



IBM Systems - iSeries
Database
Database overview

Version 5 Release 4





IBM Systems - iSeries
Database
Database overview

Version 5 Release 4

Note

Before using this information and the product it supports, read the information in “Notices,” on page 37.

Third Edition (February 2006)

This edition applies to version 5, release 4, modification 0 of IBM i5/OS (product number 5722-SS1) and to all subsequent releases and modifications until otherwise indicated in new editions. This version does not run on all reduced instruction set computer (RISC) models nor does it run on CISC models.

© Copyright International Business Machines Corporation 2004, 2006. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Database overview	1	Import and export data using the Import and Export wizards	19
What's new for V5R4	1	Get started with SQL	19
Printable PDF	1	Create a schema	20
DB2 Universal Database for iSeries	2	Create and use a table	20
Get started with iSeries Navigator	2	Use the LABEL ON statement	22
Start iSeries Navigator	2	Insert information into a table	23
Create a schema with iSeries Navigator (SAMPLELIB)	2	Get information from a single table	26
Edit the list of schemas displayed	3	Get information from multiple tables	28
Create a table and define a column	4	Change information in a table	30
Create and use a view	10	Delete information from a table.	33
Delete database objects	14	Create and use a view	33
iSeries Navigator database tasks	14	SQL versus traditional file access terminology	35
Map your database using Database Navigator maps	15	Code license and disclaimer information.	36
Query your database using the Run SQL Scripts interface	16	Appendix. Notices	37
Create and manage objects using iSeries Navigator	18	Programming Interface Information	38
Manage check pending constraints	18	Trademarks	39
		Terms and conditions	39

Database overview

| DB2 Universal Database™ for iSeries™ shares characteristics with many other implementations of DB2®.
| But if you have just migrated to iSeries, you might be wondering how DB2 UDB differs on other IBM®
| servers, or you might need to know how IBM's Universal Database compares to other relational
| databases, and what advantages iSeries brings to database development.

These links should help you understand the various strengths of iSeries as a database platform. They can help you assess which data access methods make most sense for your organization as well as build a rough framework for developing and maintaining your database implementation on iSeries.

You can also explore other database information using the main navigation bar.

Note: By using the code examples, you agree to the terms of the “Code license and disclaimer information” on page 36.

What's new for V5R4



This topic highlights changes to the IBM iSeries Navigator for V5R4.

| **New iSeries Navigator wizards for data import and export**

| In V5R4, wizards have been added to the iSeries Navigator so that you can import and export data with
| the iSeries Navigator. See “Import and export data using the Import and Export wizards” on page 19 for
| details.

How to see what's new or changed

To help you see where technical changes have been made, this information uses:

- The  image to mark where new or changed information begins.
- The  image to mark where new or changed information ends.

To find other information about what's new or changed this release, see the Memo to users.

Printable PDF

Use this to view and print a PDF of this information.


To view or download the PDF version of this document, select Database overview (about 630 KB).

Saving PDF files

To save a PDF on your workstation for viewing or printing:

1. Right-click the PDF in your browser (right-click the link above).
2. Click the option that saves the PDF locally.
3. Navigate to the directory in which you want to save the PDF.
4. Click **Save**.

Downloading Adobe Reader

You need Adobe Reader installed on your system to view or print the PDF. You can download a free copy from the Adobe Web site (www.adobe.com/products/acrobat/readstep.html) .

DB2 Universal Database for iSeries

DB2 Universal Database for iSeries is the relational database manager that is fully integrated on your iSeries. Because it is integrated on iSeries, DB2 Universal Database for iSeries is very easy to use and manage.

DB2 Universal Database for iSeries also provides a wealth of functions and features such as triggers, stored procedures, and dynamic bitmapped indexing that serve a wide variety of application types. These applications range from traditional host-based applications to client/server solutions to business intelligence applications.

As an interface to DB2 Universal Database for iSeries, the DB2 UDB Query Manager and SQL Development Kit adds an interactive query and report writing interface, as well as precompilers and tools to assist in writing SQL application programs in high-level programming languages. Conforming to the industry standard Structured Query Language (SQL), the SQL implementation for i5/OS™ operating system allows you to define, manipulate, query, and control access to your iSeries data. It works equally well with iSeries files and SQL tables.

The DB2 Universal Database for iSeries topic tells you about how to take advantage of DB2 Universal Database for iSeries to access and manage iSeries data, through an application or a user interface. Find how-to information, underlying concepts, reference information, or examples you are looking for here.

Get started with iSeries Navigator

This tutorial describes how to create and work with schemas, tables, and views using iSeries Navigator.

iSeries Navigator Database is a graphical interface that you can use to perform many of your common administrative database operations. Most of the iSeries Navigator operations are based on Structured Query Language (SQL), but you do not need to fully understand SQL to perform them.

Related concepts

“iSeries Navigator database tasks” on page 14

This topic describes the database tasks you can perform using the iSeries Navigator interface.

Related information

Getting to know iSeries Navigator

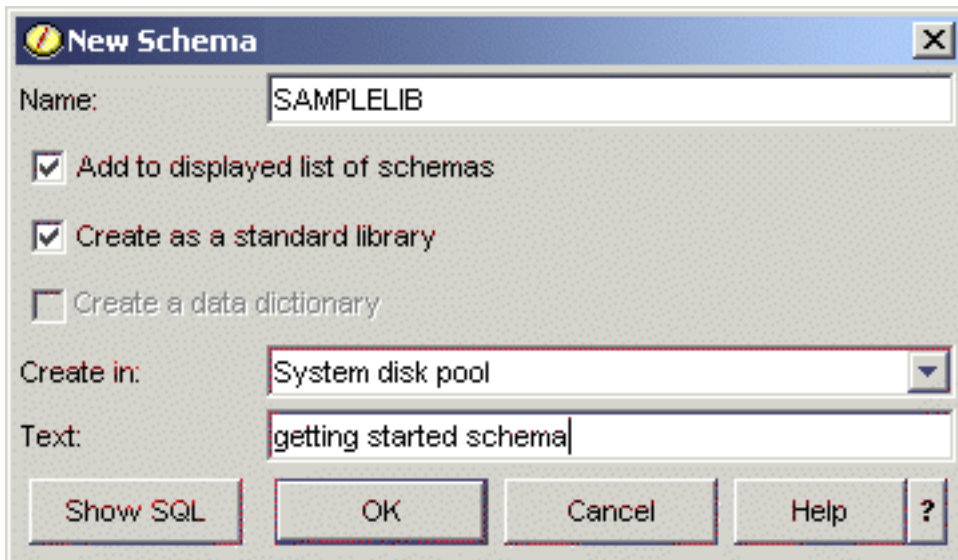
Start iSeries Navigator

1. Double-click the **iSeries Navigator** icon.
2. Expand the system you want to use.

Create a schema with iSeries Navigator (SAMPLELIB)

1. In the iSeries Navigator window, expand the system that you want to use.
2. Expand **Databases** and the database that you want to work with.
3. Right-click **Schemas**, and select **New Schema**.
4. On the New Schema window, type SAMPLELIB in the name field.
5. To add to the list of schemas to be displayed, select **Add to displayed list of schemas**.
6. Select **Create as a standard library**.

7. Specify a disk pool to contain the schema. Choose one so that the schema is created on the system disk pool.
8. **Optional:** Specify a description.
9. Click **OK**.



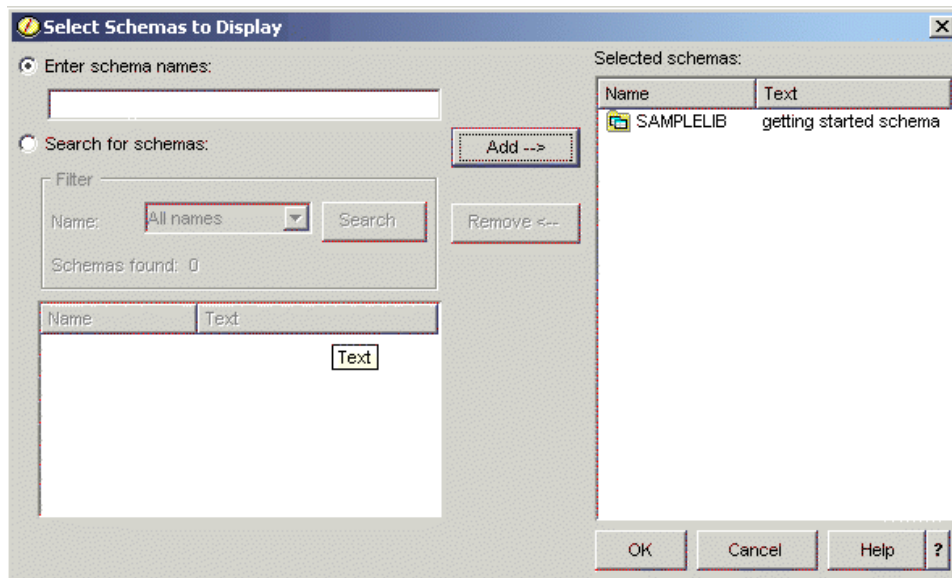
Note: See Working with multiple databases for more information about creating schemas in user disk pools.

Edit the list of schemas displayed

After you have successfully created a schema, you can create tables, views, indexes, stored procedures, user-defined function, and user-defined types in it.

To edit the list of schemas displayed when you click **Schemas**:

1. Right-click **Schemas**, and select **Select Schemas to Display**.
2. On the Select Schemas to Display window, you can edit the list by selecting **Enter schema names** and specifying a schema, or by selecting **Search for schemas** and performing a search. Select the schema you want to display, then click **Add**.
3. You can remove a schema from the list of schemas to display by selecting that schema from the list of schemas to display and clicking **Remove**.



4. For now, leave SAMPLELIB as the schema displayed.

Create a table and define a column

A table is a basic database object that is used to store information. After you have created a table, you can define columns, create indexes, and add triggers and constraints by using the Table Properties window.

