Contents

The Purpose of DOMAIN

Working With DOMAIN

Object Editing

Ordering the Manual

Manual Order Number: ESD611-042WOU

This online help is applicable to ADABAS D Version 6.1.1 PE and to all subsequent releases, unless otherwise indicated in new editions or technical newsletters.

Specifications contained herein are subject to change and these changes will be reported in subsequent revisions or editions.

Readers' comments are welcomed. Comments may be addressed to the Documentation Department at the address on the back cover.

Printed in the Federal Republic of Germany

The SOFTWARE AG documentation often refers to numerous hardware and software products by their trade names. In most, if not all cases, these designations are claimed as trademarks or registered trademarks by their respective companies.

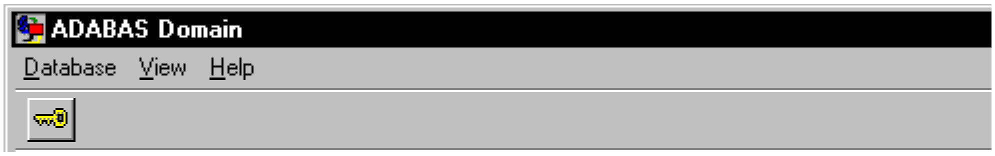
The Purpose of DOMAIN

DOMAIN is the DBA (database administration) tool for ADABAS that combines the complete data definition functionality of ADABAS with menu-driven data dictionary functions. DOMAIN is a multiple-document interface (MDI) Windows 3.1 application with drag-and-drop functions.

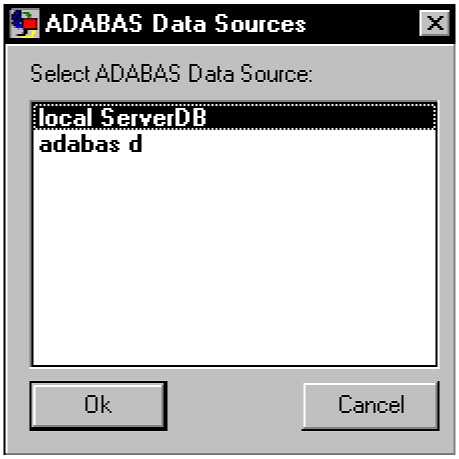
The data definition functionality corresponds to the CREATE/ALTER/DROP operations applied to the **Database Objects** (***Table***, ***Column***, ***Domain***, ***User***, etc.). This data dictionary functionality provides information on objects and their relationships to other objects (How is the 'Customer' table defined? What are the programs in which the 'Customer' table is accessed?).

Logging onto the Database

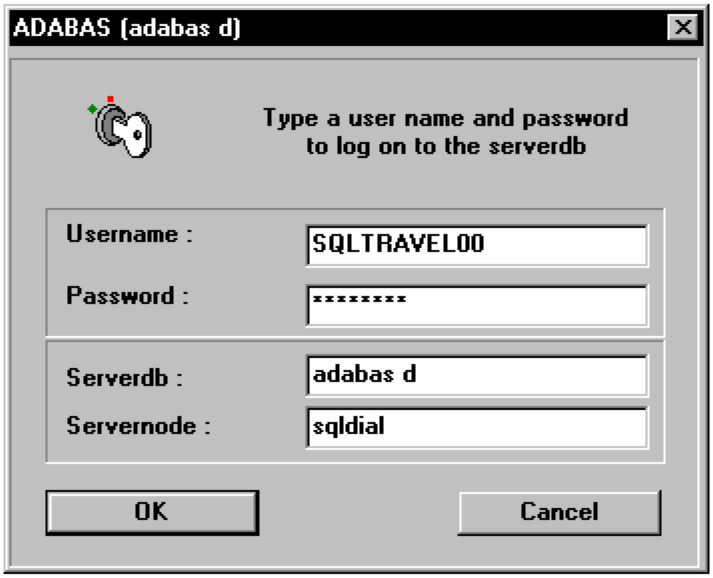
DOMAIN opens with an empty screen and the following menu bar:



You can log onto the database by clicking on the **Database/Open Database** menu item. A list box containing a list of all ADABAS databases installed with the aid of the ODBC administrator is then displayed.



After selecting the desired database, a logon screen is displayed. Enter there username and password.



Type a user name and password to log on to the serverdb

Username : SQLTRAVEL00

Password : ********

Serverdb : adabas d

Servernode : sqldial

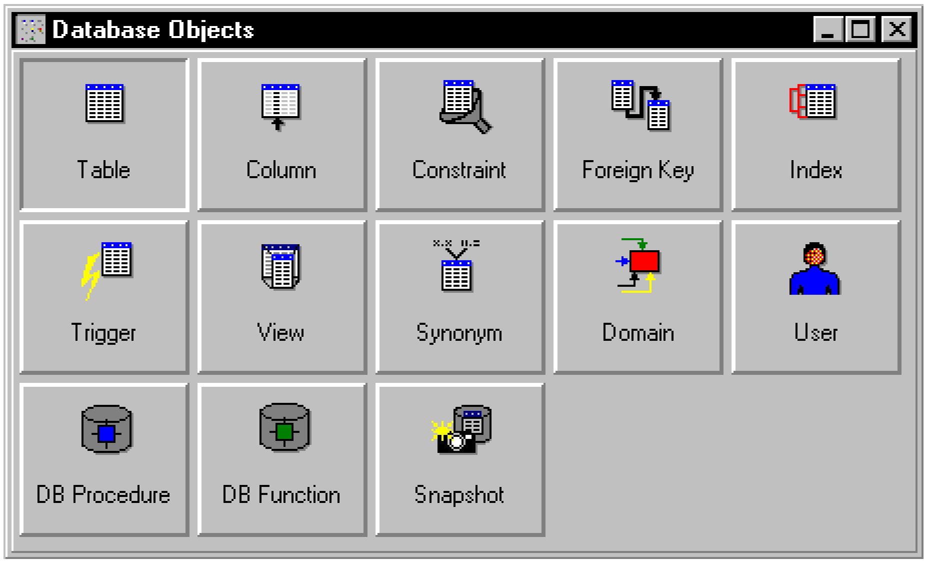
If DOMAIN has been used previously, it has noted the logon parameters that were used in the drop-down menu for the **Database** menu function. Click on this menu item if you wish to reuse these settings to log on directly. In this case, a dialog box is displayed only if a password is required for this user. As soon as the connection to the database is established, you can choose whether the password of this user should be saved or not by using the **View/Options...Connect** menu item.

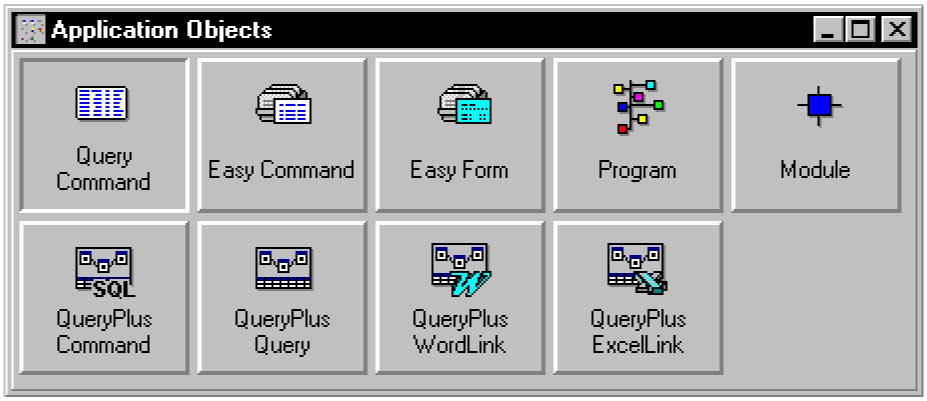
Unless the username and password are enclosed in double quotation marks, they are converted to uppercase letters when entered in the logon screen. Double quotation marks are also required when the username contains special characters. If the username itself contains a double quote, the double quote must be entered twice.

If you click the 🔑 button you can to connect to the database with the parameters of the last session - this corresponds to the first parameter set of the **Database** menu item.

Object Menu

Once you have logged on successfully, the DOMAIN main menu appears. This menu displays all objects that are managed by DOMAIN. In order to show their relationships to one another and for the sake of clarity, these objects are divided into two groups: **Database Objects** and **Application Objects**.





Select the group to be displayed under the **View** menu item.

Database Objects are all objects managed by ADABAS using CREATE/ ALTER/DROP statements. **Application Objects** are those objects managed by applications or other ADABAS tools (e.g., *EASY*, *QUERY* and *QueryPlus*). Although you cannot create or alter these objects using DOMAIN, you can determine their relationships to other objects; e.g., you can find out which tables are used in which programs.

The ***Domain*****,** ***User*** objects occur in relationships with many other objects and, thus, form a separate group within the database objects. ***Domains*** describe data elements that can be used in tables or forms.

The ***User*** object identifies all users defined in the database along with their access privileges for such objects as ***Table*, *Column*, *Index*, *Module*, *DB Procedure*,** etc**.**

The ***DB Procedure, Trigger,*** and ***DB Function*** objects allow the user to relocate application logic to the database server.

The ***DB Procedure*** object describes the DB procedures that are defined in the database and executed in the server. These procedures form a software layer that can be used by any application.

The ***Trigger*** object describes a procedure that is called implicitly after changing the table or column to which the trigger is related (INSERT, UPDATE, DELETE). Triggers are used to check complicated integrity rules, to initiate derived database modifications for the corresponding row or any other row, or to implement complicated rules for access protection.

The ***DB Function*** object describes user-specific functions that can be used in SELECT statements. ***DB Functions*** also allow the user to relocate application logic to the database server.

Using DOMAIN, the developer of a ***DB Procedure,*** a ***Trigger,*** or a ***DB Function*** can create a specification of the object as a comment that can then be called by all authorized users. Users of the ***DB Procedure, Trigger,*** or ***DB Function*** can also easily view parameter definitions and descriptions. The ***DB Procedure, Trigger,*** or ***DB Function*** objects themselves are defined with the aid of ADABASIC.
The ***Snapshot*** object is a special *read-only* table, the contents of which are a replicated partial dataset of one or more base tables. The ***Snapshot*** contents are explicitly updated with the REFRESH statement (see the ADABAS Reference online help).

The ***Module*** object describes a compilation unit in C or COBOL, or in ADABASIC. A module in ADABASIC exists for each ***DB Procedure***, ***Trigger***, and ***DB Function***.

Before executing a function, single-click on the object to which the function is to be applied. All functions that are possible for the selected object are made available in the menu bar. The most important functions are also offered in a toolbar and can be called by clicking on them directly.

DOMAIN presents the objects as a list or as an individual object (design mode). Some of the more complex objects (e.g., ***Table***) consist of several other objects in design mode. For less complex objects, the complete description of the object is contained in the object list. Any number of object lists and windows for individual objects of different object types can be displayed simultaneously on the screen.

When you select a specific object from the object list, or if you are in design mode, all operations in the menu bar refer to this object. In particular, you are provided with a list of all objects to which the selected object is related.

From the main menu, you can access

- The list of all objects of the selected object type that you are authorized to view by selecting the **Object/List All Objects** menu item or by clicking on the toolbar button.

- The list of all objects of the selected object type of which you are the owner by selecting the **Object/List Own Objects** menu item, by clicking on the toolbar button, or by double-clicking.

- A dialog box for entering the search arguments in order to define the relevant object list more precisely by selecting the **Object/Select Objects** menu item or by clicking on the toolbar button. The object list is then displayed.

- The empty design window for the selected object by selecting the **Object/New...** menu item or by clicking on the toolbar button.

- The initial DOMAIN window from which you can again log onto the database (e.g., under a different username) by selecting the **Object/Close Database** menu item.

The **menu bar** offers only those functions that are possible for the selected object type; other functions are displayed as deactivated menu items.

Some menu functions are always available and activated:

Object/Close:	Always exits the current window; changes can still be saved.
Object/Print:	Prints out the current object list or relationship display.
Object/SQL:	Opens the SQL window for ad-hoc queries.
Object/Exit:	Exits DOMAIN; changes can still be saved.
Window/Close All:	closes all windows except the main menu
Window/Cascade ..Arrange Icons:	see Windows
Help:	displays help information

Since DOMAIN is a multiple-document interface (MDI) Windows 3.1 application, several lists and several definition windows for different object types can be simultaneously displayed on the screen. While defining an object, you can display different lists for this or for other objects.

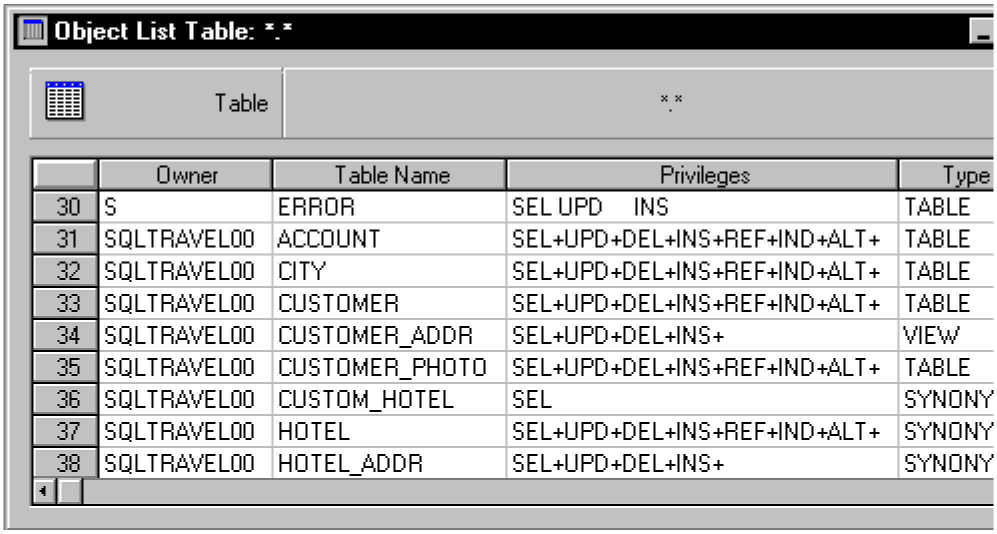
Object Lists

The object list contains an object occurrence (e.g., a table) on each line. The columns of the object list are the object attributes. Most objects are described in full in the object list. However, some of the database objects are made complex by the fact that they comprise a number of objects (a table comprises several columns); for this reason, their complete structure is displayed in design mode rather than in the object list.

Within an object list, you can scroll linewise or pagewise using the scroll arrows of the scroll bar on the right, or you can jump directly to the first or last row of the list.

Listing All Objects of One Type

From the object type menu, select the **Object/List All Objects** menu function or click on the toolbar button to display a list of all objects of the selected object type. This list contains all the objects that you are authorized to view. The search arguments used for this list are displayed as '*.*' in the bar above the displayed list.



Object List Table: *.*

Table *.*

	Owner	Table Name	Privileges	Type
30	S	ERROR	SEL UPD INS	TABLE
31	SQLTRAVEL00	ACCOUNT	SEL+UPD+DEL+INS+REF+IND+ALT+	TABLE
32	SQLTRAVEL00	CITY	SEL+UPD+DEL+INS+REF+IND+ALT+	TABLE
33	SQLTRAVEL00	CUSTOMER	SEL+UPD+DEL+INS+REF+IND+ALT+	TABLE
34	SQLTRAVEL00	CUSTOMER_ADDR	SEL+UPD+DEL+INS+	VIEW
35	SQLTRAVEL00	CUSTOMER_PHOTO	SEL+UPD+DEL+INS+REF+IND+ALT+	TABLE
36	SQLTRAVEL00	CUSTOM_HOTEL	SEL	SYNONY
37	SQLTRAVEL00	HOTEL	SEL+UPD+DEL+INS+REF+IND+ALT+	SYNONY
38	SQLTRAVEL00	HOTEL_ADDR	SEL+UPD+DEL+INS+	SYNONY

Listing All Objects of One Type of Which You Are the Owner

From the object type menu, select the **Object/List Own Objects** menu function or click on the toolbar button to display a list of all objects of which you are the owner. The search arguments displayed in the bar above the list contain the owner names of the objects (<username>.*).

