_code,
   vendor_name, product_line_code, product_line_name)
 select a.catalog_num,
      trim(m.manu_name) || " "|| s.description,
      m.manu_code, m.manu_name, s.stock_num, s.description
   from stores_demo :catalog a, stores_demo :manufact m,
      stores_demo :stock s
   where a.stock_num = s.stock_num and
      a.manu_code = s.manu_code and
       s.manu_code = m.manu_code;
insert into sales (customer_code, district_code,
   time_code, product_code,
   units_sold, revenue, cost, net_profit)
   select c.customer_num, g.district_code, t.time_code, p.product_code,
   SUM(i.quantity), SUM(i.total_price),
   SUM(i.quantity * x.cost),
   SUM(i.total_price) - SUM(i.quantity * x.cost)
   from stores_demo :customer c, geography g, time t,
       product p,
      stores_demo :items i, stores_demo :orders o, cost x
    where c.customer_num = o.customer_num and
       o.order_num = i.order_num and
       p.product_line_code = i.stock_num and
      p.vendor_code = i.manu_code and
      t.order_date = o.order_date and
      p.product_code = x.product_code and
      c.city = g.district_name
      GROUP BY 1,2,3,4;

connect to "stores_demo ";
load from 'add_orders.unl'
   insert into stores_demo :orders;
load from 'add_items.unl'
   insert into stores_demo :items;

connect to "sales_demo";
load from 'costs.unl'
   insert into cost;
load from 'time.unl'
   insert into time;

insert into geography(district_name, state_code, state_name)
   select distinct c.city, s.code, s.sname
from stores_demo :customer c, stores_demo :state s
   where c.state = s.code;
update geography      -- converts state_code values to region values
   set region = 1
   where state_code = "CA";
update geography
   set region = 2
   where state_code <> "CA";

insert into customer (customer_code, customer_name, company_name)
   select c.customer_num, trim(c.fname) || " " || c.lname, c.company
   from stores_demo :customer c;

insert into product (product_code, product_name, vendor_code,
   vendor_name, product_line_code, product_line_name)
 select a.catalog_num,
      trim(m.manu_name) || " " || s.description,
      m.manu_code, m.manu_name, s.stock_num, s.description
   from stores_demo :catalog a, stores_demo :manufact m,
      stores_demo :stock s
   where a.stock_num = s.stock_num and
      a.manu_code = s.manu_code and
       s.manu_code = m.manu_code;

insert into sales (customer_code, district_code,
   time_code, product_code,
   units_sold, revenue, cost, net_profit)
   select c.customer_num, g.district_code, t.time_code, p.product_code,
   SUM(i.quantity), SUM(i.total_price),
   SUM(i.quantity * x.cost),
   SUM(i.total_price) - SUM(i.quantity * x.cost)
   from stores_demo :customer c, geography g, time t, product p,
      stores_demo :items i, stores_demo :orders o, cost x
    where c.customer_num = o.customer_num and
       o.order_num = i.order_num and
       p.product_line_code = i.stock_num and
      p.vendor_code = i.manu_code and
      t.order_date = o.order_date and
      p.product_code = x.product_code and
      c.city = g.district_name
      GROUP BY 1,2,3,4;
```

User-defined routines for the Object-relational database model

The **superstores_demo** database does not replace the **stores_demo** database. Both databases are available. The **superstores_demo** database schema is not compatible with earlier versions with **stores_demo**. In many cases, you cannot use test queries developed for **stores_demo** against the tables of **superstores_demo** because the tables differ.

No SQL command files are associated specifically with **superstores_demo**. However, there are user-defined routines that you can use with the screens described in Chapter 3, “The Query-language option,” on page 3-1.

The **superstores_demo** database includes examples of the following features:

- Collection types: SET, LIST
- Named row types: location_t, loc_us_t, loc_non_us_t
- Unnamed row types
- Type and table inheritance
- Built-in data types: BOOLEAN, SERIAL8, INT8
- Distinct data type: percent
- Smart large objects: BLOB and CLOB

The **superstores_demo** database has row types and tables to support the following table-inheritance hierarchies:

- customer/retail_customer
- customer/whlsale_customer
- location/location_us
- location/location_non_us

For a complete description of the **superstores_demo** tables and inheritance hierarchies, see the *IBM Informix Guide to SQL: Reference*. For more information about user-defined routines, see *IBM Informix User-Defined Routines and Data Types Developer's Guide*.

Appendix C. Accessibility

IBM strives to provide products with usable access for everyone, regardless of age or ability.

Accessibility features for IBM Informix products

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use information technology products successfully.

Accessibility features

The following list includes the major accessibility features in IBM Informix products. These features support:

- Keyboard-only operation.
- Interfaces that are commonly used by screen readers.
- The attachment of alternative input and output devices.

Keyboard navigation

This product uses standard Microsoft Windows navigation keys.

Related accessibility information

IBM is committed to making our documentation accessible to persons with disabilities. Our publications are available in HTML format so that they can be accessed with assistive technology such as screen reader software.

IBM and accessibility

See the *IBM Accessibility Center* at http://www.ibm.com/able for more information about the IBM commitment to accessibility.

Dotted decimal syntax diagrams

The syntax diagrams in our publications are available in dotted decimal format, which is an accessible format that is available only if you are using a screen reader.

In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), the elements can appear on the same line, because they can be considered as a single compound syntax element.

Each line starts with a dotted decimal number; for example, `3` or `3.1` or `3.1.1`. To hear these numbers correctly, make sure that your screen reader is set to read punctuation. All syntax elements that have the same dotted decimal number (for example, all syntax elements that have the number `3.1`) are mutually exclusive alternatives. If you hear the lines `3.1 USERID` and `3.1 SYSTEMID`, your syntax can include either `USERID` or `SYSTEMID`, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number `3` is followed by a series of syntax elements with dotted decimal number `3.1`, all the syntax elements numbered `3.1` are subordinate to the syntax element numbered `3`.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, the word or symbol is preceded by the backslash (\) character. The * symbol can be used next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is read as 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol that provides information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, that element is defined elsewhere. The string following the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 refers to a separate syntax fragment OP1.

The following words and symbols are used next to the dotted decimal numbers:

? Specifies an optional syntax element. A dotted decimal number followed by the ? symbol indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element (for example, 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that syntax elements NOTIFY and UPDATE are optional; that is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

! Specifies a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicates that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the same dotted decimal number can specify a ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In this example, if you include the FILE keyword but do not specify an option, default option KEEP is applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP only applies to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

***** Specifies a syntax element that can be repeated zero or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line `5.1* data-area`, you know that you can include more than one data area or you can include none. If you hear the lines `3*`, `3 HOST`, and `3 STATE`, you know that you can include `HOST`, `STATE`, both together, or nothing.

Notes:

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you can write `HOST STATE`, but you cannot write `HOST HOST`.
3. The `*` symbol is equivalent to a loop-back line in a railroad syntax diagram.

+ Specifies a syntax element that must be included one or more times. A dotted decimal number followed by the + symbol indicates that this syntax element must be included one or more times. For example, if you hear the line `6.1+ data-area`, you must include at least one data area. If you hear the lines `2+`, `2 HOST`, and `2 STATE`, you know that you must include `HOST`, `STATE`, or both. As for the * symbol, you can repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the `*` symbol, is equivalent to a loop-back line in a railroad syntax diagram.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

Trademarks

Index

Printed in USA

SC27-3535-02