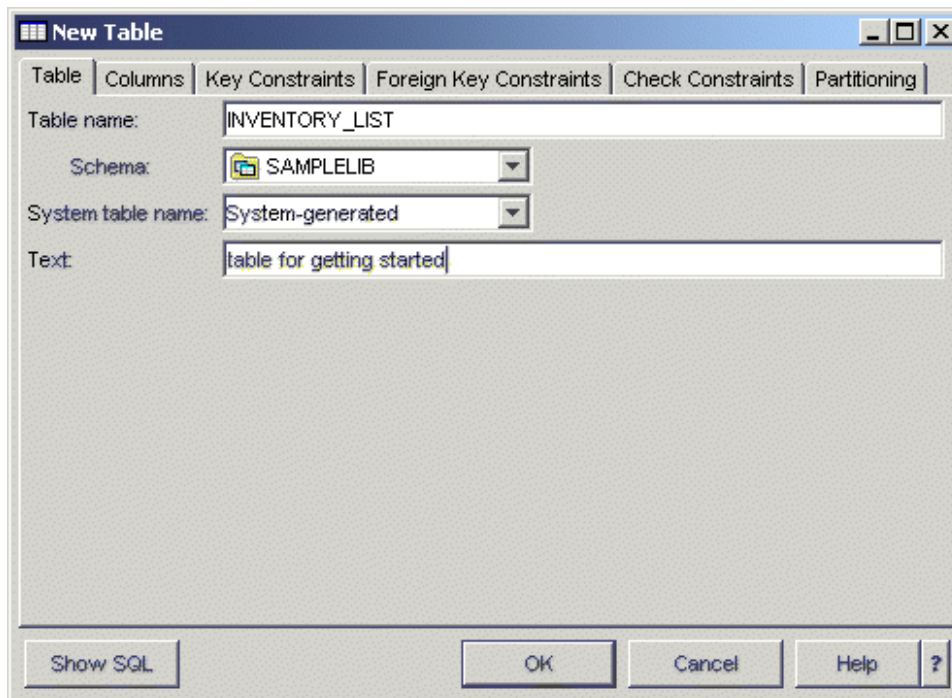
When you are creating a table, you need to understand the concepts of null value and default value. A null value indicates the absence of a column value for a row. It is not the same as a value of zero or all blanks. It means *unknown*. It is not equal to any value, not even to other null values. If a column does not allow the null value, a value must be assigned to the column. This value is either a default value or a user supplied value.

If no value is specified for a column when a row is added to a table, the row is assigned a default value. If the column is not assigned a specific default value, the column uses the system default value.

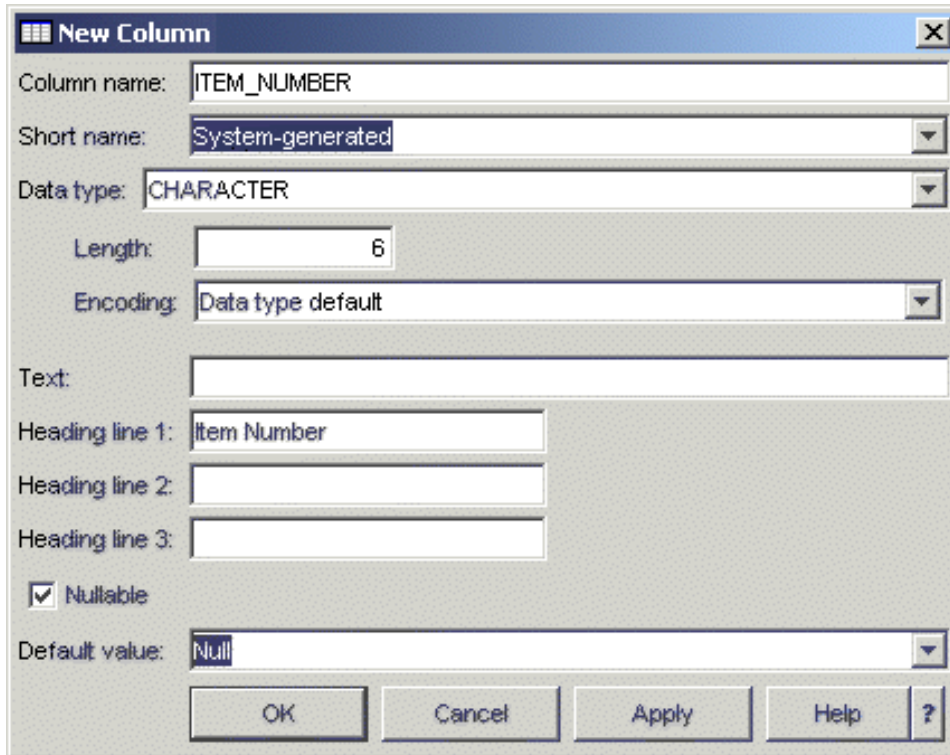
This example shows you how to create a table to maintain information about the current inventory of a business. It has information about the items kept in the inventory, their cost, quantity currently on hand, the last order date, and the number last ordered. The item number is a required value. It cannot be null. The item name, quantity on hand, and order quantity have user-supplied default values. The last order date and quantity allow the null value.

To create a table, follow these steps:

1. In the iSeries Navigator window, expand the system that you want to use.
2. Expand **Databases** and the database that you want to work with.
3. Expand **Schemas**.
4. Right-click **SAMPLELIB** and select **New**.
5. Select **Table** → **Table**.
6. On the New Table window, specify INVENTORY_LIST as the table name.
7. Select **SAMPLELIB** in the **Schema** field.
8. Select **System-generated** in the **System table name** field.
9. Specify a description in the **Text** field (optional).



10. Next, define a column for the new table. Click the **Columns** tab.
11. Click the **Add** button.
12. Enter ITEM_NUMBER in the **Column name** field.
13. You can specify a short name in the **Short column name** field. If you do not specify a short name, the system automatically generates a name. If the column name is 10 characters or less, then the short name is the same as the column name. You can perform queries by using either column name. Just leave this space as the default, **System-generated**, for now.
14. Select **CHARACTER** as the Data type.
15. Specify a length of 6 for this column. For data types where the size is predetermined, the size is filled in and you cannot change the value.
16. Leave the **Encoding** option as the default, **Data type default**.
17. You can specify a description for the column in the **Text** field. This step is optional.
18. Enter a column heading in the **Heading** fields. The heading is the label that appears at the top of the column for displaying or printing. You are limited to 60 characters, 20 per line.
19. Deselect the **Nullable** option. This ensures that a value must be placed in this column in order for the row insert to be successful.
20. In the **Default value** field, enter 0.
21. Click **OK** to create the table.



The new table INVENTORY_LIST appears.

Define additional columns on a table

You can define columns on a new or existing table. To add columns to the table you just created, navigate to the table INVENTORY_LIST by expanding **Database** → **Schemas** → **SAMPLELIB** → **Tables**. In the detail pane, right-click the table INVENTORY_LIST and select **Definition**.

1. To define a column on the Table Definition window, select the **Columns** tab.
2. Click **Add**.
3. Add the following columns to Table INVENTORY_LIST:

Column name	Type	Length	Precision	Scale	Nullable	Default value
ITEM_NAME	VARCHAR	20			No	UNKNOWN
UNIT_COST	DECIMAL		8	2	No	0
QUANTITY_ON_HAND	SMALLINT				Yes	NULL
LAST_ORDER_DATE	DATE				Yes	NULL
ORDER_QUANTITY	SMALLINT				Yes	20

When you finish defining these columns, click **OK** to create the table.

Create the supplier table (SUPPLIERS)

Later in our examples, you will need a second table. This table will contain information about suppliers of our inventory items, which items they supply, and the cost of the item from that supplier.

Create a table called SUPPLIERS in SAMPLELIB. This table will have three columns: SUPPLIER_NUMBER, ITEM_NUMBER, and SUPPLIER_COST.

Notice that this table has a common column with table INVENTORY_LIST: ITEM_NUMBER. Rather than create a new ITEM_NUMBER column, you can copy the column definition used for ITEM_NUMBER in INVENTORY_LIST table.

Copy column definitions

To copy column definitions, follow these steps:

1. On the SUPPLIER Table Properties or the New Table window, click **Browse**.
2. On the Browse Tables window, expand **SAMPLELIB**.
3. Click **INVENTORY_LIST**. The columns in that table are listed, along with their data type, size, and description.
4. Select **ITEM_NUMBER**.
5. Click **Add** to copy this column definition to table SUPPLIERS.
6. Close the Browse Columns window.

Add the last two columns for table SUPPLIERS with the following values:

Column name	Type	Length	Precision	Scale	Nullable	Default value
SUPPLIER_NUMBER	CHAR	4			No	0
SUPPLIER_COST	DECIMAL		8	2	Yes	NULL

Insert information into a table

To insert, edit, or delete data in a table, you must have authority to that table. To add data to the table INVENTORY_LIST:

1. In the iSeries Navigator window, expand the system that you want to use.
2. Expand **Databases** and the database that you want to work with.
3. Expand **Schemas**.
4. Select **SAMPLELIB**.
5. Double-click **Tables**.
6. Right-click INVENTORY_LIST and select **Edit Contents**.
7. From the Rows menu, select **Insert**. A new row appears.
8. Enter the information from the table below under the appropriate headings.

Note: The values you enter must satisfy all constraints and satisfy the type of each column. If there is a unique constraint or index over the table, the values you enter must define a unique key value. If you do not enter a value in a column, the default value will be entered, if allowed. For this exercise, insert only those values shown in the table below so that the default values are used.

ITEM_NUMBER	ITEM_NAME	UNIT_COST	QUANTITY_ON_HAND
153047	Pencils, red	10.00	25
229740	Lined tablets	1.50	120
544931		5.00	
303476	Paper clips	2.00	100
559343	Envelopes, legal	3.00	500
291124	Envelopes, standard		
775298	Chairs, secretary	225.00	6
073956	Pens, black	20.00	25

From the **File** menu, select **Save**.

9. Add the following rows to the SAMPLELIB.SUPPLIERS table.

ITEM_NUMBER	SUPPLIER_NUMBER	SUPPLIER_COST
153047	1234	10.00
229740	1234	1.00
303476	1234	3.00
153047	9988	8.00
559343	9988	3.00
153047	2424	9.00
303476	2424	2.50
775298	5546	225.00
303476	3366	1.50
073956	3366	17.00

10. From the **File** menu, select **Save**. The sample schema now contains two tables with several rows of data in each.

View the contents of a table

You can display the contents of your tables and views. You can only view the contents. To make changes to a table, you must edit the table.

To view the contents of INVENTORY_LIST, follow these steps:

1. In the iSeries Navigator window, expand the system that you want to use.
2. Expand **Databases** and the database that you want to work with.
3. Expand **Schemas**.
4. Select **Tables**.
5. Click SAMPLELIB.
6. Right-click INVENTORY_LIST and select **View Contents**.

ITEM_NUMBER	ITEM_NAME	UNIT_COST	QUANTITY_ON_HAND	LAST_ORDER_DATE	ORDER_QUANTITY
153047	PENCILS, RED	10.00	25		20
229740	LINED TABLETS	1.50	120		20
303476	PAPER CLIPS	2.00	100		20
559343	ENVELOPES, LEGAL	3.00	500		20
544931	UNKNOWN	5.00			20
775298	CHAIRS, SECRETARY	225.00	6		20
073956	PENS, BLACK	20.00	25		20
291124	ENVELOPES, STANDARD	0			20

Change information in a table

You can use iSeries Navigator to change the data values in the columns of a table. Suppose you want to update a column using iSeries Navigator to indicate that you received an order for more paper clips today. Keep in mind that the value you enter must be valid for that column.

1. Navigate to table INVENTORY_LIST. Right-click the table and select **Edit Contents**.
2. Enter the current date in the LAST_ORDER_DATE column for the row Paper clips. Be sure to use the correct date format for your system.
3. Change the ORDER_QUANTITY to 50.
4. Save the changes, then view the table contents by using **View Contents**.

The paper clip row reflects the changes you made.