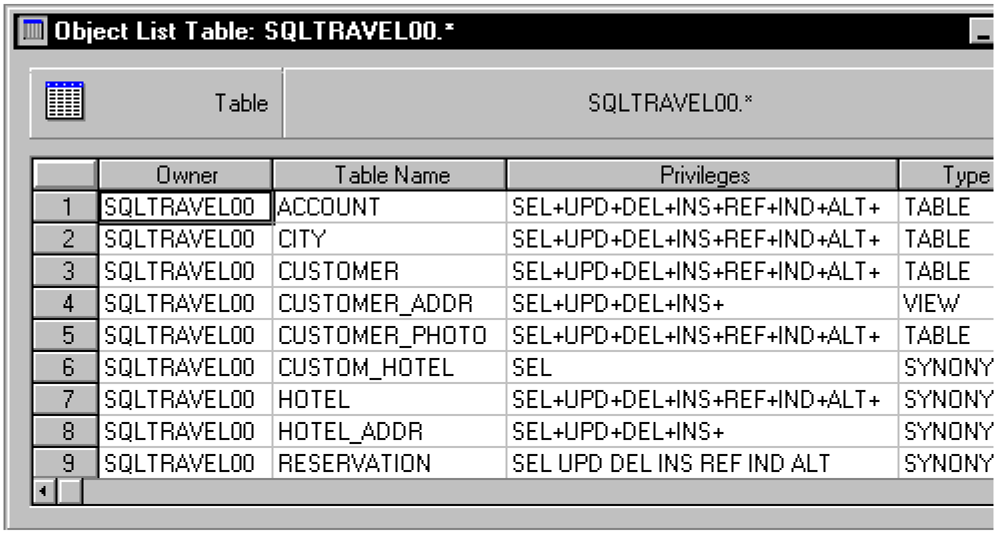


Table SQLTRAVEL00.*

	Owner	Table Name	Privileges	Type
1	SQLTRAVEL00	ACCOUNT	SEL+UPD+DEL+INS+REF+IND+ALT+	TABLE
2	SQLTRAVEL00	CITY	SEL+UPD+DEL+INS+REF+IND+ALT+	TABLE
3	SQLTRAVEL00	CUSTOMER	SEL+UPD+DEL+INS+REF+IND+ALT+	TABLE
4	SQLTRAVEL00	CUSTOMER_ADDR	SEL+UPD+DEL+INS+	VIEW
5	SQLTRAVEL00	CUSTOMER_PHOTO	SEL+UPD+DEL+INS+REF+IND+ALT+	TABLE
6	SQLTRAVEL00	CUSTOM_HOTEL	SEL	SYNONY
7	SQLTRAVEL00	HOTEL	SEL+UPD+DEL+INS+REF+IND+ALT+	SYNONY
8	SQLTRAVEL00	HOTEL_ADDR	SEL+UPD+DEL+INS+	SYNONY
9	SQLTRAVEL00	RESERVATION	SEL UPD DEL INS REF IND ALT	SYNONY

Selecting an Object List

Select the **Object/Select Objects** menu function or click on the toolbar button to display a dialog box in which you can enter search arguments for selecting the object list. The dialog box varies according to the object; for example, you can enter the owner and table name as a search pattern for a ***Table*** object, and you can restrict the list to the base tables or include view, synonym, result, or snapshot tables. If you enter '*' in all the text boxes, DOMAIN displays all objects for which you have access privileges.

All expressions known for the LIKE predicate of the SELECT statement (see the ADABAS Reference online help) can be used as a search pattern. For example, for the search pattern '???(AB)*', all names are searched that begin with three arbitrary characters that are followed by an uppercase 'A' or 'B' and any number of arbitrary characters.

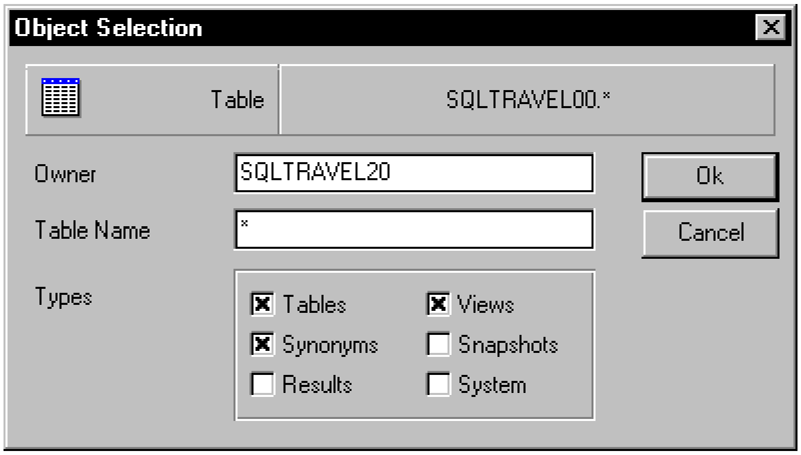


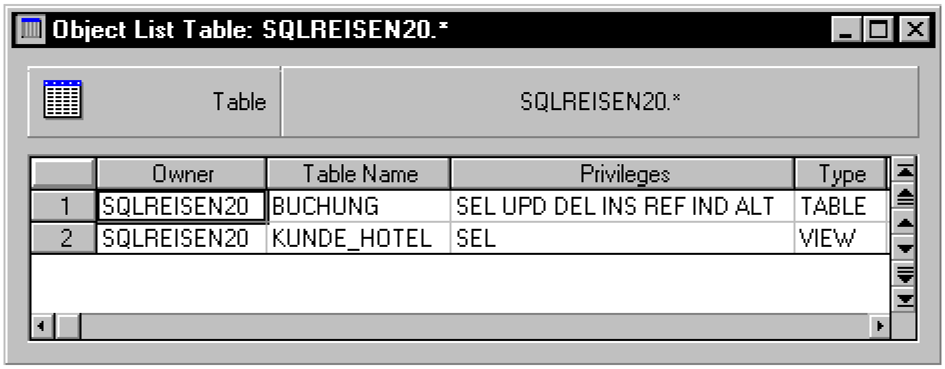
Table SQLTRAVEL00.*

Owner: SQLTRAVEL20

Table Name: *

Types:
☒ Tables ☒ Views
☒ Synonyms □ Snapshots
□ Results □ System

In this example, all tables are displayed that belong to users whose names begin with 'SQLTRAVEL2' and the current user is authorized to access. Note that the search arguments to be entered in the dialog box are case sensitive.



SQLREISEN20.*

	Owner	Table Name	Privileges	Type
1	SQLREISEN20	BUCHUNG	SEL UPD DEL INS REF IND ALT	TABLE
2	SQLREISEN20	KUNDE_HOTEL	SEL	VIEW

Customizing the Object Lists

As is typical when working with Windows, you can customize the tabular-format object lists. Until they are changed, any customizations you select apply to all object lists of *one* object type. To customize the object lists, you must already be familiar with Windows interfaces; for this reason, the description below uses specialized Windows terminology:

- Select the Windows **Edit/Copy** menu item in order to pass the rows of the tabular presentation as usual to other Windows tools.

- Select the **Edit/Comment** menu item to open the window in which the comment for the object is displayed. You can create and modify the comment if you are the owner of the object.

- Select the **Format** menu item to delete columns from and add columns to the tabular presentation (**Show Columns**), and to define fixed columns (**Freeze Columns**). Use **Size to Fit** to output columns at their minimum width. Use **Gridlines** to enable and disable grid lines.

- Under **Records**, use the **Goto** menu item to page through and the **Refresh** menu item or the toolbar button to refresh the display. This also updates the list to reflect changes to the object. Select **Quick Sort** to sort the table column by column using up to three columns as sort criteria. When doing so, only the visible rows are selected, not the total set of result rows.

To exit an object list, close the window or call the **Object/Close** function.

From the object list, you can access:

- Object editing or design mode, in which you can display or modify the object selected using the **Object/Design** menu item or clicking on the toolbar button. For application objects, there is no design mode.

- Object editing to define a new object of the selected object type using the **Object/New** menu item or by clicking on the toolbar button.

- The list of objects that **use** the selected object by selecting the **Relships/Uses Relships** menu item. All relationships are provided for selection in which the object occurs as **using** object.

- The list of objects **used by** the selected object by selecting the **Relships/Used by Relships** menu item. All relationships are provided for selection in which the object occurs as **used** object.

- The relationship display to select the desired relationship from all relationships involving the object by selecting the **Relships/All Object Relships** menu item.

- A list of statistical information about the chosen object by selecting the **Object/Statistic** menu item or by clicking on the 123 button when you are in a ***Table, Index,*** or ***User*** list.

Design Mode

Select the **Object/Design** function or click on the toolbar button to change to design mode, in which you can display the entire object as well as modify it.

Creating a New Object

To create a new object, select the **Object/New...** menu item. This menu item either branches directly to an empty design window for the object or to a submenu. Thus, for example, for the ***Table*** object, you can choose among **New/Table**, **New/Table Like**, and **New/Table as Query**. For a detailed description of how to define a particular object, please refer to the section named after the object.

You can also click on the toolbar button. Normally, this corresponds to the **Object/New** function. If the **Object/New** function comprises a number of submenu items, the toolbar button corresponds to the first subfunction (e.g., **Object/New/Table** for the ***Table*** object).

The **Object/Copy** function provides you with another way to create a new object. **Object/Copy** copies an existing object to an empty design window. Here you can make any changes you wish before finally creating the object under the desired name.

After you call the **Object/Save** menu function, an object is not actually created until you have entered the name of the object and have successfully executed the action. In the event of an error, DOMAIN issues a message and does not exit design mode.

If you use the **Object/Close** function to exit design mode after modifying the object, DOMAIN asks you whether you would like to save the changes.

Relationship Lists

First, select a specific object, either by clicking on the relevant line in the object list or by changing to the object's design mode. You can then display all the objects that are in a relationship with the selected object. Select the **Relships** menu item to change to the relationship display.

See also

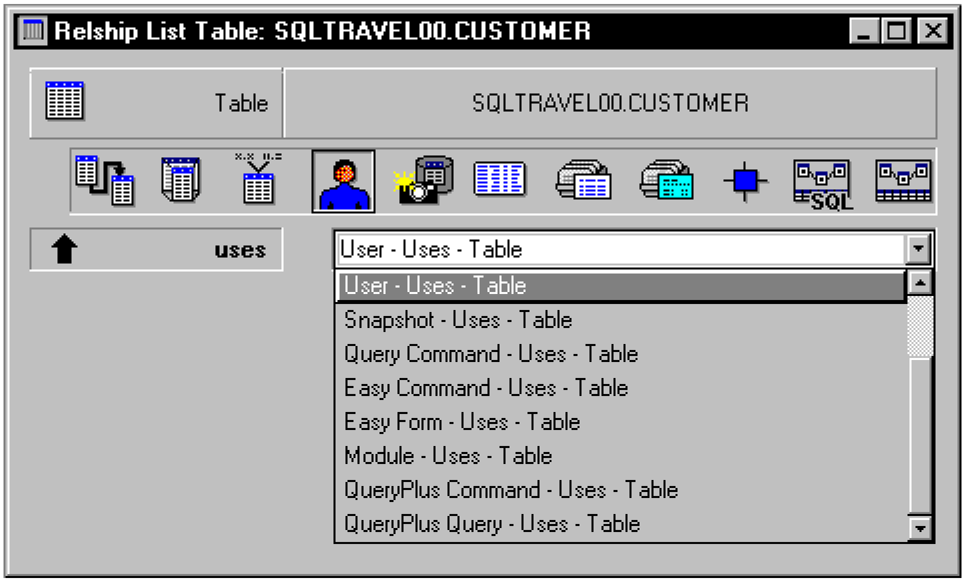
Detailed Relationship Lists

Relationship Lists in the Form of Object Lists

Hierarchical Relationship Lists

Detailed Relationship Lists

Example of the Relationship Display **Relships/Used by Relships** for a Table:



Relship List Table: SQLTRAVEL00.CUSTOMER

Table SQLTRAVEL00.CUSTOMER

uses

User - Uses - Table

User - Uses - Table
Snapshot - Uses - Table
Query Command - Uses - Table
Easy Command - Uses - Table
Easy Form - Uses - Table
Module - Uses - Table
QueryPlus Command - Uses - Table
QueryPlus Query - Uses - Table

The title bar of the detailed relationship display shows the composite name of the selected object. Below this bar, the objects to which the selected object type is related are represented by their icons. In addition, the relationships are displayed as a list of relationship names below the object icons in the combo box.

Select the **Relships/Uses Relships** menu item to display all relationships in which the selected object type occurs as the **using** object. Select the **Relships/Used by Relships** menu item to display all relationships in which the selected object type occurs as the **used** object. Select the **Relships/All Object Relships** menu item to display all relationships involving the selected object type either in a combo box or as an object icon.

In the last case, an object icon can be displayed twice, because it can be related to the selected object as both a using and used object (see Synonym **Uses Synonym** and Synonym **Refers toView**).

Two ways to display the list of objects that have relationships to each other:

1. Double Click on the object icon to display the list of all own objects that have a relationship to the selected object. The name of the selected relationship is displayed in the combo box.

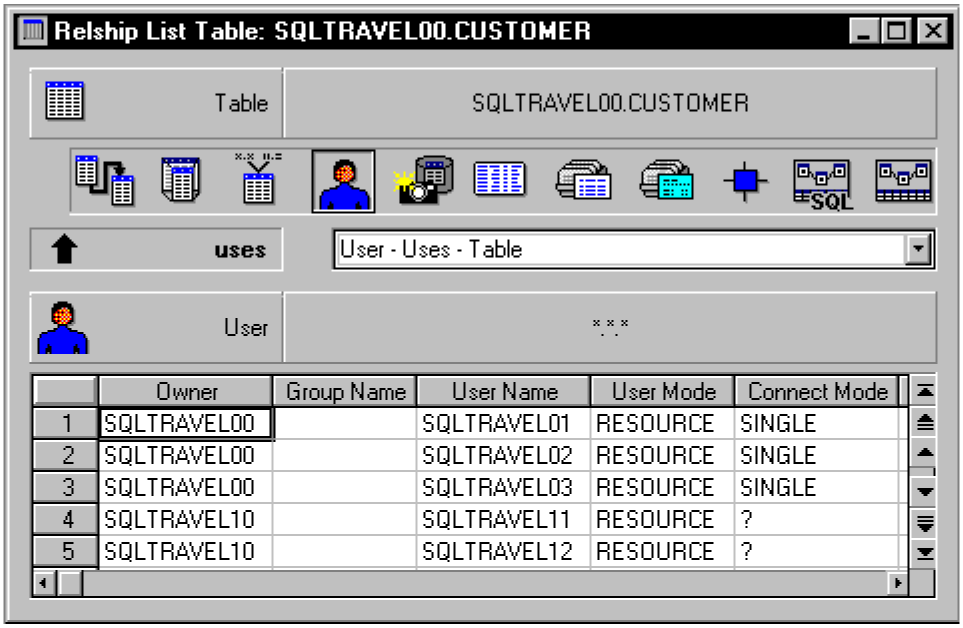
2. First, you can select the relationship either using the combo box or by clicking on the object icon. Then you can select the objects that have a relationship to the current object. Select the **Object/List All Objects** function (toolbar button) to obtain a list of all objects and the **Object/List Own Objects** function

(

toolbar button) to obtain a list of your own objects that have the selected relationship to the current object. When you select the **Object/Select Objects** function (

toolbar button), a dialog box is displayed in which you can further qualify the list to be generated.