Delete information from a table

You can delete data from a table by using iSeries Navigator. You can delete information from a single column in a row or delete the row entirely. Keep in mind that if a column requires a value, you will not be able to delete it without deleting the entire row.

1. Open table INVENTORY_LIST by double-clicking on it.
2. Delete the column value for ORDER_QUANTITY for the **Envelopes, standard** row. Because this is a column that allows Null values, you can delete the value.
3. Delete the column value for UNIT_COST for the **Lined tablets** row. Because this column does not allow Null values, the deletion is not allowed.

You can also delete an entire row without removing all of the column values one at a time.

1. Open table INVENTORY_LIST by double-clicking it.
2. Click the cell to the left of the UNKNOWN row. This highlights the entire row.
3. Select **Delete** from the **Rows** menu or press the Delete key on your keyboard. The UNKNOWN row is deleted.
4. Delete all of the rows from table INVENTORY_LIST that do not have a value in the QUANTITY_ON_HAND column.
5. Save the changes and view the contents by using **View Contents**.

You should have a table that contains the following data:

ITEM_NUMBER	ITEM_NAME	UNIT_COST	QUANTITY_ON_HAND	LAST_ORDER_DATE	ORDER_QUANTITY
153047	Pencils, red	10.00	25		20
229740	Lined tablets	1.50	120		20
303476	Paper clips	2.00	100	2003-09-22	50
559343	Envelopes, legal	3.00	500		20
775298	Chairs, secretary	225.00	6		20
073956	Pens, black	20.00	25		20

Copy and move a table

iSeries Navigator allows you to copy or move tables from one schema or system to another. By copying a table, you create more than one instance of the table. By moving a table, you transfer the table to its new location while removing the instance from its former location.

Copy a table:

Create a new schema called LIBRARY1 and add it to the list of schemas displayed. After you have created this new schema, copy INVENTORY_LIST over to LIBRARY1. To copy a table:

1. In the iSeries Navigator window, expand the system that you want to use.
2. Expand **Databases** and the database that you want to work with.

3. Expand **Schemas**.
4. Double-click **Tables**.
5. Click SAMPLELIB.
6. Right-click INVENTORY_LIST and select **Copy**.
7. Right-click LIBRARY1 and select **Paste**.

Move a table:

Now that you have copied table INVENTORY_LIST to LIBRARY1, move table SUPPLIERS to LIBRARY1. To move a table:

1. In the iSeries Navigator window, expand the system that you want to use.
2. Expand **Databases** and the database that you want to work with.
3. Expand **Schemas**.
4. Double-click **Tables**.
5. Click SAMPLELIB.
6. Right-click SUPPLIERS and select **Cut**.
7. Right-click LIBRARY1 and select **Paste**.

Note: You can move a table by dragging and dropping the table on the new schema. Moving a table to a new location does not always remove it from the source system. For example, if you have read authority but not delete authority to the source table, you can move the table to the target system. However, you cannot delete the table from the source system, causing two instances of the table to exist.

Create and use a view

You might find that no single table contains all the information you need. You might also want to give users access to only part of the data in a table. Views provide a way to divide the table so that you deal with only the data you need. A view reduces complexity and, at the same time, restricts access.

In order to create a view, you must have the correct authority to the tables or physical files on which the view is based. See the CREATE VIEW statement in the SQL Reference topic for a list of authorities needed.

If you did not specify column names in the view definition, the column names are the same as those for the table on which the view is based.

You can make changes to a table through a view even if the view has a different number of columns or rows than the table. For INSERT, columns in the table that are not in the view must have a default value.

You can use the view as though it were a table, even though the view is totally dependent on one or more tables for data. The view has no data of its own and therefore requires no storage for the data. Because a view is derived from a table that exists in storage, when you update the view data, you are really updating data in the table. Therefore, views are automatically kept up-to-date as the tables they depend on are updated.

Create a view over a single table

The following example shows how to create a view on a single table. The view is built on the INVENTORY_LIST table. The table has six columns, but the view uses only three of the columns: ITEM_NUMBER, LAST_ORDER_DATE, and QUANTITY_ON_HAND.

To create a view over a single table:

1. In the iSeries Navigator window, expand the system that you want to use.
2. Expand **Databases** and the database that you want to work with.
3. Expand **Schemas**.
4. Right-click SAMPLELIB and select **New**, then **View**.
5. On the New View window, type RECENT_ORDERS in the **Name** field.
6. Specify SAMPLELIB in the **Schema** field.
7. Optionally, you can specify a description.
8. Select a check option. A check option on a view specifies that the values inserted or updated into a row must conform to the conditions of the view. For this view, select **None**.
9. Click **OK**. The New View definition window appears.

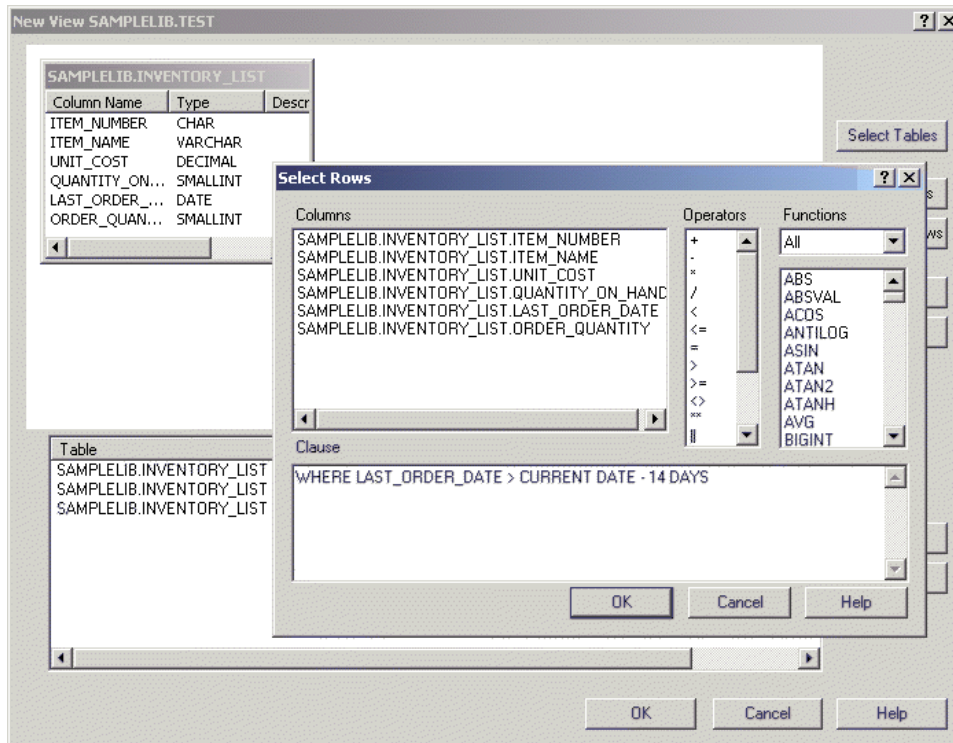


10. On the New View window, click **Select tables**.
11. On the Browse for Tables window, expand SAMPLELIB, then select INVENTORY_LIST.
12. Click **Add**.
13. Click **OK**. INVENTORY_LIST should now be in the work area on the New View window.
14. To choose the columns that you want in the new view, click them in the selected tables and drag-and-drop them in the selection grid on the bottom half of the window. Select ITEM_NUMBER, LAST_ORDER_DATE, and QUANTITY_ON_HAND.
15. The order that the columns appear in the selection grid is the order that they will appear in the view. To change the order, select a column and drag it to its new position. Put the columns in the following order: ITEM_NUMBER, LAST_ORDER_DATE, QUANTITY_ON_HAND.

Create a WHERE clause

The view is now essentially finished, but for this example, you only want to view those items that have been ordered in the last 14 days. To specify this information, you need to create a WHERE clause:

1. Click **Select Rows**.
2. On the Select Rows window, enter the following command: WHERE LAST_ORDER_DATE > CURRENT DATE - 14 DAYS. You can select the elements that make up this WHERE clause by selecting them from the options shown, or you can enter them in the **Clause** field.
3. Click **OK**.
4. To view the SQL used to generate this view, click **Show SQL**.
5. Click **OK** to create the view.



- To display the contents of RECENT_ORDERS, right-click RECENT_ORDERS and select **View Contents**.

You should see the following information displayed:

ITEM_NUMBER	LAST_ORDER_DATE	QUANTITY_ON_HAND
303476	2003-09-22	100

In the preceding example, the columns in the view have the same name as the columns in the table because you did not specify new names. The schema that the view is created into does not need to be the same schema as the table it is built over. You can use any schema.

Create a view combining data from multiple tables

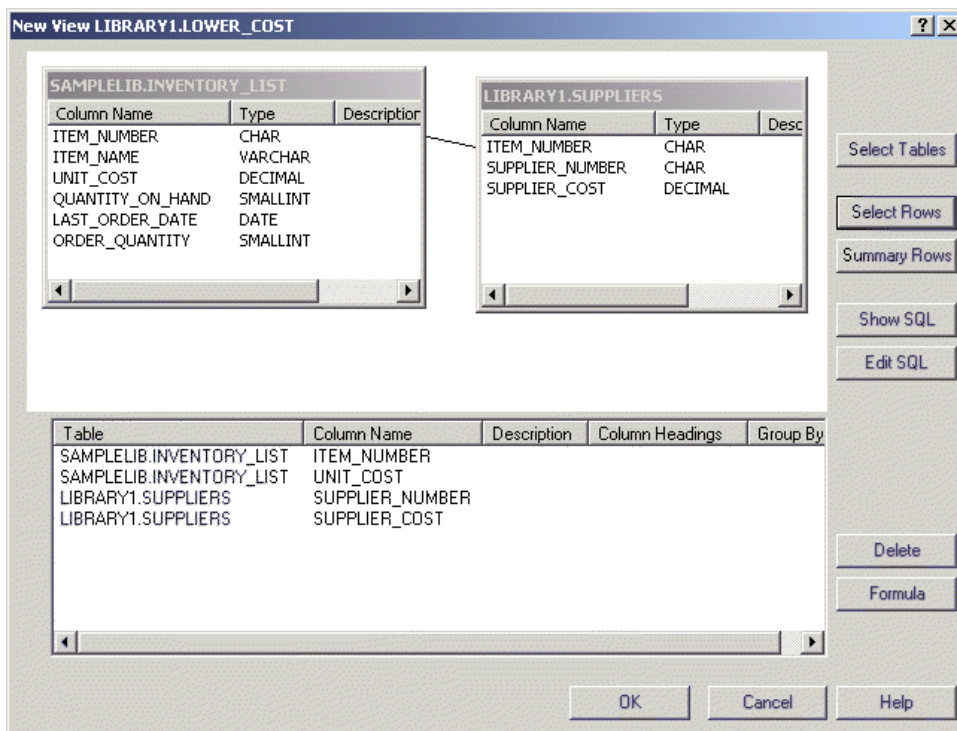
You can create a view combining information from more than one table by selecting more than one table in the work area of the New View window. You can create a simple view from more than one table by selecting the columns that you want to include from different tables and clicking **OK**. However, this example shows how to create a view that joins information from two different tables and returns only those rows that you want to see, much like using a WHERE clause.

In this example, you create a view that contains only those item numbers for suppliers that can supply an item at lower cost than the current unit cost. This requires selecting ITEM_NUMBER and UNIT_COST from the INVENTORY_LIST table and joining them with SUPPLIER_NUMBER and SUPPLIER_COST from the SUPPLIERS table. A WHERE clause is used to limit the number of rows returned.

To create a view called LOWER_COST:

- Navigate to schema LIBRARY1. Right-click **Views** and select **New**.
- Select INVENTORY_LIST from SAMPLELIB and SUPPLIERS from LIBRARY1.
- Click **OK**. Both tables should appear in the working area of the window.
- Select ITEM_NUMBER and UNIT_COST from INVENTORY_LIST.

5. Select SUPPLIER_NUMBER and SUPPLIER_COST from SUPPLIERS.
6. To define the join, select ITEM_NUMBER from INVENTORY_LIST and drag it to ITEM_NUMBER in SUPPLIERS. A line is drawn from one column to the other and the Join window opens.
7. On the Join window, select **Return rows with a matching condition (Inner Join)**.
8. Click **OK**.
9. Click **Select Rows** to create a WHERE clause for the view. Double-click LIBRARY1.SUPPLIERS.SUPPLIER_COST, then double-click the < operator and finally double-click SAMPLELIB.INVENTORY_LIST.UNIT_COST . As you click the items, they appear in the window. You can also type this in directly.
10. Click **OK** to create the view, LOWER_COST.



Note: You can view the SQL used to create this view by selecting **Show SQL**. You can also edit the SQL by selecting **Edit SQL**. Edit SQL starts Run SQL Scripts, where you can edit your SQL statement. Be aware, however, that if you change the SQL, you will need to run the statement from Run SQL Scripts rather than returning to the New View window. If you return to the New View window, your changes are not saved.

To display the contents of this new view, right-click LOWER_COST and select **View Contents**. The rows that you see through this view are only those rows that have a supplier cost that is less than the unit cost.

ITEM_NUMBER	UNIT_COST	SUPPLIER_NUMBER	SUPPLIER_COST
153047	10.00	9988	8.00
153047	10.00	2424	9.00
229740	1.50	1234	1.00
303476	2.00	3366	1.50
073956	20.00	3366	17.00

Delete database objects

After you have created these objects on your system, you might want to drop them to save on system resource. You need Delete authority to perform these tasks.

Note: To keep the information in these tables, create a third schema and copy the tables and views to it.

1. First, drop INVENTORY_LIST table from LIBRARY1:
 - a. In the iSeries Navigator window, expand the system that you want to use.
 - b. Expand **Databases** and the database that you want to work with.
 - c. Expand **Schemas** and select LIBRARY1.
 - d. Select **Tables**.
 - e. Right-click INVENTORY_LIST and select **Delete** or press the Delete key.
 - f. On the Object deletion confirmation window, select **Delete**. INVENTORY_LIST table is dropped.
2. Next, delete SUPPLIERS from LIBRARY1, and delete LIBRARY1:
 - a. Right-click SUPPLIERS and select **Delete** or press the Delete key.
 - b. On the Object deletion confirmation window, select **Yes**.
 - c. A new window opens, indicating that the view, LOWER_COST, is dependent on SUPPLIERS. The view should also be deleted. Click **Delete**.
 - d. SUPPLIERS and LOWER_COST are deleted. Now that LIBRARY1 is empty, delete it by right-clicking it and selecting **Delete**.
 - e. On the Object deletion confirmation window, select **Yes**. LIBRARY1 is deleted.
3. Finally, delete SAMPLELIB:
 - a. Navigate to SAMPLELIB in the **Schemas** menu.
 - b. Right-click SAMPLELIB and select **Delete**.
 - c. On the Object deletion confirmation window, select **Delete**.
 - d. A new window opens, indicating that the table INVENTORY_LIST and view RECENT_ORDERS are dependent on INVENTORY_LIST. These should also be deleted. Click **Yes**.

SAMPLELIB, INVENTORY_LIST, and RECENT_ORDERS are deleted.

Related information

iSeries Navigator tasks

iSeries Navigator database tasks

This topic describes the database tasks you can perform using the iSeries Navigator interface.

In addition to the tasks described in the “Get started with iSeries Navigator” on page 2 topic, there are many other ways to use iSeries Navigator with your DB2 Universal Database for iSeries. See the following links for information about how to use iSeries Navigator with your database:

Tasks in the Database programming topic:

Other tasks that are included in the Database programming topic include:

- Adding triggers
- Authorizing a user or group to files
- Copying a file (table)
- Creating a schema
- Defining public authority for a file
- Displaying attributes for a file (table)

- Displaying locked rows
- Moving a file (table)
- Reorganizing a file (table)
- Setting a default public authority for new files
- Working with journals:
 - Creating a journal
 - Creating a journal receiver
 - Adding a remote journal
 - Removing a remote journal
 - Activating a remote journal
 - Deactivating a remote journal
 - Displaying journal information
 - Swapping journal receivers
 - Starting and stopping a journal

Tasks in the Database performance and query optimization topic

Tasks in the Database performance and query optimization topic include:

- Examine debug messages in the job log
- Gather information about embedded SQL statements using PRTSQLINF
- Monitor your queries using Start Database Monitor (STRDBMON)
- Monitor your database with the memory-resident database monitor
- View implementation of your queries using Visual Explain
- Change the attributes of your queries with the Change Query Attributes (CHGQRYA) command
- Collect statistics with the Statistics Manager
- | • Display access plan in Query Plan Cache
- | • Display index advisor information
- | • Display information with Database Health Monitor
- | • Determining unnecessary indexes
- | • Manage index rebuilds

Related tasks

“Get started with iSeries Navigator” on page 2

This tutorial describes how to create and work with schemas, tables, and views using iSeries Navigator.

Map your database using Database Navigator maps

Database Navigator enables you to visually represent the relationships of database objects on your system. This representation is called a map. In essence, the Database Navigator Map is a snapshot of your database and the relationships that exist among all of the objects in the map.

Using Database Navigator, you can explore the complex relationships of your database objects using a graphical representation that presents the tables in your database, the relationships between tables, and indexes and constraints that are attached to tables. The primary workspace for Database Navigator is a window that is divided into several main areas. The map is displayed in the right pane. You can perform a variety of tasks by right-clicking an object. The Locator pane is found on the left side of the window. You can use this pane to locate specific objects to include in the map or to specify a type of object to include in the map.

1. You can use Database Navigator Maps by expanding the system name, **Databases**, and the database that you want to use.
2. To display a list of existing maps in the right pane, click **Database Navigator Maps** to display a list of existing maps in the right pane.
3. To create a new map, right-click **Database Navigator Maps** and select **New** → **Map**.

Tips for using Database Navigator:

- To change the size of either side of the window, drag the bar (splitter) that separates the two sides.
- Be sure to right-click the objects in both the left and right sides of the window. The right-click menus give you quick access to common functions.
- To quickly open a schema and display the objects in it, double-click the schema.
- To access the various Database Navigator commands use either the Menu bar or the Toolbar.

Query your database using the Run SQL Scripts interface

The Run SQL Scripts window in iSeries Navigator allows you to create, edit, run, and troubleshoot scripts of SQL statements. When you have finished working with the scripts, you can save them to your PC.

You can start the Run SQL Scripts interface by expanding the system name, **Databases**, and right-clicking the database that you want to connect to.

You can use the Examples list to build your scripts, manually create your statement, retrieve the SQL for an existing object using the generate SQL function, or build a script using SQL Assist.

You can check the syntax of your SQL by clicking **Check Syntax**. Additional ways of debugging your programs and scripts include debugging messages in the job log and starting the iSeries System debugger. When syntax checking is complete, you can save the script by selecting **Save** from the **File** menu.

To run an SQL script, select one of the following options from the **Run** menu:

- **All** - Runs your SQL script from the beginning to the end. If an error occurs and the **Stop on Error** option is turned on, the program stops and the statement where the error occurred remains selected.
- **From Selected** - Starts your SQL script from the first statement that is selected or from the current cursor position and continues to the end of the script.
- **Selected** - Runs the statements that are selected.

The results are added to the end of the **Messages** tab. If the **Smart Statement Selection** option on the **Options** menu is not checked, the text that is selected is run as a single SQL statement.

Related tasks

“Build SQL statements with SQL Assist” on page 17

“View the job log” on page 17

“Start the iSeries System Debugger” on page 18

“Create a view combining data from multiple tables” on page 34

Related information

Using interactive SQL

Stop Run SQL Scripts

To stop or cancel an SQL scripts run, select one of the following options from the **Run** menu:

- **Stop After Current** - Stops running the SQL script after the currently running statement ends.
- **Cancel Request** - Requests that the system cancel the current SQL statement. However, as not all SQL statements can be canceled, the SQL statement might continue to completion even after this option is

used. SQL statements that have already completed host processing before Cancel Request is pressed will also continue to completion. For example, Select statements that have already completed query processing but have not yet returned the results to the client typically cannot be canceled.

View the job log

The job log displays messages that are related to your job.

To see query optimizer and other database debugging messages, follow these steps:

1. Select **Include Debug Messages in Job Log** from the **Options** menu.
2. Run the statement again.
3. If the Job Log dialog box is open when you do this, refresh the view to see new messages.

To view the job log, Select **Job Log** from the **View** menu.

The job log is not cleared when **Clear Run History** is used, so you can use the job log to see messages that are no longer in the Output pane.

Related tasks

“Query your database using the Run SQL Scripts interface” on page 16

The Run SQL Scripts window in iSeries Navigator allows you to create, edit, run, and troubleshoot scripts of SQL statements. When you have finished working with the scripts, you can save them to your PC.

Generate SQL for objects

- | The Generate SQL function allows you to reconstruct the SQL used to create existing database objects.
 - | This process is often referred to as *reverse engineering*. You can generate SQL for most database objects.
 - | Additionally, if you generate SQL for a table that has constraints or triggers associated with it, the SQL will be generated for those as well. You can generate the SQL for one object or many at a time. You also
 - | have the option of sending the generated SQL to the Run SQL Scripts window for running or editing, or
 - | you can write the generated SQL directly to a database or PC file.
- | To generate SQL for an object, right-click the object and select **Generate SQL**.
 - | You can also start the Generate SQL interface from Run SQL Scripts by selecting **Insert Generated SQL**
 - | from the **Edit** menu.

Build SQL statements with SQL Assist

You can build your SQL statements interactively with the SQL Assist function. SQL Assist helps you to build select, insert, update, and delete statements.

1. To start SQL Assist, select **SQL Assist** from the **Edit** menu in Run SQL Scripts. From the SQL Assist interface, you can choose tables to work with and build selection criteria. The statement is built in the bottom portion of the interface.
2. Click **OK** to return the statement you built to the Run SQL Scripts interface.
3. You can edit, run, and save your statement.