The following example shows the result of executing **Object/List All Objects**:



Relship List Table: SQLTRAVEL00.CUSTOMER

Table: SQLTRAVEL00.CUSTOMER

uses: User - Uses - Table

User *.*.*

	Owner	Group Name	User Name	User Mode	Connect Mode
1	SQLTRAVEL00		SQLTRAVEL01	RESOURCE	SINGLE
2	SQLTRAVEL00		SQLTRAVEL02	RESOURCE	SINGLE
3	SQLTRAVEL00		SQLTRAVEL03	RESOURCE	SINGLE
4	SQLTRAVEL10		SQLTRAVEL11	RESOURCE	?
5	SQLTRAVEL10		SQLTRAVEL12	RESOURCE	?

The list of objects with the selected relationship to the object that was initially selected is displayed in the bottom half of the relationship list form. The **customizing** options for object lists described above also apply to this object list.

Navigating From One Object to Another

All operations can be executed for the object list within the relationship display that can be executed for the simple object list. After selecting an object, you can use the **Relships** menu function to obtain the object list of other relationships to other objects. In this way, you can navigate through the relationships from one object to another and stop wherever you wish in order to modify an object or to view or modify the comment for an object.

Relationship Lists in the Form of Object Lists

From the object list, you can access the submenu in which the object types related to the selected object type are displayed using the **Relships/Object Usage** menu item. If you select one of the object types without any further specification, a list of all objects that are related to the selected object is displayed immediately. This list cannot be limited.

The submenu lists all used objects and, after selecting the **Used by** menu item, the using objects for the selected object type.

Instead of selecting the **Relships/Object Usage** menu item, you can click on any cell within the object list using the right mouse button. A drop-down menu is displayed with the same functions as provided under the **Relships/Object Usage** menu function.

Hierarchical Relationship Lists

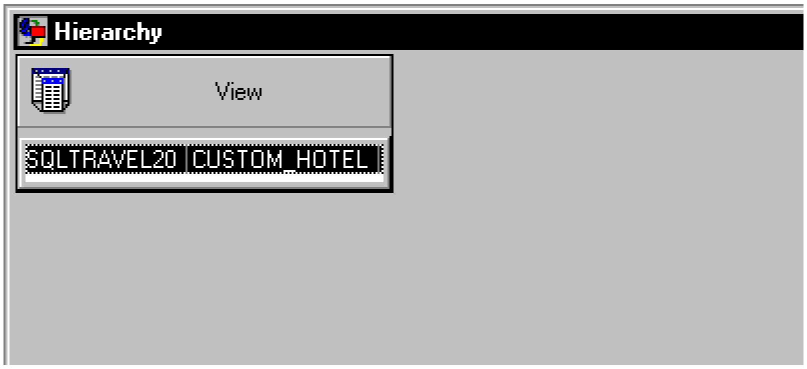
Select the **Relships/Hierarchy** menu item to open a new window in which the selected object is created as the root of relationship displays. Starting from this root, several relationship lists on different hierarchical levels can be displayed in a window. In contrast to the other object lists, only the key columns of each object are displayed.

List boxes which display the selected relationships can be designed in the same way as for the direct display of relationship lists in form of object lists.

First, select the desired object by clicking on the relevant line in the list box using the left mouse button. You can then select an objct type by using the **Relship/Object Usage** menu function. A list box containing all the objects that are in a relation with the selected object will be displayed.

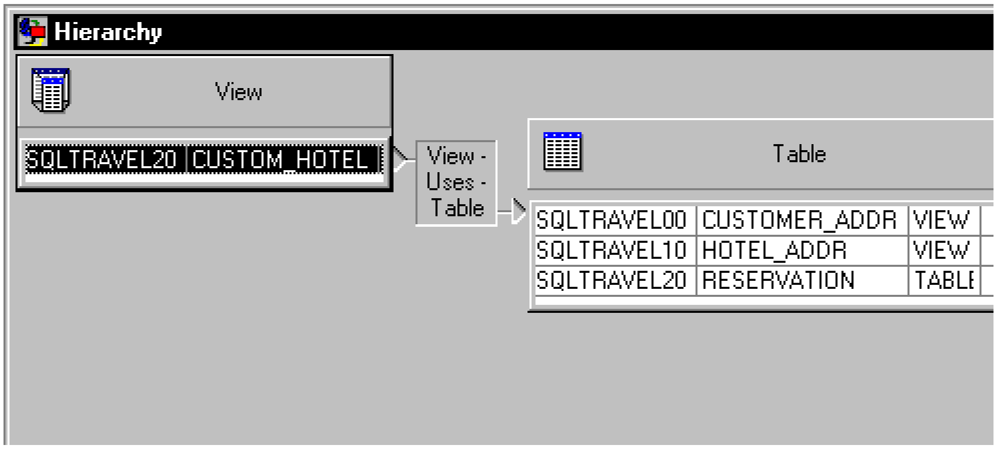
Pressing the right mouse button displays a drop-down menu with the same functions as provided under the **Relships/Object Usage** menu function.

In the following example, first, the root of the relationships hierarchy is created by selecting the **Relship/Hierarchy** function and an object from the View Object list:



SQLTRAVEL20	CUSTOM_HOTEL

Then the list of tables used by the view is created and displayed by using the **Relship/Object Usage/Table** function:

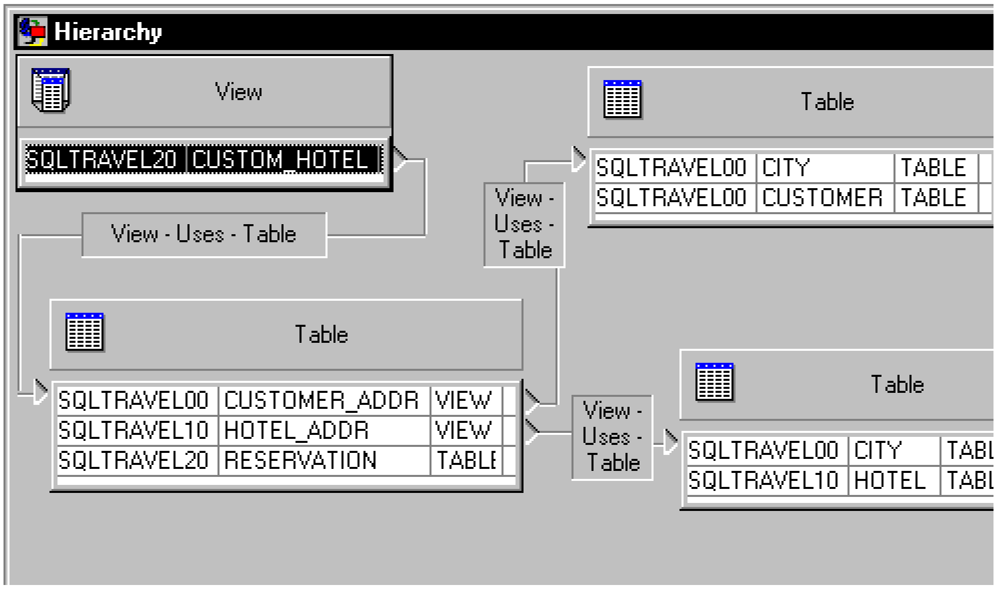


SQLTRAVEL20	CUSTOM_HOTEL

SQLTRAVEL00	CUSTOMER_ADDR	VIEW
SQLTRAVEL10	HOTEL_ADDR	VIEW
SQLTRAVEL20	RESERVATION	TABL

The relationship is represented by a line joining the basic object with the relationship list. The relationship name is displayed on this line within a frame. Arrows at the end of the line indicate the direction of the relationship (**uses** → or **used by** ←).

After selecting the lines of the second View List box and of the **Relship/Object/ Table** function one after the other, more relationships can be displayed. The list boxes are moved and the frames of the relationship names are changed giving the following representation:



SQLTRAVEL20	CUSTOM_HOTEL

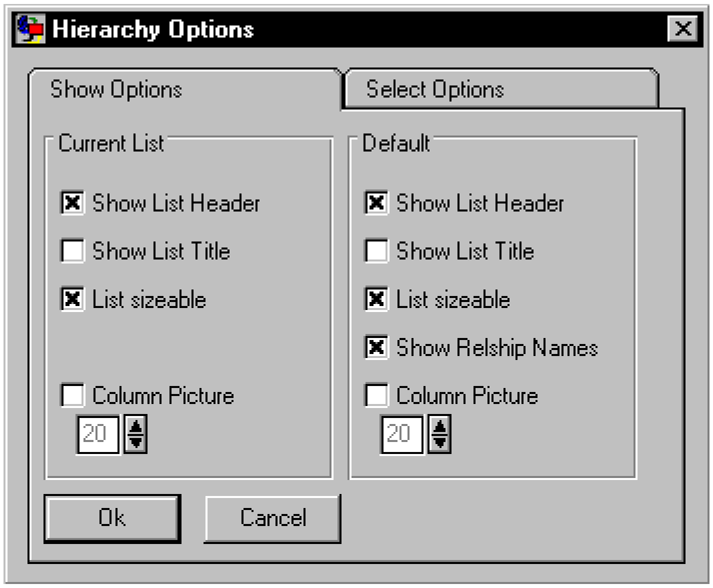
SQLTRAVEL00	CITY	TABLE
SQLTRAVEL00	CUSTOMER	TABLE

SQLTRAVEL00	CUSTOMER_ADDR	VIEW
SQLTRAVEL10	HOTEL_ADDR	VIEW
SQLTRAVEL20	RESERVATION	TABLE

SQLTRAVEL00	CITY	TABL
SQLTRAVEL10	HOTEL	TABL

Options for the Selection or Display of Hierarchy Lists

Hierarchy options can be used to change the selection of relationship data and its representation on the screen. Select the **View/Options** menu function to display the following option select box:



Current List

☒ Show List Header

□ Show List Title

☒ List sizeable

□ Column Picture

20

Default

☒ Show List Header

□ Show List Title

☒ List sizeable

☒ Show Relship Names

□ Column Picture

20

1. Options for the Display of Lists and Relationships

The options are grouped in those relating to the current list box (**Current List**) and in those relating to any list box created later (**Default**).

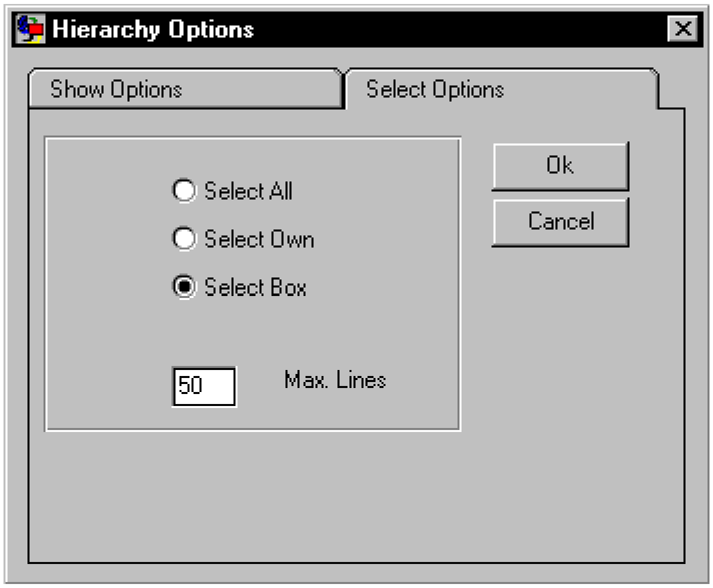
- **Show List Header**: displays the bar with the object icon and type.

- **Show List Title**: displays the title bar of the list with the column names.

- **List sizeable**: allows the size of the list to be changed by using the mouse.

- **Show Relship Names** (This option cannot be set for individual lists): displays the relationship names.

- **Column Picture**: displays the icon of the object type at the beginning of each list box line. Specify a value between 1 and 32 in the dialog box below the icon to set the size of the icon.



○ Select All

○ Select Own

● Select Box

50 Max. Lines

2. **Options to Select Data from the Relationship List (Select Relship)**

- **Select All**: displays all the objects in the relationship list.
- **Select Own**: displays all objects you own in the relationship list.
- **Select Box**: displays a dialog box in which the list to be created can be restricted by a search pattern.
- **MaxLines**: maximum number of lines displayed in a list box (1 <= **MaxLines** <= 999). If the number of objects found exceeds the specified value (**MaxLines**), a message is displayed. To restrict a result set, you can, e.g., use a **Select Box**.

Positioning and Sizing Lists and Relationship Names

Each new list box is placed to the right of the list box of the basic object. The list boxes and frames of relationship names can be moved in any direction within the window using the mouse. The column width and size of the list box can be changed (**List sizeable**).
After changing a list box (in size or position), the position of the relationship names is automatically adapted.
To change the size of the frame of a relationship name, double-click on the relationship name to set the frame to "sizeable"; then drag the frame at the size handles into the desired size using the mouse. To disable the sizing mode, double-click on the background of the window.
As soon as the relationship names have been changed in size or position, their position is no longer automatically adapted to the position of the list boxes.

Each list can be removed by using the **Edit/Delete** menu function. If **Edit/Delete** is activated for the last list in the window, the window is closed.

The **View/SizeToFit** menu function adapts the width of the list box to the current column widths.

All functions common for the object lists are available. In particular, it is possible to use the **Relship/Hierarchy** menu function to open another window for the display of relationship hierarchies.

All functions have an effect on the selected object or list box.

Uppercase/Lowercase Letters in Object Names

The names of DB objects are automatically converted into uppercase letters when entered. If you wish to define object names containing lowercase letters or special characters, you must enclose these characters in double quotation marks. The object name does not appear in the specified form until you enter the first double quotation mark. Object names identical to a keyword must also be enclosed in double quotation marks.

The object names or search patterns are specified without double quotation marks in dialog boxes where search patterns are entered. In this case, the characters are case sensitive.

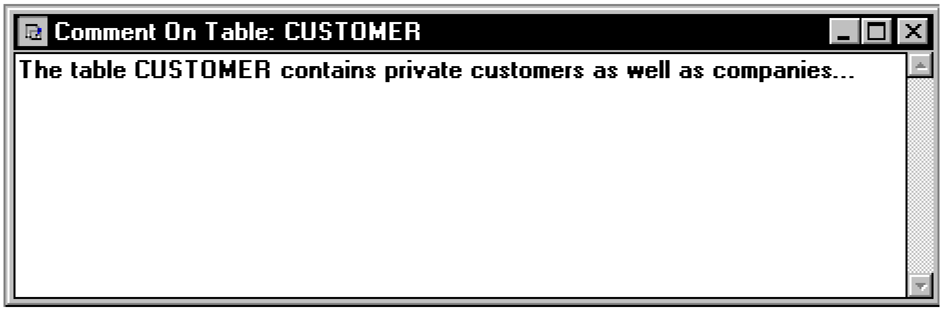
Comment Management

DOMAIN allows you to create a comment for every object you own. All users who are authorized to view an object can also read the object's comment.

Using the **Edit/Comment** function or the 📝 toolbar button, you can create (as the owner) or view the comment in all object lists, including those in the relationship display. If the object does not have a comment, an empty editing window is displayed in which the owner of the object can enter and save a comment.

The menu bar of the comment window provides the known editing functions. Use the **File (Load Text** or **Save Text)** menu item to load the comment from or write it to a file.