Related tasks

“Query your database using the Run SQL Scripts interface” on page 16

The Run SQL Scripts window in iSeries Navigator allows you to create, edit, run, and troubleshoot scripts of SQL statements. When you have finished working with the scripts, you can save them to your PC.

Start the iSeries System Debugger

The iSeries System Debugger provides a new graphical user debugging environment on the iSeries server. You can use iSeries System Debugger to debug and test programs that run on your iSeries server, including those that run in the i5/OS PASE environment.

To start the System Debugger from the Run SQL Scripts interface, select **Debugger** from the **Run** menu.

Related tasks

“Query your database using the Run SQL Scripts interface” on page 16

The Run SQL Scripts window in iSeries Navigator allows you to create, edit, run, and troubleshoot scripts of SQL statements. When you have finished working with the scripts, you can save them to your PC.

Related information

iSeries System Debugger

Create and manage objects using iSeries Navigator

You can create and manage many objects in iSeries Navigator.

You can create and manage the following objects:

- Schemas
- Tables- including materialized query tables and partitioned tables.
- Aliases
- Sequences
- SQL Packages
- User-defined functions (UDFs)
- User-defined distinct types (UDTs)
- Procedures
- Indexes
- Triggers
- Constraints
- Views
- Journals
- Journal Receivers

Most objects are created from the **Schema** container object. To navigate to the schema container, follow these steps:

- Expand the system name, **Databases**, and the database that you want to use. Expand **Schemas**, right-click the schema that you want to work with, and select **New**
- Select the type of object that you want to create.
- Alternately, you can expand the schema that you want to work with and right-click the container type that you want to create. Schemas are created from the **Schemas** container. SQL packages are created at the system level. Right-click the system name and select **New** → **SQL package**.

Manage check pending constraints

You can view and change constraints that have been placed in a check pending state by the system. Check pending refers to a state in which a mismatch exists between either a parent and foreign key in the case of a referential constraint or between the column value and the check constraint definition in the case of a check constraint.

To view constraints that have been placed in a check pending state, follow these steps:

1. Expand the system name and **Databases**. Right-click the database that you want to use, and select **Manage check pending constraints**.
2. From this interface, you can view the definition of the constraint and the rows that are in violation of the constraint rules. Select the constraint that you want to work with and then select **Edit Check Pending Constraint** from the **File** menu.
3. You can either alter or delete the rows that are in violation.

Related information

Check pending status in referential constraints

Import and export data using the Import and Export wizards

You can import and export data using the Import and Export wizards in iSeries Navigator. These wizards use the Copy from Import File (CPYFRMIMPF) and the Copy to Import File (CPYTOIMPF) commands to process this request.

Files and database tables that you can import data from or export data to include:

- Integrated file systems files
- Source physical files
- Program described files
- Database tables with a single non-numeric column. The column cannot be a LOB data type

The iSeries Navigator refers to an import file as a data file.

To start the Import or Export Wizard, follow these steps:

1. In the iSeries Navigator window, expand the system you want to use.
2. Expand **Databases**.
3. Expand the database and schema that you want to work with.
4. Click the **Tables** container.
5. If you want to import data from a data file, right-click the table that you want to import data to and select **Data** → **Import**. If you want to export data in a table to a file, right-click the table that you want to export data to and select **Data** → **Export**.

Related information

- Copy From Import File (CPYFRMIMPF) command
- Copy To Import File (CPYTOIMPF) command
- Copying between different servers

Get started with SQL

This topic describes how to create and work with schemas, tables, and views using SQL statements in interactive SQL.

The syntax for each of the SQL statements used in this chapter is described in detail and descriptions of how to use SQL statements and clauses in more complex situations are provided in the SQL Reference topic.

In this article, the examples use the interactive SQL interface to show the use of SQL statements. Each SQL interface provides methods for using SQL statements to define tables, views, and other objects, methods for updating the objects, and methods for reading data from the objects.

First, start interactive SQL:

1. Type STRSQL NAMING(*SQL).
2. Press Enter.

When the Enter SQL Statements display appears, you are ready to start typing SQL statements.

If you are reusing an existing interactive SQL session, make sure that you set the naming mode to **SQL naming**. You can specify this on the F13 (Services) panel, option 1 (Change session attributes).

Related information

SQL Programming

Create a schema

A schema is the basic object in which tables, views, indexes, and packages are placed. For more information about creating a schema, see SQL CREATE SCHEMA statement in the SQL Reference topic.

Note: The term *collection* can be used synonymously with schema.

To create a sample schema named SAMPLECOLL, follow these steps:

1. Enter the following SQL statement on the Enter SQL Statements panel:

Enter SQL Statements

Type SQL statement, press Enter.
Current connection is to relational database SYSTEM1

==> **CREATE SCHEMA SAMPLECOLL** _____

Bottom

F3=Exit F4=Prompt F6=Insert line F9=Retrieve F10=Copy line
F12=Cancel F13=Services F24=More keys

2. Press Enter.

Note: Running this statement causes several objects to be created and takes several seconds.

After you have successfully created a schema, you can create tables, views, and indexes in it. Tables, views, and indexes can also be created in libraries instead of schemas.

Create and use a table

You can create a table by using the SQL CREATE TABLE statement. The CREATE TABLE statement allows you to create a table, define the physical attributes of the columns in the table, and define constraints to restrict the values that are allowed in the table.

When creating a table, you need to understand the concepts of null value and default value. A null value indicates the absence of a column value for a row. It is not the same as a value of zero or all blanks. It means "unknown." It is not equal to any value, not even to other null values. If a column does not allow the null value, a value must be assigned to the column, either a default value or a user supplied value.

A default value is assigned to a column when a row is added to a table and no value is specified for that column. If a specific default value was not defined for a column, the system default value is used. For more information about the default values used by INSERT, see the SQL Reference.

You are going to create a table to maintain information about the current inventory of a business. The table have information about the items kept in the inventory, their cost, quantity currently on hand, the last order date, and the number last ordered. The item number is a required value. It cannot be null. The item name, quantity on hand, and order quantity have user supplied default values. The last order date and quantity ordered allow the null value.

You also create a second table. This table contain information about suppliers of your inventory items, which items they supply, and the cost of the item from that supplier.

1. Create the first table named INVENTORY_LIST:

- a. On the Enter SQL Statements display, type CREATE TABLE and press F4 (Prompt). The following display is shown (with the input areas not yet filled in):

```

Specify CREATE TABLE Statement

Type information, press Enter.

Table . . . . . INVENTORY_LIST_____ Name
Collection . . . . . SAMPLECOLL_____ Name, F4 for list

Nulls: 1=NULL, 2=NOT NULL, 3=NOT NULL WITH DEFAULT

Column          FOR Column  Type          Length  Scale  Nulls
ITEM_NUMBER_____  _____  CHAR          6       _     2
ITEM_NAME_____    _____  VARCHAR       20      _     3
UNIT_COST_____    _____  DECIMAL       8       2     3
QUANTITY_ON_HAND_____  _____  SMALLINT      _____  _     1
LAST_ORDER_DATE_____  _____  DATE          _____  _     1
ORDER_QUANTITY_____  _____  SMALLINT      _____  _     1
_____            _____  _____      _____  _     3
                                                    Bottom

Table CONSTRAINT . . . . . N      Y=Yes, N=No
Distributed Table . . . . . N      Y=Yes, N=No

F3=Exit  F4=Prompt  F5=Refresh  F6=Insert line  F10=Copy line
F11=Display more attributes  F12=Cancel  F14=Delete line  F24=More keys

```

- b. Type the table name INVENTORY_LIST and schema name SAMPLECOLL at the **Table** and **Collection** prompts, as shown.
- c. Each column you want to define for the table is represented by an entry in the list on the lower part of the display. For each column, type the name of the column, the data type of the column, its length and scale, and the null attribute.
- d. Press F11 to see more attributes that can be specified for the columns. This is where a default value can be specified.

```

Specify CREATE TABLE Statement

Type information, press Enter.

Table . . . . . INVENTORY_LIST_____ Name
Collection . . . . . SAMPLECOLL_____ Name, F4 for list

Data: 1=BIT, 2=SBCS, 3=MIXED, 4=CCSID

Column          Data  Allocate  CCSID  CONSTRAINT  Default
ITEM_NUMBER_____  -    _____  _____  N            _____
ITEM_NAME_____    -    _____  _____  N            '***UNKNOWN***'_____
UNIT_COST_____    -    _____  _____  N            _____
QUANTITY_ON_HAND_____  -    _____  _____  N            NULL_____
LAST_ORDER_DATE_____  -    _____  _____  N            _____
ORDER_QUANTITY_____  -    _____  _____  N            20_____
_____            -    _____  _____  -            _____
                                                    Bottom

Table CONSTRAINT . . . . . N      Y=Yes, N=No
Distributed Table . . . . . N      Y=Yes, N=No

F3=Exit  F4=Prompt  F5=Refresh  F6=Insert line  F10=Copy line
F11=Display more attributes  F12=Cancel  F14=Delete line  F24=More keys

```

Note: Another way of entering column definitions is to press F4 (Prompt) with your cursor on one of the column entries in the list. A display that shows all of the attributes for defining a single column appears.

- e. When all the values have been entered, press Enter to create the table. The Enter SQL Statements display is shown again with a message indicating that the table has been created.

Note: You can type this CREATE TABLE statement on the Enter SQL Statements display as follows:

```
CREATE TABLE SAMPLECOLL.INVENTORY_LIST
  (ITEM_NUMBER CHAR(6) NOT NULL,
   ITEM_NAME VARCHAR(20) NOT NULL WITH DEFAULT '***UNKNOWN***',
   UNIT_COST DECIMAL(8,2) NOT NULL WITH DEFAULT,
   QUANTITY_ON_HAND SMALLINT DEFAULT NULL,
   LAST_ORDER_DATE DATE,
   ORDER_QUANTITY SMALLINT DEFAULT 20)
```

2. Create a second table named SUPPLIERS. There are two methods you can use:
 - a. Type the following command directly on the Enter SQL Statements display.
 - b. Press F4 (Prompt) to use the interactive SQL displays to create the definition.

```
CREATE TABLE SAMPLECOLL.SUPPLIERS
  (SUPPLIER_NUMBER CHAR(4) NOT NULL,
   ITEM_NUMBER CHAR(6) NOT NULL,
   SUPPLIER_COST DECIMAL(8,2))
```

Use the LABEL ON statement

Normally, the column name is used as the column heading when showing the output of a SELECT statement in interactive SQL. By using the LABEL ON statement, you can create a more descriptive label for the column name. Because you run your examples in interactive SQL, you use the LABEL ON statement to change the column headings. Even though the column names were descriptive, it is easier to read if the column headings show each part of the name on a single line. It also allows you to see more columns of data on a single display.

To change the labels for your columns, follow these steps:

1. Enter LABEL ON COLUMN on the Enter SQL Statements display.
2. Press F4 (Prompt). The following display appears:

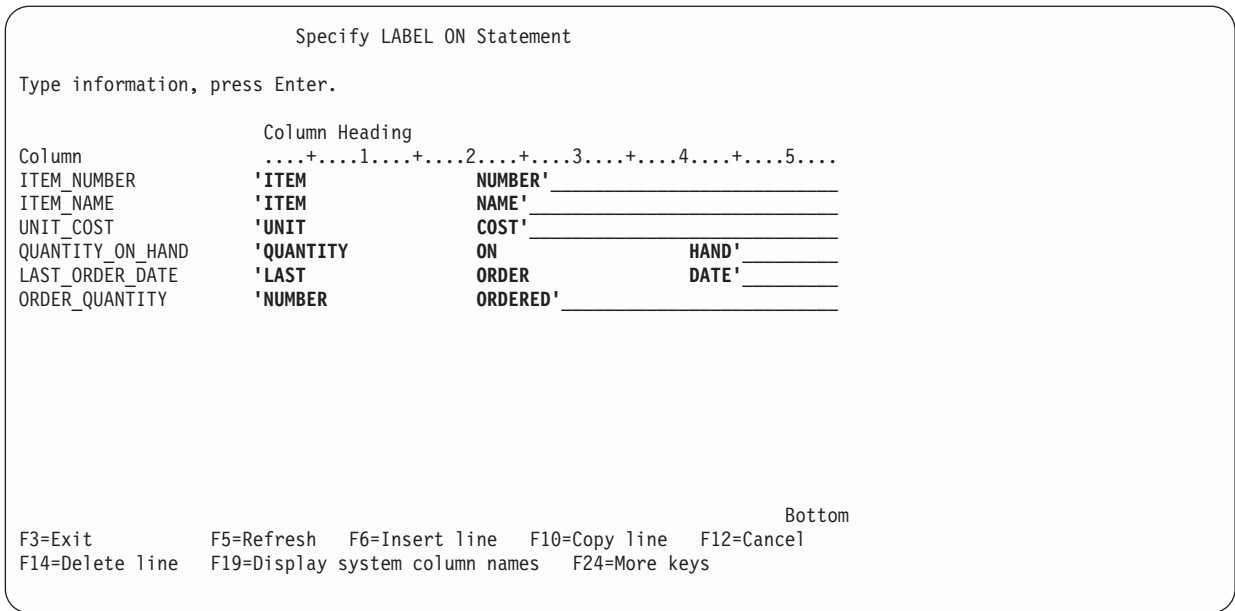
Specify LABEL ON Statement

Type choices, press Enter.

Label on 2		1=Table or view 2=Column 3=Package 4=Alias
Table or view INVENTORY_LIST_____	Name, F4 for list	
Collection . . SAMPLECOLL_	Name, F4 for list	
Option 1		1=Column heading 2=Text

F3=Exit F4=Prompt F5=Refresh F12=Cancel F20=Display full names
F21=Display statement

3. Type in the name of the table and schema containing the columns for which you want to add labels.
4. Press Enter. The following display is shown, prompting you for each of the columns in the table.



5. Type the column headings for each of the columns. Column headings are defined in 20 character sections. Each section is displayed on a different line when showing the output of a SELECT statement. The ruler across the top of the column heading entry area can be used to easily space the headings correctly.
6. Press Enter.

The following message indicates that the LABEL ON statement was successful:
 LABEL ON for INVEN00001 in SAMPLECOLL completed.

The table name in the message is the system table name for this table, not the name that was actually specified in the statement. DB2 Universal Database for iSeries maintains two names for tables with names longer than ten characters. For more information about system table names, see the CREATE TABLE statement in the SQL Reference topic.

Note: The LABEL ON statement can also be typed in directly on the Enter SQL statements display as follows:

```
LABEL ON SAMPLECOLL.INVENTORY_LIST
(ITEM_NUMBER      IS 'ITEM      NUMBER ',
 ITEM_NAME        IS 'ITEM      NAME ',
 UNIT_COST        IS 'UNIT      COST ',
 QUANTITY_ON_HAND IS 'QUANTITY  ON      HAND ',
 LAST_ORDER_DATE  IS 'LAST      ORDER    DATE ',
 ORDER_QUANTITY   IS 'NUMBER    ORDERED ')
```

Insert information into a table

After you create a table, you can insert, or add, information (data) into it by using the SQL INSERT statement.

1. On the Enter SQL Statements display, type INSERT and press F4 (Prompt). The Specify INSERT Statement display is shown.

```

Specify INSERT Statement

Type choices, press Enter.

INTO table . . . . . INVENTORY_LIST_____ Name, F4 for list
Collection . . . . . SAMPLECOLL__ Name, F4 for list

Select columns to insert
INTO . . . . . Y Y=Yes, N=No
Insertion method . . . . 1 1=Input VALUES
                                     2=Subselect

Type choices, press Enter.

WITH isolation level . . 1 1=Current level, 2=NC (NONE)
                                     3=UR (CHG), 4=CS, 5=RS (ALL)
                                     6=RR

F3=Exit F4=Prompt F5=Refresh F12=Cancel F20=Display full names
F21=Display statement

```

2. Type the table name and schema name in the input fields as shown.
3. Change the **Select columns to insert INTO** prompt to **Yes**.
4. Press Enter to see the display where the columns you want to insert values into can be selected.

```

Specify INSERT Statement

Type sequence numbers (1-999) to make selections, press Enter.

Seq Column          Type          Length Scale
1__ ITEM_NUMBER      CHARACTER     6
2__ ITEM_NAME       VARCHAR      20
3__ UNIT_COST       DECIMAL      8 2
4__ QUANTITY_ON_HAND SMALLINT      4
__  LAST_ORDER_DATE DATE
__  ORDER_QUANTITY SMALLINT      4

Bottom

F3=Exit F5=Refresh F12=Cancel F19=Display system column names
F20=Display entire name F21=Display statement

```

In this example, insert into four of the columns. Allow the other columns have their default value inserted. The sequence numbers on this display indicate the order that the columns and values are listed in the INSERT statement.

5. Press Enter to show the display where values for the selected columns can be typed.

Specify INSERT Statement

Type values to insert, press Enter.

Column	Value
ITEM_NUMBER	'153047'
ITEM_NAME	'Pencils, red'
UNIT_COST	10.00
QUANTITY_ON_HAND	25

Bottom

F3=Exit F5=Refresh F6=Insert line F10=Copy line F11=Display type
 F12=Cancel F14=Delete line F15=Split line F24=More keys

Note: To see the data type and length for each of the columns in the insert list, press F11 (Display type). This shows a different view of the insert values display, providing information about the column definition.

- Type the values to be inserted for all of the columns and press Enter. A row containing these values is added to the table. The values for the columns that were not specified have a default value inserted. For LAST_ORDER_DATE it is the null value because no default was provided and the column allows the null value. For ORDER_QUANTITY it is 20, the value specified as the default value on the CREATE TABLE statement.
- Type the INSERT statement on the Enter SQL Statements display as:


```
INSERT INTO SAMPLECOLL.INVENTORY_LIST
  (ITEM_NUMBER,
   ITEM_NAME,
   UNIT_COST,
   QUANTITY_ON_HAND)
VALUES ('153047 ',
       'Pencils,red ',
       10.00,
       25)
```
- To add the next row to the table, press F9 (Retrieve) on the Enter SQL Statements display. This copies the previous INSERT statement to the typing area. You can either type over the values from the previous INSERT statement or press F4 (Prompt) to use the Interactive SQL displays to enter data.
- Continue using the INSERT statement to add the following rows to the table.

Values not shown in the chart below should not be inserted so that the default is used. In the INSERT statement column list, specify only the column names for which you want to insert a value. For example, to insert the third row, specify only ITEM_NUMBER and UNIT_COST for the column names and only the two values for these columns in the VALUES list.

ITEM_NUMBER	ITEM_NAME	UNIT_COST	QUANTITY_ON_HAND
153047	Pencils, red	10.00	25
229740	Lined tablets	1.50	120
544931		5.00	
303476	Paper clips	2.00	100
559343	Envelopes, legal	3.00	500

ITEM_NUMBER	ITEM_NAME	UNIT_COST	QUANTITY_ON_HAND
291124	Envelopes, standard		
775298	Chairs, secretary	225.00	6
073956	Pens, black	20.00	25

Add the following rows to the SAMPLECOLL.SUPPLIERS table.

SUPPLIER_NUMBER	ITEM_NUMBER	SUPPLIER_COST
1234	153047	10.00
1234	229740	1.00
1234	303476	3.00
9988	153047	8.00
9988	559343	3.00
2424	153047	9.00
2424	303476	2.50
5546	775298	225.00
3366	303476	1.50
3366	073956	17.00

The sample schema now contains two tables with several rows of data in each.

Get information from a single table

Now that you have inserted all the information into our tables, you must look at it again. In SQL, this is done with the SELECT statement. The SELECT statement is the most complex of all SQL statements. This statement is composed of three main clauses:

1. The SELECT clause, which specifies those columns containing the data.
2. The FROM clause, which specifies the table or tables containing the columns with the data.
3. The WHERE clause, which supplies conditions that determine which rows of data are retrieved.

In addition to the three main clauses, there are several other clauses described in SQL Programming topic and in the SQL Reference topic that can affect the final form of returned data.

1. To see the values you inserted into the INVENTORY_LIST table, type SELECT and press F4 (prompt). The following display is shown:

Specify SELECT Statement

Type SELECT statement information. Press F4 for a list.

```

FROM tables . . . . . SAMPLECOLL.INVENTORY_LIST_____
SELECT columns . . . . . *_____
WHERE conditions . . . . . _____
GROUP BY columns . . . . . _____
HAVING conditions . . . . . _____
ORDER BY columns . . . . . _____
FOR UPDATE OF columns . . . . . _____

```

Bottom

Type choices, press Enter.

```

DISTINCT rows in result table . . . . . N Y=Yes, N=No
UNION with another SELECT . . . . . N Y=Yes, N=No
Specify additional options . . . . . N Y=Yes, N=No

```

F3=Exit F4=Prompt F5=Refresh F6=Insert line F9=Specify subquery
F10=Copy line F12=Cancel F14=Delete line F15=Split line F24=More keys

2. Type the table name in the **FROM tables** field on the display. To select all columns from the table, type * for the **SELECT columns** field on the display.

Display Data

Position to line Data width : 71
. Shift to column
.1.2.3.4.5.6.7.