The comment can also be accessed in design mode. In the case of more complex objects (e.g., tables), select the **Edit/Comment** menu item to display a submenu from which you must select either ***Table*** or the individual, associated subobjects ***Column***, ***Constraint***, ***Foreign Key***, ***Index,*** and ***Trigger***. If the ***Column*** object is selected, you can choose either the table comment or the column comment.

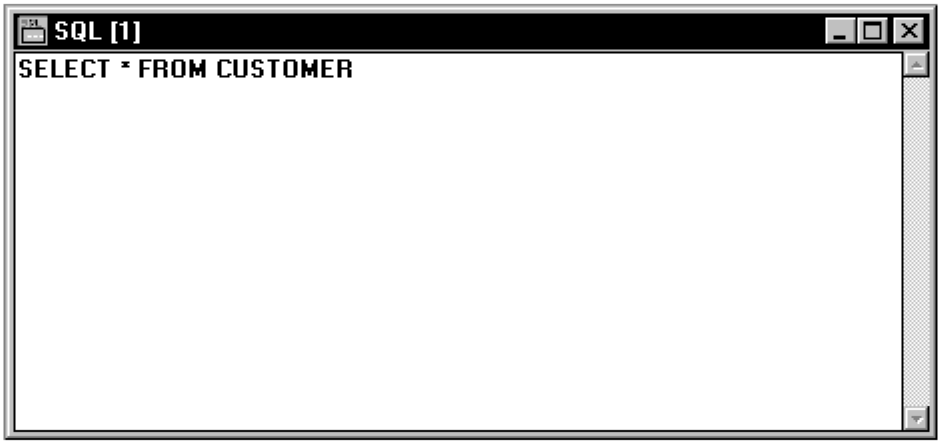


The table CUSTOMER contains private customers as well as companies...

If you like, you can also view the comments of all objects within an object list by selecting the **View/Options** function and activating the **Show Comments in Object Lists** option under **More Options**.

SQL Window

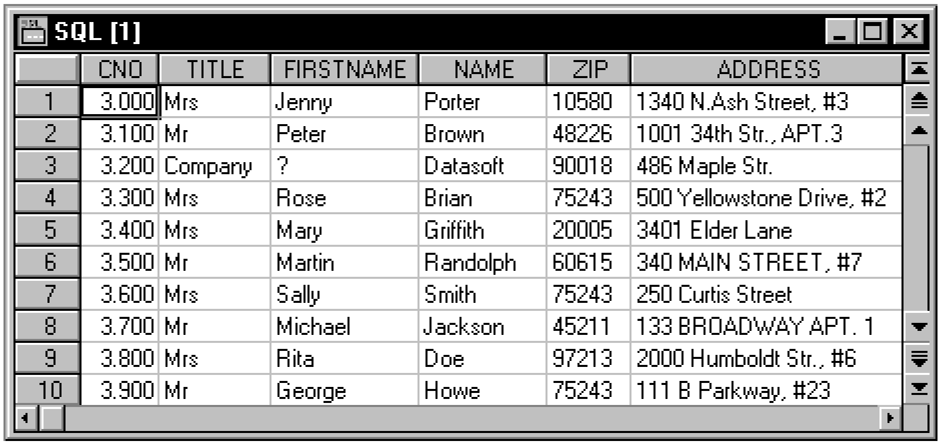
DOMAIN allows you to formulate and execute any SQL statements in the SQL window. Select the **Object/SQL Window** or the SQL toolbar button to open a window in which you can edit the statement.



SQL [1]

```
SELECT * FROM CUSTOMER
```

To execute the SQL statement, click on the ! toolbar button, or press the key combination Ctrl / Enter. The result of the request will be displayed.

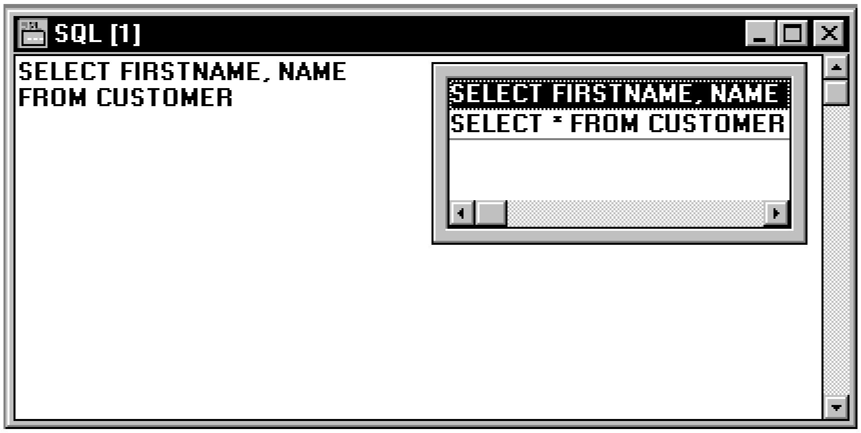


SQL [1]

	CNO	TITLE	FIRSTNAME	NAME	ZIP	ADDRESS
1	3.000	Mrs	Jenny	Porter	10580	1340 N.Ash Street, #3
2	3.100	Mr	Peter	Brown	48226	1001 34th Str., APT.3
3	3.200	Company	?	Datasoft	90018	486 Maple Str.
4	3.300	Mrs	Rose	Brian	75243	500 Yellowstone Drive, #2
5	3.400	Mrs	Mary	Griffith	20005	3401 Elder Lane
6	3.500	Mr	Martin	Randolph	60615	340 MAIN STREET, #7
7	3.600	Mrs	Sally	Smith	75243	250 Curtis Street
8	3.700	Mr	Michael	Jackson	45211	133 BROADWAY APT. 1
9	3.800	Mrs	Rita	Doe	97213	2000 Humboldt Str., #6
10	3.900	Mr	George	Howe	75243	111 B Parkway, #23

Click on the toolbar button or press the key combination Ctrl / Enter to return from the result display to the window with the SQL statement. You can use the ✕ toolbar button to clear the editing window for the input of a new SQL statement. The and toolbar buttons can be used to display the previous or next SQL statement.

In the right upper quarter of the SQL window, the ▤ toolbar button displays the list of all SQL statements executed since the opening of the SQL window.



```
SELECT FIRSTNAME, NAME
FROM CUSTOMER
```

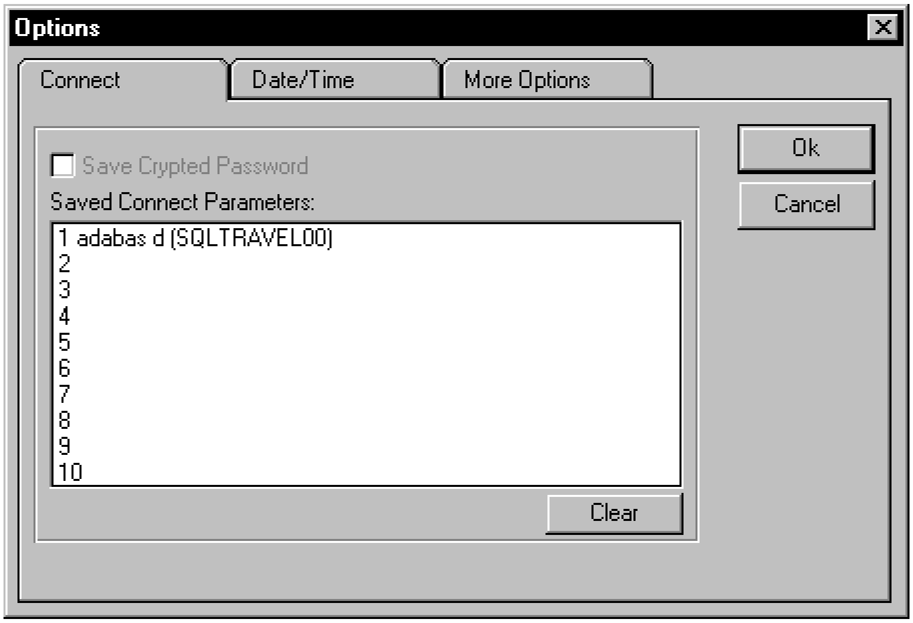
```
SELECT FIRSTNAME, NAME
SELECT * FROM CUSTOMER
```

Double-click on a statement to fetch it from the list into the SQL window. There it can be executed again. Each statement entered newly is included in the list. A statement fetched from the list is only re-included in the list if it has been modified.

The result of the SQL statement can be changed as described in the chapter 'Object Lists', section 'Object Lists'.

Settings

***Connect* Options**



□ Save Crypted Password

Saved Connect Parameters:

1 adabas d (SQLTRAVEL00)
2
3
4
5
6
7
8
9
10

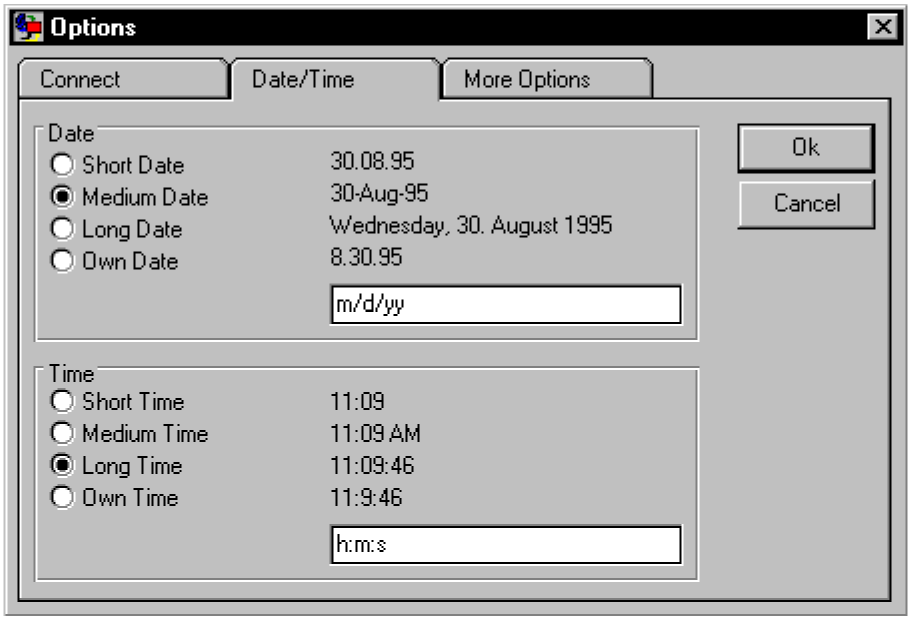
DOMAIN stores the connect parameters of the session during a connect in order to use them for a new connect. If you use the **Save Crypted Password** option, DOMAIN also stores the password so that no entries are needed when connecting again. For a non-saved password, DOMAIN requires you to enter the password again.
The **Save Crypted Password** option has only an effect on the password of the current session.

All connect parameters stored so far are displayed in the list **Saved Connect Parameters**. You can mark individual connect parameters or groups of them, and delete them by using the **Clear** toolbar button.

Date/Time Formats

DOMAIN uses the date (short and long date format) and time (long time format) formats from the Windows settings. The date format **Medium Date** as well as the time formats **Short Date** and **Medium Date** are usual formats provided for selection in addition; however, they cannot be modified. DOMAIN also offers its own free format that can be defined in the input field. The notational conventions are those used in Windows.

In this example, **Medium Date** has been selected as date format and **Long Time** as time format.



Date

○ Short Date 30.08.95
● Medium Date 30-Aug-95
○ Long Date Wednesday, 30. August 1995
○ Own Date 8.30.95

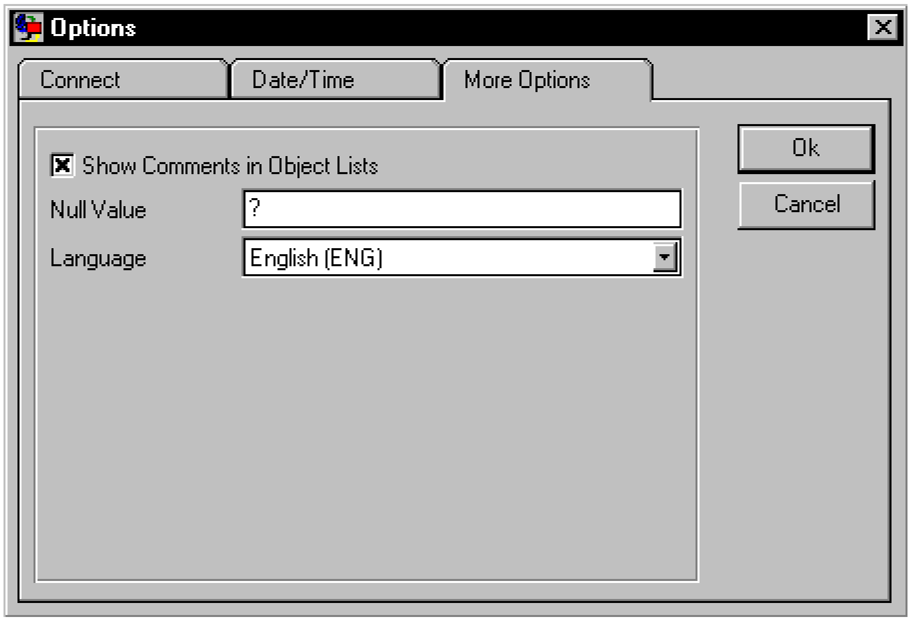
m/d/yy

Time

○ Short Time 11:09
○ Medium Time 11:09 AM
● Long Time 11:09:46
○ Own Time 11:9:46

h:m:s

Display of Comments and Representation of Null Values



☒ Show Comments in Object Lists

Null Value ?

Language English (ENG)

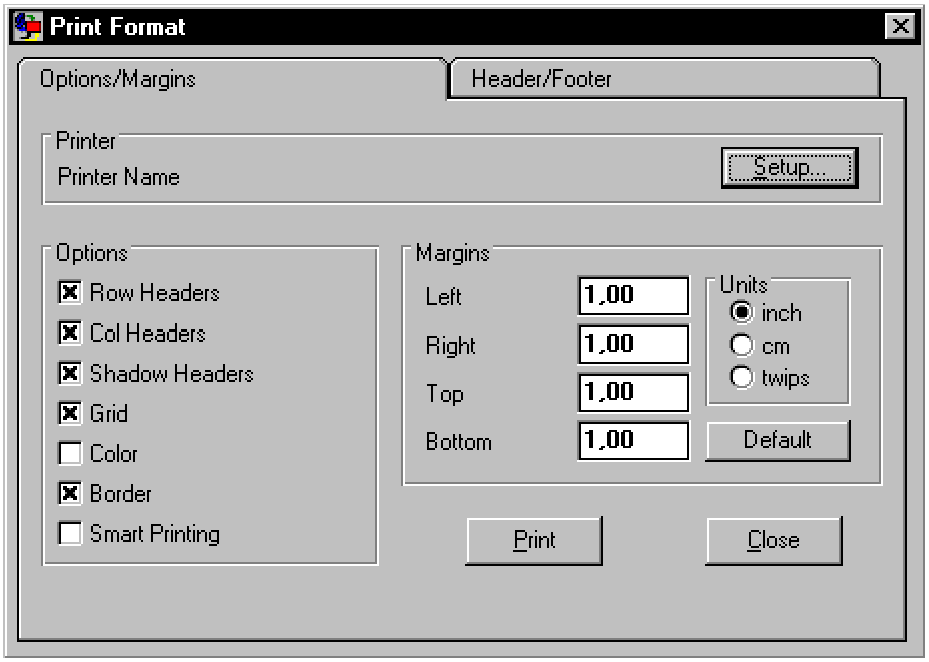
You can use the **Show Comments in Object Lists** option to choose whether or not the comments for an object are always to be displayed as the last column in the object list.

You can use the null value option to specify how null value database contents are to be displayed.

DOMAIN uses the setting of Windows for the language of the help information (English or German). If you want to use for DOMAIN a language version other than that set in Windows, you can specify the language in the **Language** field.

Printing

DOMAIN allows you to print all object lists and relationship displays. If you click on the 🖨 toolbar button or select the **Object/Print** function, the following dialog box is displayed:



Printer
Printer Name

Options

☒ Row Headers
☒ Col Headers
☒ Shadow Headers
☒ Grid
☐ Color
☒ Border
☐ Smart Printing

Margins

Left 1,00
Right 1,00
Top 1,00
Bottom 1,00

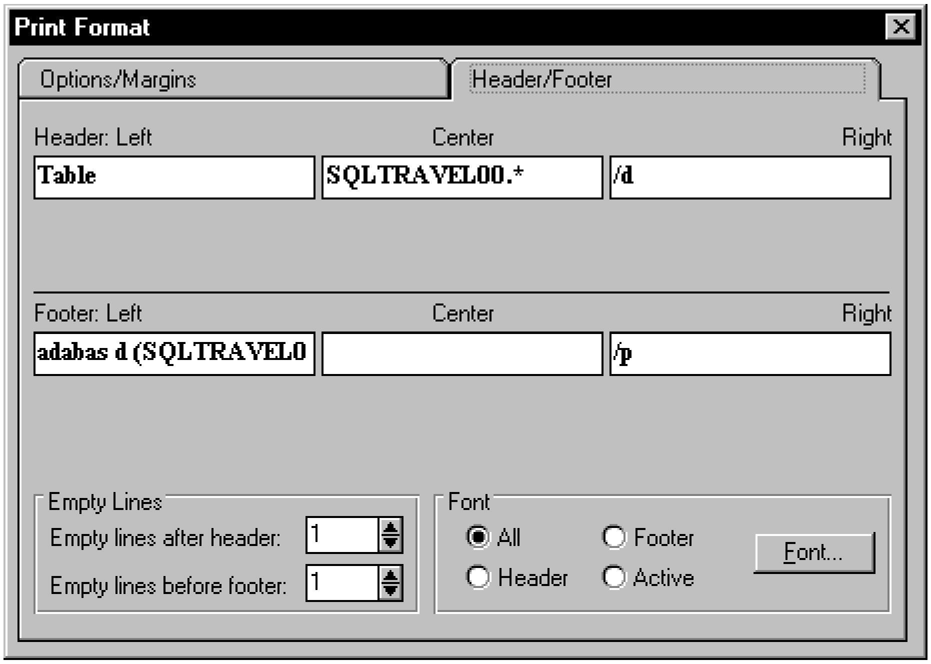
Units
● inch
○ cm
○ twips

Use the **Setup** button to access the Windows printer setup parameters. With **Options**, you can determine whether the row numbers (**Row Headers**) or column names (**Col Headers**) are to be printed as well. **Shadow Headers** has the effect that the row numbers and column names are shaded when printed. **Grid** indicates whether grid lines are desired in the printout. **Color** prints the colors as displayed. **Border** provides the list with a frame.

The **Smart Printing** option attempts to print lists that are too wide for a print page on one page applying the following procedure:

1. The attempt is made to print the list in landscape mode.
2. If the list is too wide for the landscape mode, the list is reduced up to 60% of its original size.
3. If the list is still too wide after reduction, the column widths are adapted according to the longest column contents.
4. If the last step does not produce the desired result, the list is printed unmodified.

Use the **Default** button to set the default values for the margins.



Header: Left
Table
Center
SQLTRAVEL00.*
Right
/d

Footer: Left
adabas d (SQLTRAVEL0
Center
Right
/p

Empty Lines
Empty lines after header: 1
Empty lines before footer: 1

Font
● All ○ Footer
○ Header ○ Active

On the second page of the printer setup, you can define the contents and fonts of the headers and footers, as well as the number of blank lines below the header or above the footer.

The Toolbar

The toolbar always contains the most important functions for the current DOMAIN window. A brief explanation of each toolbar button is displayed when you position the mouse pointer over it without pressing a button. The toolbar is described in greater detail below.

: Opens a new database session.

: Provides a list of all objects that you are authorized to view for the selected object type.

: Provides a list of all objects you own for the selected object type.

: Provides a dialog box for selecting an object list.

: Changes to design mode for the purpose of creating a new object.

: Changes to design mode for the purpose of displaying and modifying the selected object.

: Provides a selection list of tables for defining ***Views*** and ***Snapshots,*** or a ***Table as Query***.

: Opens a file, for example, to load an SQL statement or comment.

: Saves the contents of the editing window in a file.

: Saves the object in the databse.

: Executes the SQL statement formulated in the SQL window.

: Editing function for removing the selected area (**Edit/Cut**).

: Editing function for copying the selected area (**Edit/Copy**).

: Editing function for inserting the copied/removed area (**Edit/Paste**).

: Clears the editing window.

: Displays the comment for the object selected in the object list; the owner is authorized to modify the comment.

🔍: Zoom function; provides an editing window to facilitate input for some input fields.

AZ↓: Sorts column(s) in tabular presentation in ascending order.

ZA↓: Sorts column(s) in tabular presentation in descending order.

SQL: Opens an editing window for formulating an SQL statement.

↑: Displays the previous SQL statement in the SQL window.

↓: Displays the next SQL statement in the SQL window.

✎: Returns from the result table to the SQL statement.

↙: Executes the **Records/Refresh** function.

123: Shows statistical information about the chosen object in a ***Table, Index,*** or ***User*** list.

→: Inserts a blank line before the inserted marked row (**Edit/Insert Row)**.

→: Deletes the marked row and represents the row as deleted on display (**Edit/Delete Row)**.

←: Cancels the delete (**Edit/Undelete Row)**.

☰: Displays a list of all SQL statements executed since the opening of the SQL window.

🖶: Prints the displayed object list or relationship display (**Object/Print)**.

?: Provides general help information.

Object Editing

You can access either Object editing to define a new object or modify the object selected using the menu item Object/New or Object/Design or click on then toolbar buttons 🗋 or 📝. There is no design mode for application objects.

Table
Defining Columns in a Table
Defining Constraints
Defining and Deleting Foreign Keys
Index
Triggers
View
Synonym
Domain
User
DB Procedure
DB Function
Snapshot

Table

Select the **Object/New** menu function and then **Table** from the submenu displayed, or click on the □ toolbar button to enter design mode where you can define a new table. Both operations are available in the object menu, object list, and relationship display.

The ***Table*** object describes an ADABAS table. A table is either a base, view, synonym, snapshot, or result table; i.e., the table list contains ***View***, ***Synonym***, and ***Snapshot*** objects ('Type' column in the table list). Each user has defined access privileges (or none at all) for a specific table.

Relationships between tables and other objects

1.	Table	Contains	Column
2.	Table	Uses	Constraint
3.	Table	Contains	Trigger
4.	Easycommand	Uses	Table
5.	Easyform	Uses	Table
6.	Foreignkey	Refers	Table
7.	Module	Uses	Table
8.	Qpcommand	Uses	Table
9.	Qpquery	Uses	Table
10.	Querycommand	Uses	Table
11.	Snapshot	Uses	Table
12.	Synonym	Refers	Table
13.	User	Uses	Table
14.	View	Uses	Table

1. A table comprises one or more columns; each column is assigned to exactly one table (1-N).

2. A table can use a number of constraints; a constraint can be used by a number of tables (N[0]-M[0]).

3. A table can contain triggers; each trigger is assigned to exactly one table (1-N[0]).

4. An EASY command uses one or more tables; a table can be used by an EASY command (N[0]-M).

5. An EASY form uses a table; a table can be used in an EASY form (N[0]-M).

6. A foreign key refers to exactly one primary table; the primary key for a table can be assigned to a foreign key of another table (0-1).

7. A module can use a number of tables; a table can be used by a number of modules (N[0]-M[0]).

8. A QueryPlus command uses one or more tables; a table can be used by a

number of QueryPlus commands (N[0]-M).

9. A QueryPlus query uses one or more tables; a table can be used by a number of QueryPlus queries (N[0]-M).

10. A QUERY command uses one or more tables; a table can be used by a number of QUERY commands (N[0]-M).

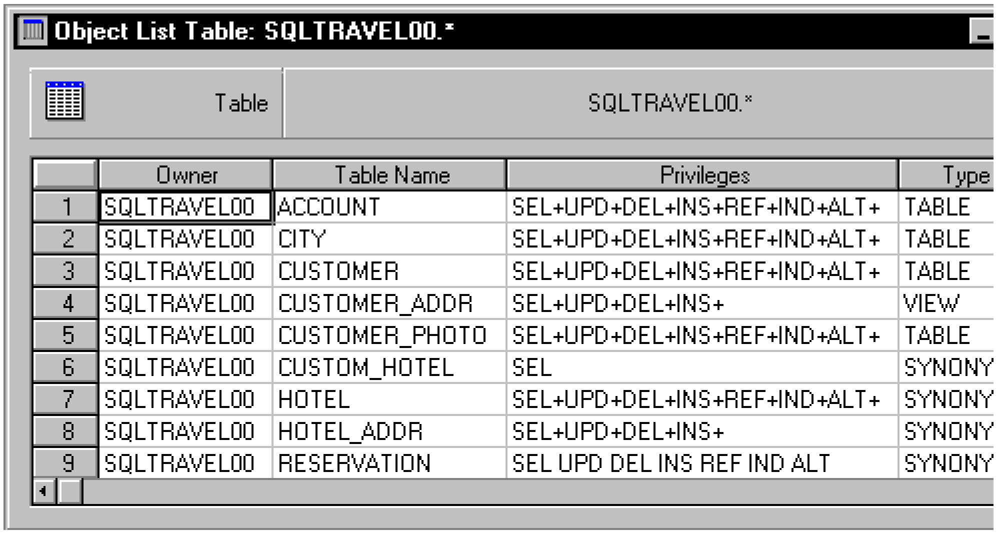
11. A snapshot refers to one or more tables; a table can be used by a snapshot (N[0]-M).

12. A synonym refers to a base or view table; a number of synonym definitions can exist for a table (N[0]-1).

13. A user can use a number of tables; a table can be used by one or more users (N-M[0]).

14. A view table refers to one or more tables; a table can be used by a view table (N[0]-M).

Example of a list of all tables for the user SQLTRAVEL00:



	Owner	Table Name	Privileges	Type
1	SQLTRAVEL00	ACCOUNT	SEL+UPD+DEL+INS+REF+IND+ALT+	TABLE
2	SQLTRAVEL00	CITY	SEL+UPD+DEL+INS+REF+IND+ALT+	TABLE
3	SQLTRAVEL00	CUSTOMER	SEL+UPD+DEL+INS+REF+IND+ALT+	TABLE
4	SQLTRAVEL00	CUSTOMER_ADDR	SEL+UPD+DEL+INS+	VIEW
5	SQLTRAVEL00	CUSTOMER_PHOTO	SEL+UPD+DEL+INS+REF+IND+ALT+	TABLE
6	SQLTRAVEL00	CUSTOM_HOTEL	SEL	SYNONY
7	SQLTRAVEL00	HOTEL	SEL+UPD+DEL+INS+REF+IND+ALT+	SYNONY
8	SQLTRAVEL00	HOTEL_ADDR	SEL+UPD+DEL+INS+	SYNONY
9	SQLTRAVEL00	RESERVATION	SEL UPD DEL INS REF IND ALT	SYNONY

To create a new table, click on the □ toolbar button or select the **Object/New/Table** menu item in the table's design mode.

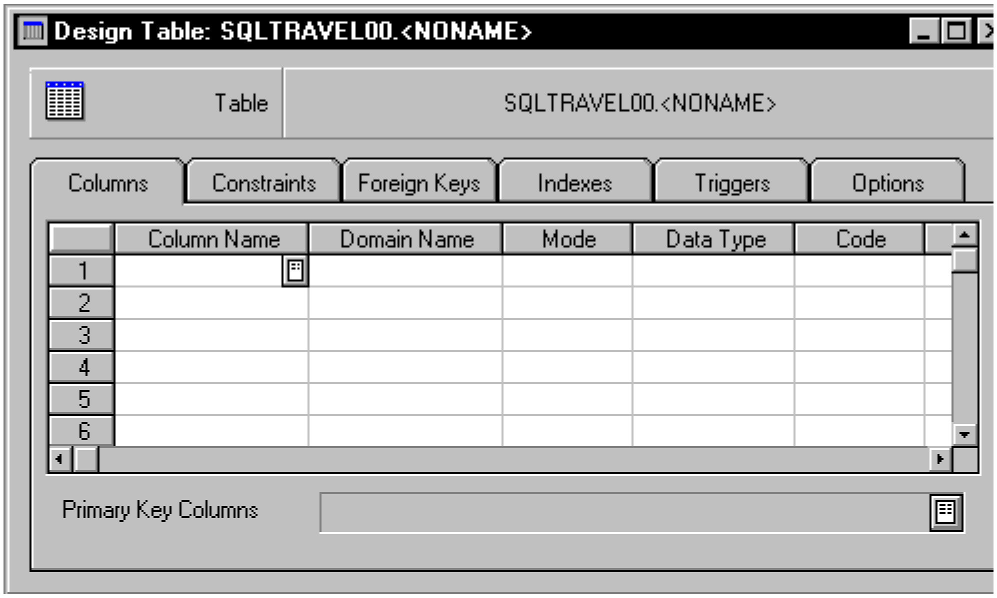

Design Table: SQLTRAVEL00.<NONAME>

Table SQLTRAVEL00.<NONAME>

The table definition comprises a number of parts:

- Columns
- Primary key
- Constraints
- Foreign keys
- Indexes
- Triggers
- Options

These components are displayed in the Table Design window on separate cardfile cards. In order to create a table, the ***Columns*** card must contain at least one definition; all the other cards can be blank.

A table is defined by filling out the cards entry by entry and in any order. You can repeat and correct this procedure as often as you wish and terminate it by selecting the **Object/Save** menu item. The table name is not entered until this point. If you exit design mode using the **Object/Close** function, DOMAIN asks whether the changes should be saved as a table definition.

See also

Defining Columns in a Table

Defining Constraints

Defining and Deleting Foreign Keys

Defining and Deleting an Index

Displaying Triggers

Defining Columns in a Table

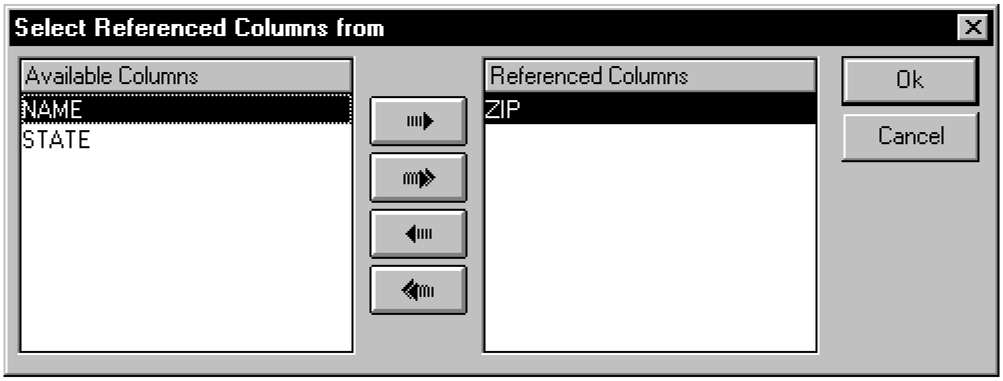
All the columns in the Table Design window are arranged in a tabular form. Here you can define the columns by entering their attributes. Press TAB to change from one attribute box to another. When you enter the column name, DOMAIN enters default values in the remaining attribute boxes in order to save you typing time; you can, of course, overwrite these values as desired.