ITEM NUMBER	ITEM NAME	UNIT COST	QUANTITY ON HAND	LAST ORDER DATE	NUMBER ORDERED
153047	Pencils, red	10.00	25	-	20
229740	Lined tablets	1.50	120	-	20
544931	***UNKNOWN***	5.00	-	-	20
303476	Paper clips	2.00	100	-	20
559343	Envelopes, legal	3.00	500	-	20
291124	Envelopes, standard	.00	-	-	20
775298	Chairs, secretary	225.00	6	-	20
073956	Pens, black	20.00	25	-	20
***** End of data *****					

F3=Exit F12=Cancel F19=Left F20=Right F21=Split

3. Press Enter and the statement runs to select all of the data for all of the columns in the table. The following output is shown:

The column headings that were defined using the LABEL ON statement are shown. The ITEM_NAME for the third entry has the default value that was specified in the CREATE TABLE statement. The QUANTITY_ON_HAND column has a null value for the rows where no value was inserted. The LAST_ORDER_DATE column contains all null values because that column is not in any of the INSERT statements and the column was not defined to have a default value. Similarly, the ORDER_QUANTITY column contains the default value for all rows.

This statement can be entered on the Enter SQL Statements display as:

```

SELECT *
FROM SAMPLECOLL.INVENTORY_LIST

```

4. To limit the number of columns returned by the SELECT statement, the columns you want to see must be specified. To restrict the number of output rows returned, the WHERE clause is used. To see only the items that cost more than 10 dollars, and only have the values for the columns ITEM_NUMBER, UNIT_COST, and ITEM_NAME returned, type SELECT and press F4 (Prompt). The Specify SELECT Statement display is shown.

```

Specify SELECT Statement

Type SELECT statement information. Press F4 for a list.

FROM tables . . . . . SAMPLECOLL.INVENTORY_LIST
SELECT columns . . . . . ITEM_NUMBER, UNIT_COST, ITEM_NAME
WHERE conditions . . . . . UNIT_COST > 10.00
GROUP BY columns . . . . .
HAVING conditions . . . . .
ORDER BY columns . . . . .
FOR UPDATE OF columns . . . . .

Bottom

Type choices, press Enter.

DISTINCT rows in result table . . . . . N Y=Yes, N=No
UNION with another SELECT . . . . . N Y=Yes, N=No
Specify additional options . . . . . N Y=Yes, N=No

F3=Exit F4=Prompt F5=Refresh F6=Insert line F9=Specify subquery
F10=Copy line F12=Cancel F14>Delete line F15=Split line F24=More keys

```

Although only one line is initially shown for each prompt on the Specify SELECT Statement display, F6 (Insert line) can be used to add more lines to any of the input areas in the top part of the display. This can be used if more columns were to be entered in the SELECT columns list, or a longer, more complex WHERE condition were needed.

- Fill in the display as shown above.
- Press Enter to run the SELECT statement. The following output appears:

```

Display Data
Data width . . . . . : 41
Position to line . . . . . Shift to column . . . . .
.....1.....2.....3.....4.
ITEM      UNIT      ITEM
NUMBER    COST      NAME
775298    225.00    Chairs, secretary
073956    20.00     Pens, black
***** End of data *****

F3=Exit F12=Cancel F19=Left F20=Right F21=Split

```

The only rows returned are those whose data values satisfy the condition specified in the WHERE clause. Furthermore, the only data values returned are from the columns you explicitly specified in the SELECT clause. Data values of columns other than those explicitly identified are not returned.

This statement can be entered on the Enter SQL Statements display as:

```

SELECT ITEM_NUMBER,UNIT_COST,ITEM_NAME
FROM SAMPLECOLL.INVENTORY_LIST
WHERE UNIT_COST > 10.00

```

Get information from multiple tables

SQL allows you to get information from columns contained in more than one table. This operation is called a join operation. In SQL, a join operation is specified by placing the names of those tables you want to join together into the same FROM clause of a SELECT statement.

Suppose you want to see a list of all the suppliers and the item numbers and item names for their supplied items. The item name is not in the SUPPLIERS table. It is in the INVENTORY_LIST table. Using the common column, ITEM_NUMBER, you can see all three of the columns as if they were from a single table.

Whenever the same column name exists in two or more tables being joined, the column name must be qualified by the table name to specify which column is really being referenced. In this SELECT statement, the column name ITEM_NUMBER is defined in both tables so the column name needs to be qualified by the table name. If the columns had different names, there is no confusion, so qualification is not needed.

1. To perform this join, the following SELECT statement can be used. Enter it by typing it directly on the Enter SQL Statements display or by prompting. If using prompting, both table names need to be typed on the FROM tables input line.

```
SELECT SUPPLIER_NUMBER, SAMPLECOLL.INVENTORY_LIST.ITEM_NUMBER, ITEM_NAME
FROM SAMPLECOLL.SUPPLIERS, SAMPLECOLL.INVENTORY_LIST
WHERE SAMPLECOLL.SUPPLIERS.ITEM_NUMBER
      = SAMPLECOLL.INVENTORY_LIST.ITEM_NUMBER
```

2. Another way to enter the same statement is to use a correlation name. A correlation name provides another name for a table name to use in a statement. A correlation name must be used when the table names are the same. It can be specified following each table name in the FROM list. The previous statement can be rewritten as:

```
SELECT SUPPLIER_NUMBER, Y.ITEM_NUMBER, ITEM_NAME
FROM SAMPLECOLL.SUPPLIERS X, SAMPLECOLL.INVENTORY_LIST Y
WHERE X.ITEM_NUMBER = Y.ITEM_NUMBER
```

In this example, SAMPLECOLL.SUPPLIERS is given a correlation name of X and SAMPLECOLL.INVENTORY_LIST is given a correlation name of Y. The names X and Y are then used to qualify the ITEM_NUMBER column name.

Running this example returns the following output:

```

                                Display Data
                                Data width . . . . . :    45
Position to line . . . . .      Shift to column . . . . .
....+....1....+....2....+....3....+....4....+
SUPPLIER_NUMBER  ITEM   ITEM
                NUMBER  NAME
      1234       153047  Pencils, red
      1234       229740  Lined tablets
      1234       303476  Paper clips
      9988       153047  Pencils, red
      9988       559343  Envelopes, legal
      2424       153047  Pencils, red
      2424       303476  Paper clips
      5546       775298  Chairs, secretary
      3366       303476  Paper clips
      3366       073956  Pens, black
***** End of data *****

F3=Exit      F12=Cancel   F19=Left    F20=Right   F21=Split
```

Note: Because no ORDER BY clause was specified for the query, the order of the rows returned by your query may be different.

The data values in the result table represent a composite of the data values contained in the two tables INVENTORY_LIST and SUPPLIERS. This result table contains the supplier number from the SUPPLIER table and the item number and item name from the INVENTORY_LIST table. Any item numbers that do not appear in the SUPPLIER table are not shown in this result table. The results are not guaranteed to be in any order unless the ORDER BY clause is specified for the SELECT statement. Because you did not change any column headings for the SUPPLIER table, the SUPPLIER_NUMBER column name is used as the column heading.

The following example shows how to use ORDER BY to guarantee the order of the rows. The statement first sorts the result table by the SUPPLIER_NUMBER column. Rows with the same value for SUPPLIER_NUMBER are sorted by their ITEM_NUMBER.

```

SELECT SUPPLIER_NUMBER, Y.ITEM_NUMBER, ITEM_NAME
FROM SAMPLECOLL.SUPPLIERS X, SAMPLECOLL.INVENTORY_LIST Y
WHERE X.ITEM_NUMBER = Y.ITEM_NUMBER
ORDER BY SUPPLIER_NUMBER, Y.ITEM_NUMBER

```

Running the previous statement produces the following output.

```

                                Display Data
                                Data width . . . . . :    45
Position to line . . . . .      Shift to column . . . . .
....+....1....+....2....+....3....+....4....+
SUPPLIER_NUMBER  ITEM      ITEM
                NUMBER  NAME
    1234         153047  Pencils, red
    1234         229740  Lined tablets
    1234         303476  Paper clips
    2424         153047  Pencils, red
    2424         303476  Paper clips
    3366         073956  Pens, black
    3366         303476  Paper clips
    5546         775298  Chairs, secretary
    9988         153047  Pencils, red
    9988         559343  Envelopes, legal
***** End of data *****
F3=Exit      F12=Cancel      F19=Left      F20=Right      F21=Split

```

Related information

SQL Reference

Change information in a table

You can use the SQL UPDATE statement to change the data values in some or all of the columns of a table.

If you want to limit the number of rows being changed during a single statement execution, use the WHERE clause with the UPDATE statement. If you do not specify the WHERE clause, all of the rows in the specified table are changed. However, if you use the WHERE clause, the system changes only the rows satisfying the conditions that you specify.

Suppose you want to use interactive SQL and are placing an order for more paper clips today.

1. To update the LAST_ORDER_DATE and ORDER_QUANTITY for item number 303476, type UPDATE and press F4 (Prompt). The Specify UPDATE Statement display is shown.

Specify UPDATE Statement

Type choices, press Enter.

Table	INVENTORY_LIST_____	Name, F4 for list
Collection	SAMPLECOLL__	Name, F4 for list
Correlation	_____	Name

F3=Exit F4=Prompt F5=Refresh F12=Cancel F20=Display full names
F21=Display statement

2. Enter the table name and schema name, as shown in the previous panel.
3. Press Enter. The display is shown again with the list of columns in the table.

Specify UPDATE Statement

Type choices, press Enter.

Table	INVENTORY_LIST_____	Name, F4 for list
Collection	SAMPLECOLL__	Name, F4 for list
Correlation	_____	Name

Type information, press Enter.

Column	Value
ITEM_NUMBER	_____
ITEM_NAME	_____
UNIT_COST	_____
QUANTITY_ON_HAND	_____
LAST_ORDER_DATE	CURRENT DATE _____
ORDER_QUANTITY	50 _____

Bottom

F3=Exit F4=Prompt F5=Refresh F6=Insert line F10=Copy line
F11=Display type F12=Cancel F14=Delete line F24=More keys

4. Specify CURRENT DATE in the LAST_ORDER_DATE field to change the value to today's date.
5. Enter the updated values as shown.
6. Press Enter to see the display on which the WHERE condition can be specified. If a WHERE condition is not specified, all the rows in the table are updated using the values from the previous display.

```

Specify UPDATE Statement

Type WHERE conditions, press Enter. Press F4 for a list.
ITEM_NUMBER = '303476'
_____
_____

Bottom

Type choices, press Enter.

WITH isolation level . . . 1          1=Current level, 2=NC (NONE)
                                     3=UR (CHG), 4=CS, 5=RS (ALL)
                                     6=RR

F3=Exit      F4=Prompt  F5=Refresh  F6=Insert line  F9=Specify subquery
F10=Copy line F12=Cancel  F14=Delete line F15=Split line  F24=More keys

```

7. Enter ITEM_NUMBER = '303476' in the WHERE condition field.
8. Press Enter to perform the update on the table. A message indicates that the function is complete.