The **Domain Name, Mode,** and **Default** attributes are optional. If you select 'CHAR' as the **Data Type**, the **Dec** attribute is no longer relevant. DOMAIN responds to these dependencies by deactivating irrelevant attributes and shading in the corresponding boxes.

DOMAIN conveniently assists you in defining attributes. As soon as an attribute box is activated, a button is displayed on the right-hand side. When you click on the ▤ button for **Column Name** or **Domain Name**, a list of column or domain definitions appears. Simply click on the desired column or domain definition in the list to transfer it as a definition of the current column or domain. To view a list other than the standard list, you can request a dialog box for entering search arguments by using the **Select** command button.

The **Mode, Data Type, Code,** and **Default** attributes are displayed as combo boxes that provide you with a list of all permissible values for selection. You can define individual values that differ from the default values only in the case of **Default.** The default value must be enclosed in apostrophes. The **Length** and **Dec** attributes can be set by clicking the ⬍ button without typing an entry.

You can either enter the **Mode** attribute immediately or define it by separately defining the primary key. You must enter the key columns, in the desired order and separated by semicolons, in the primary key box below the column definitions. You can also click on the ▤ button to change to a more convenient mode where you can transfer the key columns from the list of all columns in the order desired.



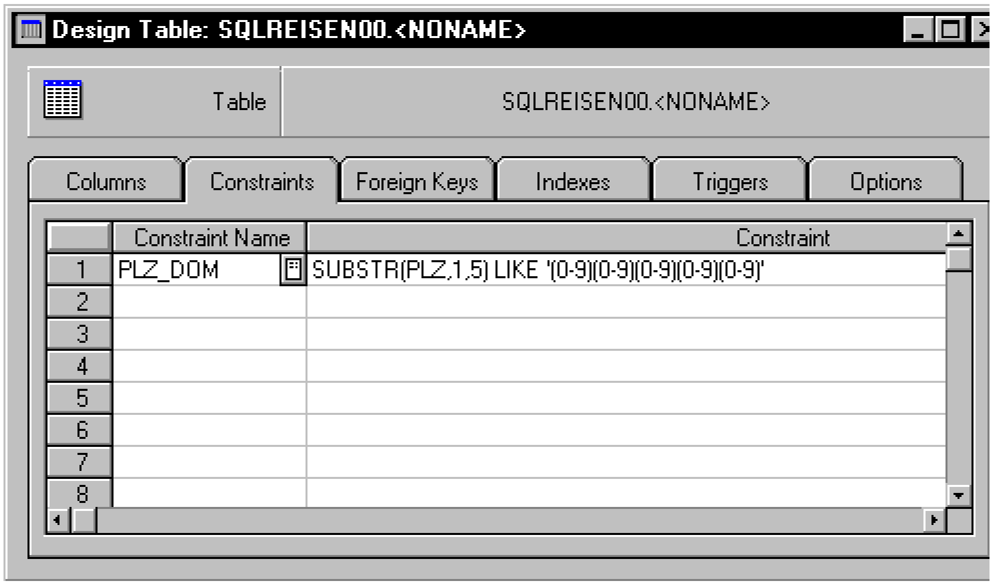
Define the key columns by moving them from the existing columns of the table in the left-hand list to the right-hand list. To do so, click on the arrow buttons; the order of the key columns is important.

The result of this type of definition is then displayed as a list of column names in the primary key box.

Subsequent changes to the key column definition for an existing table are possible only to a limited extent (see the ADABAS Reference online help).

Defining Constraints

When you click on the ***Constraints*** card, the Constraint Definition window is displayed.



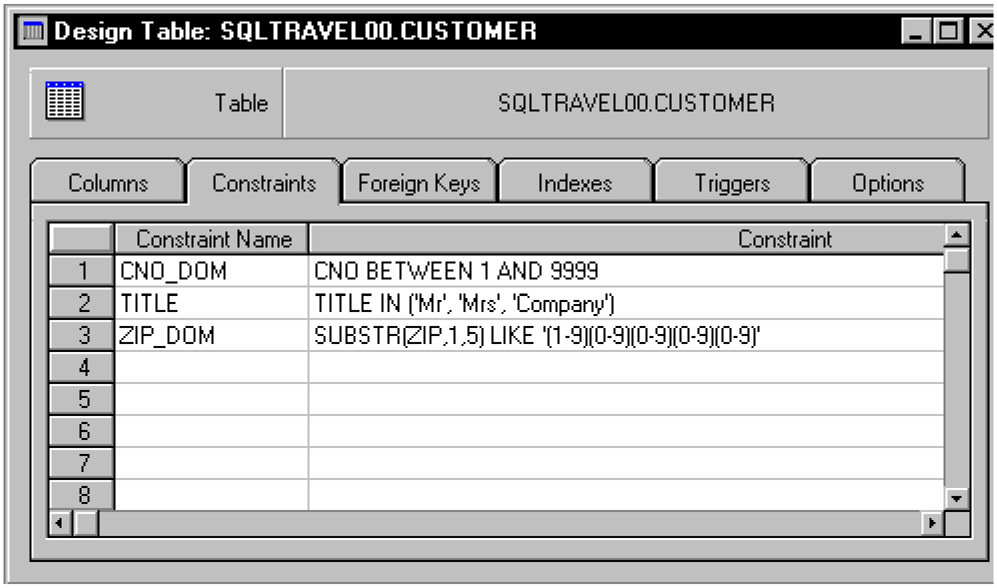
Design Table: SQLREISEN00.<NONAME>

Table SQLREISEN00.<NONAME>

	Constraint Name	Constraint
1	PLZ_DOM	SUBSTR(PLZ,1,5) LIKE '(0-9)(0-9)(0-9)(0-9)(0-9)'
2		
3		
4		
5		
6		
7		
8		

As in the case of column definitions, constraints are either entered directly into the tabular presentation or transferred from existing definitions; in the latter case, you must click on the ▤ toolbar button for **Constraint Name**. Constraint definitions apply to the whole table. You can use the **Edit-Zoom** function or the 🔍 toolbar button to facilitate the input.

If you use ***Domains*** that contain a ***Constraint*** definition for the definition of columns, these ***Constraints*** are also represented on the table definition ***Constraints*** card.



	Constraint Name	Constraint
1	CNO_DOM	CNO BETWEEN 1 AND 9999
2	TITLE	TITLE IN ('Mr', 'Mrs', 'Company')
3	ZIP_DOM	SUBSTR(ZIP,1,5) LIKE '(1-9)(0-9)(0-9)(0-9)(0-9)'
4		
5		
6		
7		
8		

Defining and Deleting Foreign Keys

When you click on the ***Foreign Keys*** card, the Foreign Key Definition window is displayed.

A foreign key links two tables such that the primary key (or the identifying columns) of the primary table appears in the secondary table. The **primary table** is the **referenced table**; it must already exist in order for the foreign key to be defined.

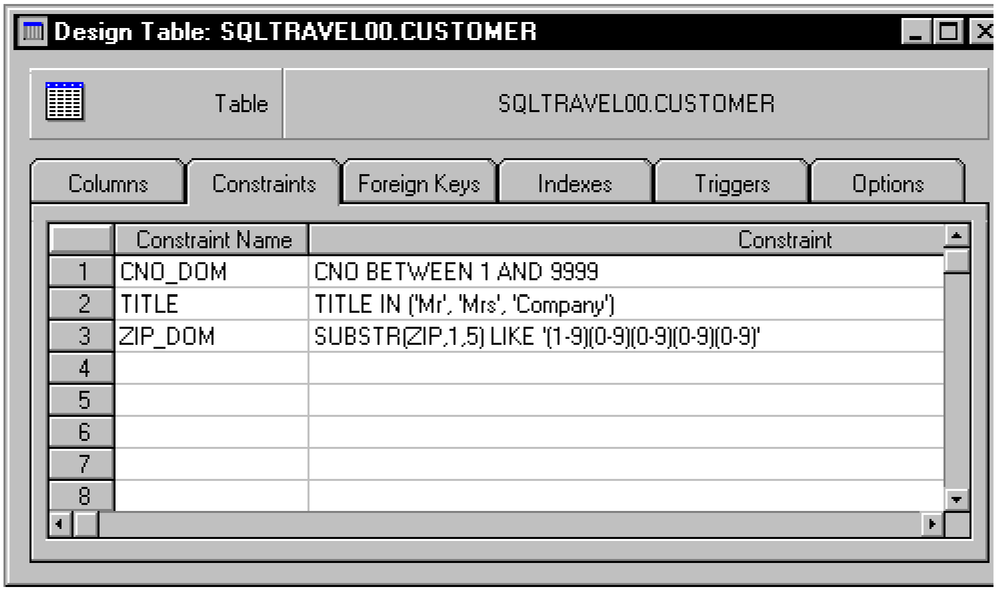
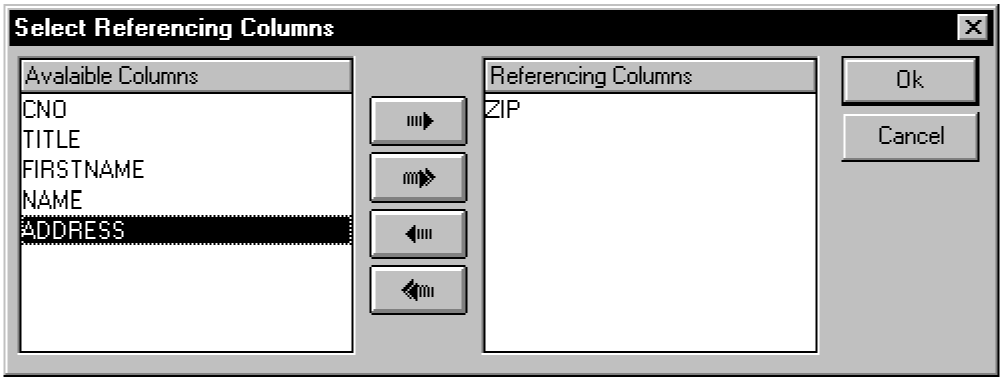

Design Table: SQLTRAVEL00.CUSTOMER

Table: SQLTRAVEL00.CUSTOMER

	Constraint Name	Constraint
1	CNO_DOM	CNO BETWEEN 1 AND 9999
2	TITLE	TITLE IN ('Mr', 'Mrs', 'Company')
3	ZIP_DOM	SUBSTR(ZIP,1,5) LIKE '(1-9)(0-9)(0-9)(0-9)(0-9)'
4		
5		
6		
7		
8		

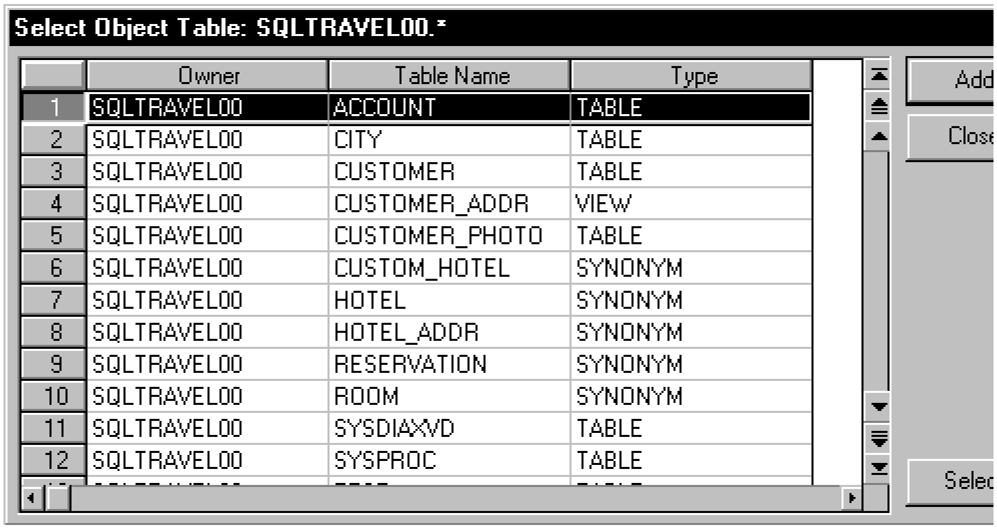
Once again, you have the option of entering the names directly in tabular form or allowing DOMAIN to assist you. When entering the names in tabular form, you must enter the column names, separated by semicolons, under the **Referencing Columns** and **Referenced Columns** headings. As in the case of primary key definitions, DOMAIN displays the columns of the primary or foreign table for the relevant selection. DOMAIN then displays the selected columns in tabular form.

In the case of **Referencing Columns**, the columns of the current table are displayed so that you can select the corresponding foreign key columns.


Select Referencing Columns

Avalaible Columns	Referencing Columns
CNO	ZIP
TITLE	
FIRSTNAME	
NAME	
ADDRESS	

In the case of the **Referenced Table**, a dialog box is displayed containing a list of tables from which you can select the relevant primary table.



Select Object Table: SQLTRAVEL00.*

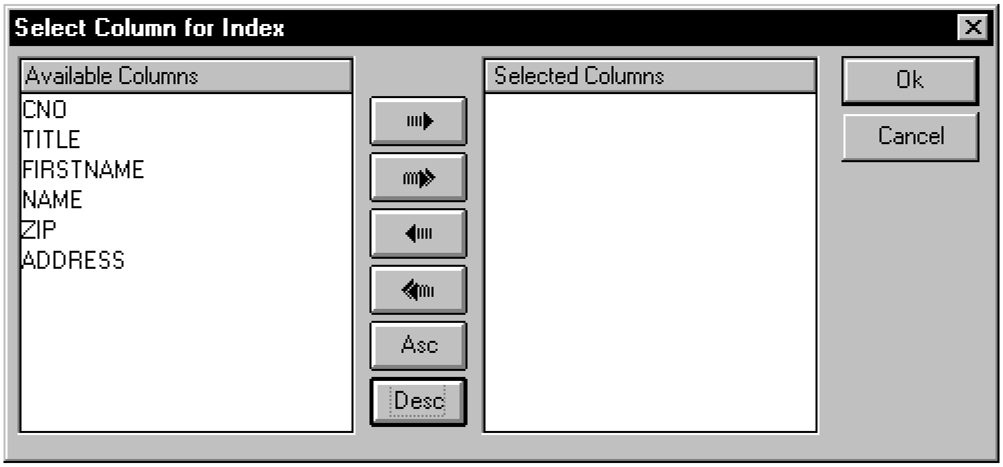
	Owner	Table Name	Type
1	SQLTRAVEL00	ACCOUNT	TABLE
2	SQLTRAVEL00	CITY	TABLE
3	SQLTRAVEL00	CUSTOMER	TABLE
4	SQLTRAVEL00	CUSTOMER_ADDR	VIEW
5	SQLTRAVEL00	CUSTOMER_PHOTO	TABLE
6	SQLTRAVEL00	CUSTOM_HOTEL	SYNONYM
7	SQLTRAVEL00	HOTEL	SYNONYM
8	SQLTRAVEL00	HOTEL_ADDR	SYNONYM
9	SQLTRAVEL00	RESERVATION	SYNONYM
10	SQLTRAVEL00	ROOM	SYNONYM
11	SQLTRAVEL00	SYSDIAXVD	TABLE
12	SQLTRAVEL00	SYSPROC	TABLE

Once a **Referenced Table** is defined, DOMAIN automatically transfers the key columns of the **Referenced Table** to **Referenced Columns**. If you do not wish to retain these columns, click on the ▤ button to change to selection mode. You can then select the desired **Referenced Columns** as well as the **Referencing Columns**.

To delete a ***Foreign Key***, delete the definition from the table display.

Defining and Deleting an Index

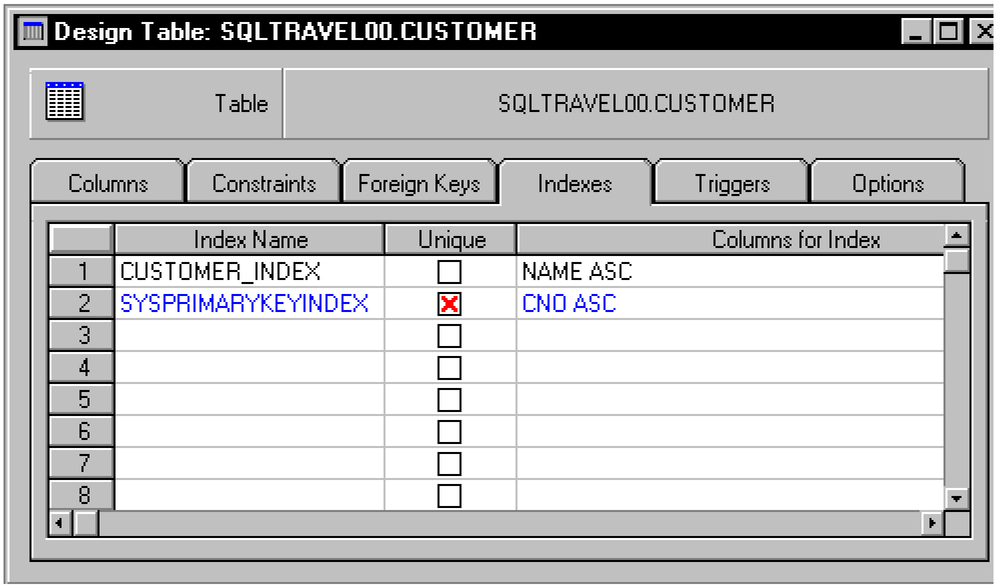
When you click on the ***Index*** card, the Index Definition window is displayed. As with the ***Foreign Key***, you can define a number of indexes for each table. The index can be defined either in tabular form or with the aid of DOMAIN. In the first case, you must enter the columns one after the other, separated by semicolons. In the latter case, a list of columns is displayed just as it is for primary key and foreign key definition.



Select Column for Index

Available Columns	Selected Columns
CNO	
TITLE	
FIRSTNAME	
NAME	
ZIP	
ADDRESS	

Click on the **Asc** and **Desc** command buttons to define the sorting mode for index columns. The **Unique** option defines whether the index is to be an identifying index.



Design Table: SQLTRAVEL00.CUSTOMER

Table: SQLTRAVEL00.CUSTOMER

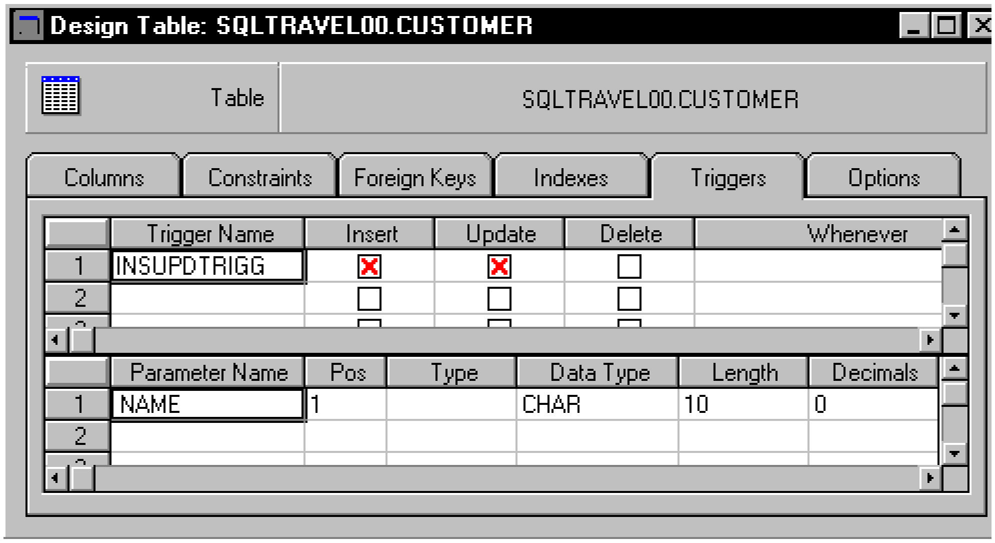
	Index Name	Unique	Columns for Index
1	CUSTOMER_INDEX	☐	NAME ASC
2	SYSPRIMARYKEYINDEX	☒	CNO ASC
3		☐	
4		☐	
5		☐	
6		☐	
7		☐	
8		☐	

To delete an index, delete the definition line from the table display.

Displaying Triggers

When you click on the ***Trigger*** card in table design mode, the trigger display appears as a master/detail display. This display comprises two lists, one above the other. The top list contains the trigger names and their attributes. The bottom list contains the parameter definitions for the trigger selected in the top list.

Example of a trigger display:



Design Table: SQLTRAVEL00.CUSTOMER

Table SQLTRAVEL00.CUSTOMER

	Trigger Name	Insert	Update	Delete	Whenever
1	INSUPDTRIGG	☒	☒	□	
2		□	□	□	

	Parameter Name	Pos	Type	Data Type	Length	Decimals
1	NAME	1		CHAR	10	0
2						

The condition and comment for each individual trigger is located in the top part of the display and can be easily displayed in a larger window using the zoom function 🔍.

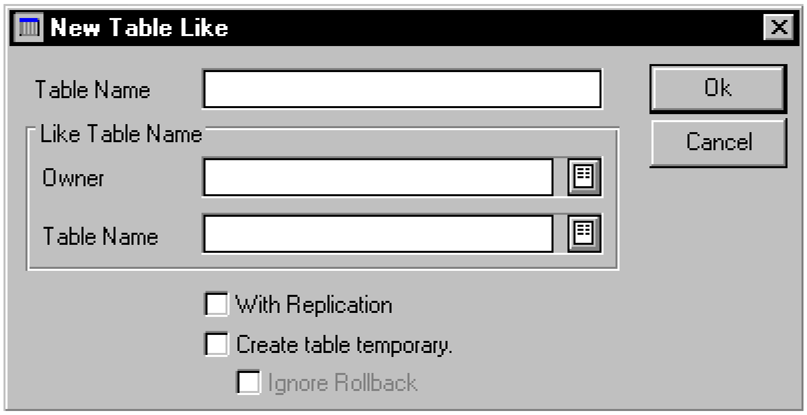
Options When Defining Tables

On the ***Options*** card for the table design, you can set the **With Replication** option for the table to be defined. This option has an effect only in a distributed database (see the ADABAS Reference online help).

For all simple **Snapshot** tables based on the table to be defined, a log of the modifications will be maintained if the **With Snapshot Log** option is set. If the **Snapshot Log** exists and the REFRESH <snapshot> statement (see the ADABAS Reference online help) is issued, the complete **Snapshot** content is not transferred, but only the modifications that affect the **Snapshot**.

Defining a Table Like Another Table (CREATE Table Like)

From a table list, select the **Object/New** menu function followed by the **Table Like** submenu item to display the dialog box shown below for entering table names. The **Like Table** text box contains the name of the table selected from the table list.



Once again, DOMAIN offers you the option of displaying a table list from which you can select the relevant table. To do so, click on the ▤ button to the right of the **Like Table Name** or **Owner** text box.

A selection list of all your tables appears. If the desired table is not one of your tables, you can use the **Select** command button to request a dialog box from which you can select any desired table list.

If you call the same **Object/New/Table Like** function from the object type list, the **Like** table contains no entries.

In a distributed configuration, the **With Replication** option allows you to specify whether or not the table is to be replicated (see the With Replication option of the CREATE TABLE statement).

Use the **Create table temporary** option to create a temporary table which is only kept for the session of the current user. When doing so, it is possible to set the option **Ignore Rollback** which has the effect that the table is not subject to transaction mechanism.

Click on the **OK** command button to create the table and close the dialog box.

Another method for defining a table **like** another table is to start from the definition of an existing table and create a new table as a copy. Use the **Object/Copy** function to transfer an existing table definition to the design mode and save it under a new name using the **Object/Save** function. Before saving, you can make any modifications that should be desired.

Defining a Table As a Query

Selecting the **Object/New/Table as Query** function activates design mode, in which you can define a query. DOMAIN first displays a table list from which you can select one or more source tables for the query.

For this type of table definition, follow the same procedure as for view definition (see the relevant section). Unlike view definition, however, the key columns, constraints, foreign keys, and indexes are **not** transferred from the base table. Instead, the contents of the base tables are transferred directly to the table defined as a **query**.

Modifying a Table

An existing table is modified in design mode. To do so, click on the table to be modified in the table list and select the **Object/Design** function or click on the toolbar button to display the definition of the selected table. You can now modify the cardfile cards containing the table definition using the usual Windows editing functions (**Edit** menu item). As when a table is created, terminate table modification by selecting the **Object/Save** function.

Naturally, certain restrictions apply when modifying a table; for more information, refer to the Reference online help.

- The **Date Type**, **Code**, **Length**, and **Decimals** attributes can be modified, as well as **Comment**, **Default**, and **Constraint**.

- Comments are defined and modified within the table list, as described above.

- You can rename columns by overwriting the old name with the new name.

- You can delete a column by first selecting it and then selecting the **Edit/Delete** function.

- You can delete a number of columns simultaneously by selecting them one after the other using the Shift key/mouse and then selecting the **Edit/Delete** function.

If you wish to modify the table structure with the intention of subsequently saving the table under another name, it is recommended that you first transfer the existing table to design mode using **Object/Copy** (for more information, refer to the section entitled 'Defining a Table Like Another Table (CREATE Table Like)').

Deleting Tables

To delete a table or tables, first select the table(s) to be deleted in the table list and then select the **Object/Delete** function. A dialog box appears in which you must confirm your selection.

If there is a ***View*** or ***Snapshot*** table based on the table to be deleted, you will receive a warning that the depending object will be deleted as well. In this case, the action must be confirmed (see ADABAS Reference manual DROP TABLE , CASCADE, RESTRICT).

Renaming Tables

To rename a table, select the table to be renamed in the table list and then select the **Object/Rename** function. A dialog box appears in which you must enter the new table name.

Managing Access Privileges for Tables

In the table list, select the table for which access privileges are to be granted and select the **Relships/Privileges** function. The privileges define which users are authorized to access particular columns of a table. Privileges describe the relationship between users and tables. Thus, the **Privileges** function is available in the object list under the **Relships** menu item. The **Relships/Privileges** function modifies the **User uses Table** and **User uses Column** relationships.

When you select the **Relships/Privileges** function, the list of all privileges granted for this table is displayed in tabular form. This list contains at least one entry since, otherwise, it would not be possible to access the table for which you would like to view or modify the privileges. If you yourself are the table owner or if you have the GRANT privilege for this table, you can now extend the list of privileges.

To define the access privileges for an additional user, enter the username, privilege type, and, if appropriate, the columns of the table, for each line. A number of privilege types can be entered as a list of privileges. The selection of table columns is also defined as a list of names with the same format.

If no column names are defined, the privilege applies to the entire table. For each user, you can enter as many lines as necessary for defining the various access privileges to the table columns.

To delete access privileges, delete the lines that define them by using **Edit/Delete** and exit the privilege window by selecting **Object/Save**.

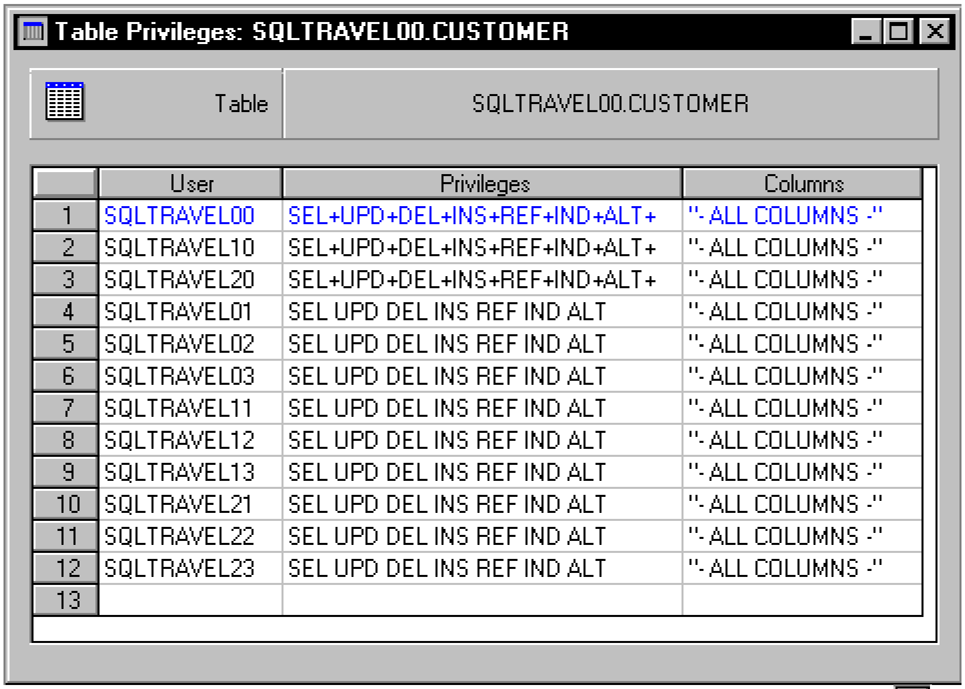


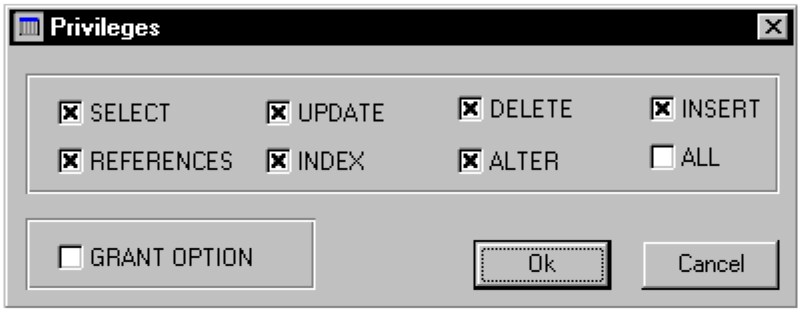
Table Privileges: SQLTRAVEL00.CUSTOMER

Table: SQLTRAVEL00.CUSTOMER

	User	Privileges	Columns
1	SQLTRAVEL00	SEL+UPD+DEL+INS+REF+IND+ALT+	"- ALL COLUMNS -"
2	SQLTRAVEL10	SEL+UPD+DEL+INS+REF+IND+ALT+	"- ALL COLUMNS -"
3	SQLTRAVEL20	SEL+UPD+DEL+INS+REF+IND+ALT+	"- ALL COLUMNS -"
4	SQLTRAVEL01	SEL UPD DEL INS REF IND ALT	"- ALL COLUMNS -"
5	SQLTRAVEL02	SEL UPD DEL INS REF IND ALT	"- ALL COLUMNS -"
6	SQLTRAVEL03	SEL UPD DEL INS REF IND ALT	"- ALL COLUMNS -"
7	SQLTRAVEL11	SEL UPD DEL INS REF IND ALT	"- ALL COLUMNS -"
8	SQLTRAVEL12	SEL UPD DEL INS REF IND ALT	"- ALL COLUMNS -"
9	SQLTRAVEL13	SEL UPD DEL INS REF IND ALT	"- ALL COLUMNS -"
10	SQLTRAVEL21	SEL UPD DEL INS REF IND ALT	"- ALL COLUMNS -"
11	SQLTRAVEL22	SEL UPD DEL INS REF IND ALT	"- ALL COLUMNS -"
12	SQLTRAVEL23	SEL UPD DEL INS REF IND ALT	"- ALL COLUMNS -"
13			

As soon as a box with the **User** heading is activated, the ▤ button appears. Click on it to obtain a list of all users and usergroups; you can then select the relevant users from this list. The keyword 'PUBLIC' means that the access privilege is granted to all users (present and future).