Running a SELECT statement to get all the rows from the table (SELECT * FROM SAMPLECOLL.INVENTORY_LIST), returns the following result:

```

Display Data
Data width . . . . . : 71
Position to line . . . . . Shift to column . . . . .
.....1.....2.....3.....4.....5.....6.....7.
ITEM  ITEM          UNIT  QUANTITY  LAST  NUMBER
NUMBER NAME          COST  ON        ORDER  ORDERED
      HAND        DATE
153047 Pencils, red      10.00    25  -        20
229740 Lined tablets    1.50    120  -        20
544931 ***UNKNOWN***    5.00     -  -        20
303476 Paper clips      2.00    100  05/30/94  50
559343 Envelopes, legal  3.00    500  -        20
291124 Envelopes, standard .00     -  -        20
775298 Chairs, secretary 225.00    6  -        20
073956 Pens, black     20.00    25  -        20
***** End of data *****

Bottom

F3=Exit      F12=Cancel  F19=Left    F20=Right   F21=Split

```

Only the entry for *Paper clips* was changed. The LAST_ORDER_DATE was changed to be the current date. This date is always the date the update is run. The NUMBER_ORDERED shows its updated value.

This statement can be typed on the Enter SQL Statements display as:

```

UPDATE SAMPLECOLL.INVENTORY_LIST
SET LAST_ORDER_DATE = CURRENT DATE,
ORDER_QUANTITY = 50
WHERE ITEM_NUMBER = '303476'

```

Related information
SQL Programming

Delete information from a table

You can delete data from a table by using the SQL DELETE statement. You can delete entire rows from a table when they no longer contain needed information or you can use the WHERE clause with the DELETE statement to identify rows to be deleted during a single statement execution.

To remove all the rows in a table that have the null value for the QUANTITY_ON_HAND column, follow these steps:

1. Enter the following statement on the Enter SQL Statements display:

```
DELETE
FROM SAMPLECOLL.INVENTORY_LIST
WHERE QUANTITY_ON_HAND IS NULL
```

To check a column for the null value, the IS NULL comparison is used.

This disclaimer information pertains to code examples.

2. After the delete operation is completed, run another SELECT statement. This results in the following table:

Display Data						
			Data width	71		
Position to line		Shift to column				
.....1.....2.....3.....4.....5.....6.....7.						
ITEM NUMBER	ITEM NAME	UNIT COST	QUANTITY ON HAND	LAST ORDER DATE	NUMBER ORDERED	
153047	Pencils, red	10.00	25	-	20	
229740	Lined tablets	1.50	120	-	20	
303476	Paper clips	2.00	100	05/30/94	50	
559343	Envelopes, legal	3.00	500	-	20	
775298	Chairs, secretary	225.00	6	-	20	
073956	Pens, black	20.00	25	-	20	
***** End of data *****						
						Bottom
F3=Exit	F12=Cancel	F19=Left	F20=Right	F21=Split		

The rows with a null value for QUANTITY_ON_HAND were deleted.

Create and use a view

You might find that no single table contains all the information you need. You might also want to give users access to only part of the data in a table. Views provide a way to subset the table so that you deal with only the data you need. A view reduces complexity and, at the same time, restricts access.

You can create a view using the SQL CREATE VIEW statement. Using the CREATE VIEW statement, defining a view on a table is like creating a new table containing just the columns and rows you want. When your application uses a view, it cannot access rows or columns of the table that are not included in the view. However, rows that do not match the selection criteria can still be inserted through a view if the SQL WITH CHECK OPTION is not used. See WITH CHECK OPTION on a View in the SQL Programming topic for more information about using WITH CHECK OPTION.

In order to create a view you must have the proper authority to the tables or physical files on which the view is based. See the CREATE VIEW statement in the SQL Reference topic for a list of authorities needed.

If you did not specify column names in the view definition, the column names are the same as those for the table on which the view is based.

You can make changes to a table through a view even if the view has a different number of columns or rows than the table. For INSERT, columns in the table that are not in the view must have a default value.

You can use the view as though it were a table, even though the view is totally dependent on one or more tables for data. The view has no data of its own and therefore requires no storage for the data. Because a view is derived from a table that exists in storage, when you update the view data, you are really updating data in the table. Therefore, views are automatically kept up-to-date as the tables they depend on are updated.

Create a view on a single table

The following example procedure shows how to create a view on a single table. The view is built on the INVENTORY_LIST table. The table has six columns, but the view uses only three of the columns: ITEM_NUMBER, LAST_ORDER_DATE, and QUANTITY_ON_HAND. The order of the columns in the SELECT clause is the order in which they appear in the view. The view contains only the rows for items that were ordered in the last two weeks. The CREATE VIEW statement looks like this:

1. Use the following command to create the view:

```
CREATE VIEW SAMPLECOLL.RECENT_ORDERS AS
SELECT ITEM_NUMBER, LAST_ORDER_DATE, QUANTITY_ON_HAND
FROM SAMPLECOLL.INVENTORY_LIST
WHERE LAST_ORDER_DATE > CURRENT DATE - 14 DAYS
```

In the preceding example, the columns in the view have the same name as the columns in the table because no column list follows the view name. The schema that the view is created into does not need to be the same schema as the table it is built over. Any schema or library can be used.

2. Run this statement:

```
SELECT *FROM SAMPLECOLL.RECENT_ORDERS
```

The result looks like this:

```
Display Data
Data width . . . . . : 26
Shift to column . . . . .
Position to line . . . . .
....+....1....+....2....+
ITEM   LAST   QUANTITY
NUMBER ORDER  ON
      DATE  HAND
303476 05/30/94 100
***** End of data *****
Bottom
F3=Exit   F12=Cancel   F19=Left   F20=Right   F21=Split
```

The only row selected by the view is the row that you updated to have the current date. All other dates in our table still have the null value so they are not returned.

Create a view combining data from multiple tables

You can create a view that combines data from two or more tables by naming more than one table in the FROM clause. In the following example procedure, the INVENTORY_LIST table contains a column of item numbers called ITEM_NUMBER, and a column with the cost of the item, UNIT_COST. These are joined with the ITEM_NUMBER column and the SUPPLIER_COST column of the SUPPLIERS table. A WHERE clause is used to limit the number of rows returned. The view will only contain those item numbers for suppliers that can supply an item at lower cost than the current unit cost.

1. Use the following command to create the view:

```
CREATE VIEW SAMPLECOLL.LOWER_COST AS
SELECT SUPPLIER_NUMBER, A.ITEM_NUMBER, UNIT_COST, SUPPLIER_COST
FROM SAMPLECOLL.INVENTORY_LIST A, SAMPLECOLL.SUPPLIERS B
WHERE A.ITEM_NUMBER = B.ITEM_NUMBER
AND UNIT_COST > SUPPLIER_COST
```


2. Run this statement:

```
SELECT *FROM SAMPLECOLL.LOWER_COST
```

The results look like this:

```

                                Display Data
                                Data width . . . . . :    51
Position to line . . . . .      Shift to column . . . . .
....+....1....+....2....+....3....+....4....+....5.
SUPPLIER_NUMBER  ITEM          UNIT  SUPPLIER_COST
                NUMBER        COST
                1234          229740  1.50          1.00
                9988          153047  10.00         8.00
                2424          153047  10.00         9.00
                3366          303476  2.00          1.50
                3366          073956  20.00         17.00
***** End of data *****
                                                                Bottom
F3=Exit      F12=Cancel    F19=Left    F20=Right   F21=Split

```

Note: Because no ORDER BY clause was specified for the query, the order of the rows returned by your query may be different.

The rows that can be seen through this view are only those rows that have a supplier cost that is less than the unit cost.

Related tasks

“Query your database using the Run SQL Scripts interface” on page 16

The Run SQL Scripts window in iSeries Navigator allows you to create, edit, run, and troubleshoot scripts of SQL statements. When you have finished working with the scripts, you can save them to your PC.

SQL versus traditional file access terminology

DB2 Universal Database for iSeries provides two access methods for manipulating database tables and data: Structured Query Language (SQL) and system file access methods. They use different words to describe some similar concepts.

As an interface to DB2 Universal Database for iSeries, the DB2 UDB Query Manager and SQL Development Kit adds an interactive query and report writing interface, as well as precompilers and tools to assist in writing SQL application programs in high-level programming languages. Conforming to the industry standard Structured Query Language (SQL), the SQL implementation for i5/OS allows you to define, manipulate, query, and control access to your iSeries data. It works equally well with i5/OS files and SQL tables.

SQL versus traditional file access terminology

SQL term	Traditional file access term
Collection. A group of related objects that consists of a library, a journal, a journal receiver, an SQL catalog, and an optional data dictionary. A collection enables the user to find the objects by name.	Library. A group of related objects that enables the user to find the objects by name.
Table. A set of columns and rows.	Physical file. A set of records.
Row. The horizontal part of a table containing a serial set of columns.	Record. A set of fields.
Column. The vertical part of a table of on data type.	Field. One of more bytes of related information of one data type.

SQL term	Traditional file access term
View. A subset of columns and rows of one or more tables.	Logical file. A subset of fields or records of up to 32 physical files.
Index. A collection of data in the columns of a table, logically arranged in ascending or descending order.	Index. A type of logical file.
Package. An object that contains control structures for SQL statements to be used by an application server.	SQL package. An object that contains control structures for SQL statements to be used by an application server.
Catalog. A set of tables and views that contain information about tables, packages, views, indexes, and constraints.	No similar object. However, the Display File Description (DSPFD) and Display File Field Description (DSPFFD) commands provide some of the same information that querying an SQL catalog provides.

Code license and disclaimer information

IBM grants you a nonexclusive copyright license to use all programming code examples from which you can generate similar function tailored to your own specific needs.

SUBJECT TO ANY STATUTORY WARRANTIES WHICH CANNOT BE EXCLUDED, IBM, ITS PROGRAM DEVELOPERS AND SUPPLIERS MAKE NO WARRANTIES OR CONDITIONS EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT, REGARDING THE PROGRAM OR TECHNICAL SUPPORT, IF ANY.

UNDER NO CIRCUMSTANCES IS IBM, ITS PROGRAM DEVELOPERS OR SUPPLIERS LIABLE FOR ANY OF THE FOLLOWING, EVEN IF INFORMED OF THEIR POSSIBILITY:

1. LOSS OF, OR DAMAGE TO, DATA;
2. DIRECT, SPECIAL, INCIDENTAL, OR INDIRECT DAMAGES, OR FOR ANY ECONOMIC CONSEQUENTIAL DAMAGES; OR
3. LOST PROFITS, BUSINESS, REVENUE, GOODWILL, OR ANTICIPATED SAVINGS.

SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF DIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, SO SOME OR ALL OF THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

Appendix. 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:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, 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 Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation

Software Interoperability Coordinator, Department YBWA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

- | The licensed program described in this information and all licensed material available for it are provided
- | by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement,
- | IBM License Agreement for Machine Code, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming Interface Information

This Database overview publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of IBM i5/OS.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

- | DB2
- | DB2 Universal Database
- | i5/OS
- | IBM
- | IBM (logo)
- | iSeries

Other company, product, and service names may be trademarks or service marks of others.

Terms and conditions

Permissions for the use of these publications is granted subject to the following terms and conditions.

Personal Use: You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of these publications, or any portion thereof, without the express consent of IBM.

Commercial Use: You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.



Printed in USA