After clicking on the ▤ toolbar button, the privileges can also be selected from the privileges form by checking them (ALL, SELECT, INSERT, DELETE, UPDATE, SELUPD, REFERENCES, INDEX, ALTER). Only those privilege types actually available in the context are offered, e.g., the 'INS', 'IND', and 'ALT' privileges are available only if you yourself have these privileges.



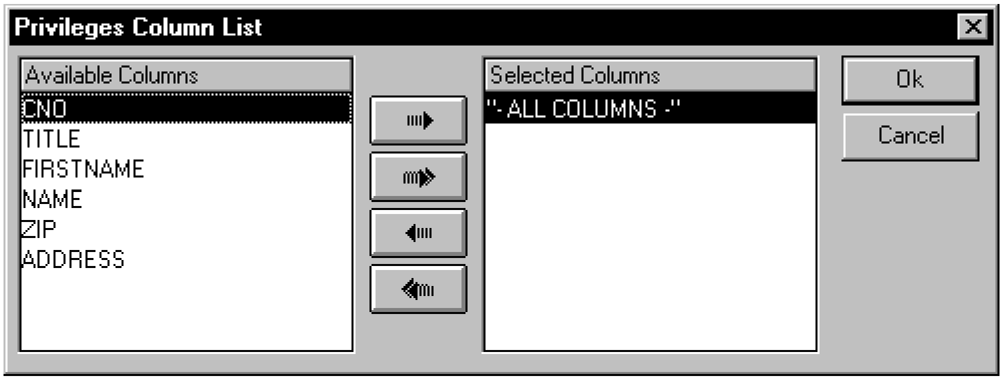
Privileges

☒ SELECT ☒ UPDATE ☒ DELETE ☒ INSERT

☒ REFERENCES ☒ INDEX ☒ ALTER □ ALL

□ GRANT OPTION

When you select the SELECT, UPDATE, or SELUPD privilege type, you can also define the columns to which the privilege is to apply. Click on the ▤ button to display a list of all the columns in the table under the **Column** heading. Select the desired columns using the arrow buttons; the sequence is irrelevant. The columns thus selected are listed in the tabular display of privileges on one line, separated by semicolons.

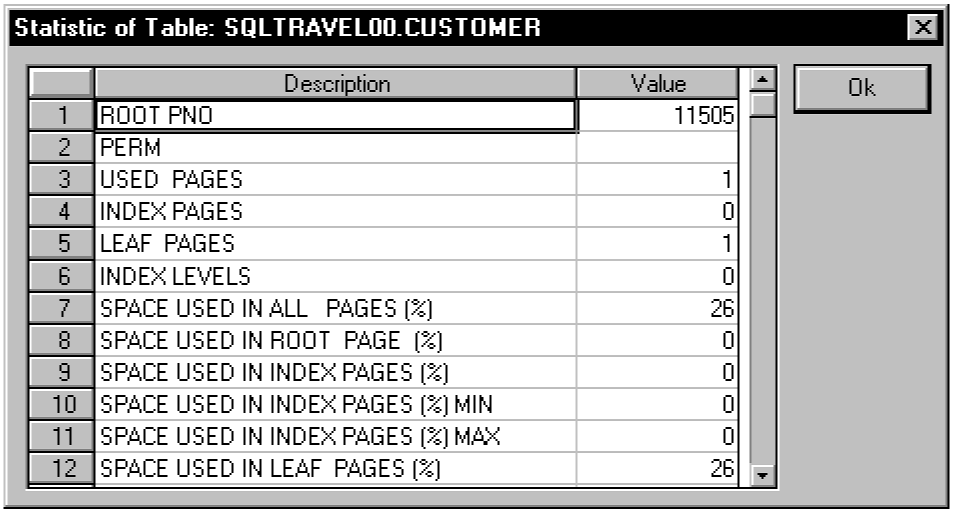


Privileges Column List

Available Columns	Selected Columns
CNO	"- ALL COLUMNS -"
TITLE	
FIRSTNAME	
NAME	
ZIP	
ADDRESS	

Statistical Information About Tables

Use the **Object/Statistic** menu function or the 123 toolbar button within the table list to display statistical information about the selected table.



	Description	Value
1	ROOT PNO	11505
2	PERM	
3	USED PAGES	1
4	INDEX PAGES	0
5	LEAF PAGES	1
6	INDEX LEVELS	0
7	SPACE USED IN ALL PAGES (%)	26
8	SPACE USED IN ROOT PAGE (%)	0
9	SPACE USED IN INDEX PAGES (%)	0
10	SPACE USED IN INDEX PAGES (%) MIN	0
11	SPACE USED IN INDEX PAGES (%) MAX	0
12	SPACE USED IN LEAF PAGES (%)	26

Column

The ***Column*** object identifies a column in a table. The names of the owner, table, and column are always required for its identification.

Relationships between a column and other objects:

1.	Column	Uses	Column
2.	Column	Refers	Domain
3.	Easycommand	Uses	Column
4.	Easyform	Uses	Column
5.	Foreignkey	Uses	Column
6.	Index	Uses	Column
7.	Module	Uses	Column
8.	Qpcommand	Uses	Column
9.	Qpquery	Uses	Column
10.	Querycommand	Uses	Column
11.	Snapshot	Contains	Column
12.	Table	Contains	Column
13.	User	Uses	Column
14.	View	Contains	Column

1. A (VIEW) column uses a table column; a table column can be used as a view column (0-1).

2. A column can be defined by a DOMAIN; a DOMAIN can be used for a number of column definitions (N[0]-1).

3. An EASY command uses one or more table columns; a column can be used by a number of EASY commands (N[0]-M).

4. An EASY form uses one or more table columns; a column can be used by a number of EASY forms (N[0]-M).

5. A foreign key contains one or more table columns; a table column can be part of a foreign key (0-N).

6. An index comprises one or more table columns; a table column can occur in an index definition (0-N).

7. A module can use a number of table columns; a table column can be used in a number of modules (N[0]-M[0]).

8. A QueryPlus command uses one or more table columns; a table column can be used by a number of QueryPlus commands (N[0]-M).

9. A QueryPlus query uses one or more table columns; a table column can be used by a number of QueryPlus queries (N[0]-M).

10. A QUERY command uses one or more table columns; a table column can be

used by a number of QUERY commands (N[0]-M).

11. A snapshot comprises one or more columns; a snapshot column is uniquely assigned to a snapshot (1-N).

12. A table comprises one or more columns; a column is uniquely assigned to a table (1-N).

13. A user can have access privileges for a number of table columns; a table column is used by one or more users (N-M[0]).

14. A view table comprises one or more columns; a view column is uniquely assigned to a view table (1-N).

A ***Column*** can be edited only within a table definition. However, it is also possible to list columns. Select design mode when a particular ***Column*** is selected to activate design mode for the associated table.

Constraint

A ***Constraint*** is a condition that limits the values of one or more columns in a table. A ***Constraint*** is defined either on the ***Constraints*** card within the table design or within a ***Domain***.

Relationship between a constraint and other objects:

1.	Table	Uses	Constraint

1. A table can have constraint definitions; a constraint is uniquely assigned to a table (1-N[0]). A domain constraint can be used in a number of tables (N[0]-M[0]).

It is possible to display a list of ***Constraints*** from the main menu. This can be helpful if you wish to have an overview of existing ***Constraints*** along with their names.

Select the **Object/Design** function from the constraint list or click on the 📝 toolbar button to activate design mode for the associated table. This opens the ***Constraint*** card, where you can define additional ***Constraints*** for the table as described in the section on 'Defining Constraints'.

Foreign Key

The ***Foreign Key*** object defines existence dependencies between the rows of two tables. A ***Foreign Key*** is created within the table design.

Relationships between a foreign key and other objects:

1.	Foreign Key	Uses	Column
2.	Foreign Key	Refers	Table

1. A foreign key contains one or more table columns; one table column can be part of a foreign key (0-N).

2 A foreign key refers to exactly one primary table; a table's primary key can be assigned to the foreign key of another table (0-1).

Generally speaking, ***Foreign Keys*** are considered within the context of the primary or foreign table, i.e., within the context of a table; however, it is also useful to be able to list all ***Foreign Key*** names as an overview (e.g., to check naming conventions).

Select the **Object/List All Objects** or **Object/List Own Objects** menu function to display the standard lists for a ***Foreign Key*** (see section on 'Object Lists'). When you select the **Object/Select Objects** function, a dialog box for entering search arguments is displayed. The owner and table name arguments describe the foreign table or the referencing table.

If the foreign key name was not explicitly specified during definition, ADABAS generates this name from the names of the primary table and foreign table so that you will be able to recognize which two tables are linked by this foreign key.

From the ***Foreign Key*** list, click on the ✎ toolbar button or select the **Object/Design** function to activate design mode for the associated foreign table. This opens the ***Foreign Keys*** card.

Click on the □ toolbar button (**Object/New** function) to display a dialog box for selecting a table list. When you select a table from the table list, design mode is activated for the table. This opens the ***Foreign Keys*** card.

The ***Foreign Key*** is defined as described in the section entitled 'Table'.

Triggers

A ***Trigger*** is a procedure executed by the database server as soon as a DML statement is applied to a base table or a view table derived from it. Thus, the three trigger types INSERT, UPDATE, and DELETE are available; an UPDATE trigger call can be restricted to specific columns.

Relationships between a trigger and other objects:

1. Table Contains Trigger2. Trigger Uses Module

1. A table can contain triggers; each trigger is assigned to exactly one table (1-N[0]).

2. A trigger definition refers to exactly one module; a module can be used for defining a number of triggers (N[0]-1).

DOMAIN provides only informational functions for the defined triggers.

From the main menu, you can access any trigger list and then click on the toolbar button or select the **Object/Design** function to select the trigger and change to design mode. In design mode, the trigger is displayed on a cardfile card with its parameters and constraints from the table definition.

As with all objects, you can define or query a comment for each trigger in the trigger list.

Index

The ***Index*** object describes one or more columns of a base table whose contents serve as the basis for the arrangement of the table rows. An ***Index*** can significantly increase the speed of table accesses.

Relationship between an index and other objects:

1. Index Uses Column

1. An index comprises one or more table columns; a table column can occur in an index definition (0-N).

From the index list, select the **Object/Design** function or click on the ✎ toolbar button to activate the ***Indexes*** card for the associated table. This card contains a detailed list of all indexes for the table.

From the index list, select the **Object/New** function or click on the □ toolbar button to change to a table list. Then click on the desired table to change to the ***Indexes*** card for the table.

The index editing window always contains all index definitions for the associated table. You can define additional indexes for this table below the defined indexes. For a detailed description of the index definition, see the sections 'Defining and Deleting an Index' and 'Table'.

See also

Statistical Information About an Index

Statistical Information About an Index

Use the **Object/Statistic** menu item or the 123 toolbar button within the ***Indexes*** list to display statistical information about the selected index.

Synonym

The ***Synonym*** object is an alternate name for a table. Synonyms can be defined for base or view tables. They are visible only to their owner.

Relationship between a synonym and other objects:

1.	Synonym	Refers	Snapshot
2.	Synonym	Refers	Table
3.	Synonym	Refers	View
4.	Easycommand	Uses	Synonym
5.	Easyform	Uses	Synonym
6.	Module	Uses	Synonym
7.	Qpcommand	Uses	Synonym
8.	Qpquery	Uses	Synonym
9.	Querycommand	Uses	Synonym
10.	Snapshot	Uses	Synonym
11.	View	Uses	Synonym

1. A synonym refers to a table; a number of synonym definitions can be assigned to a table (N[0]-1).

2. A synonym can refer to a view table; a number of synonym definitions can exist for a view table (N[0]-1[0]).

3. A synonym can refer to a snapshot; a number of synonym definitions can exist for a snapshot (N[0]-1[0]).

4. An EASY command can use a number of synonyms; a synonym can be used by a number of EASY commands (N[0]-M[0]).

5. An EASY form can use a number of synonyms; a synonym can be used by a number of EASY forms (N[0]-M[0]).

6. A module can use a number of synonyms; a synonym can be used by a number of modules (N[0]-M[0]).

7. A QueryPlus command can use a number of synonyms; a synonym can be used by a number of QueryPlus commands (N[0]-M[0]).

8. A QueryPlus query can be used by a number of synonyms; a synonym can be used by a number of QueryPlus queries (N[0]-M[0]).

9. A QUERY command can use a number of synonyms; a synonym can be used by a number of QUERY commands (N[0]-M[0]).

10. A snapshot can use a number of synonyms; a synonym can be used by a number of snapshots (N[0]-M[0]).

11. A view table can be used by a number of synonyms; a synonym can be used by

a number of view tables (N[0]-M[0]).

The ***Synonym*** object identifies a subset of the ***Table*** object**.** If you wish to obtain information on the table attributes of a synonym table, you can display the ***Columns*** associated with the synonym table in the list of tables by selecting the **Relships/ Object Usage/Columns** menu function (see section 'Table'). If you wish to create a synonym table or obtain a list of all synonyms, enter the ***Synonym*** list and select **New** or the corresponding toolbar button.

From the synonym list, select the **Object/Design** function or click on the toolbar button to activate the synonym definition which, however, contains only the name of the base table.

See also

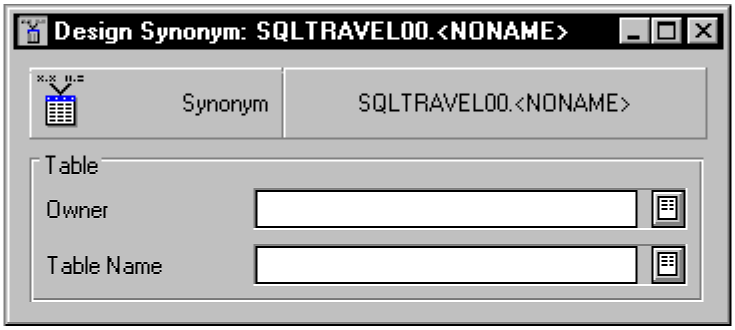
Defining a Synonym

Deleting a Synonym

Renaming a Synonym

Defining a Synonym

When you select the **Object/New** function or click on the □ toolbar button, the empty window for defining the synonym is displayed. You must specify the 'synonym name' and the table to which the synonym is to apply in the definition window. At this point, DOMAIN allows you to display a table list by clicking on the command button for the **Table Name** box. You can then transfer the name directly from the table list displayed to the synonym definition by clicking on the desired table name.
Example of a synonym definition:



Design Synonym: SQLTRAVEL00.<NONAME>

Synonym SQLTRAVEL00.<NONAME>

If the list of tables does not contain the desired table, you can request another list of tables by using the **Select** command button.

The synonym is created when you select the **Object/Save** function and specify the synonym name.

Deleting a Synonym

To delete a synonym definition from the synonym list or from the table list, select the **Object/Delete** function.

Renaming a Synonym

To rename a synonym, use the **Object/Rename** function to rename it either in the table list or in the synonym list.

View

The ***View*** object describes a view table. A view table describes the logical view of one or more tables that may themselves be view tables. A view table can be handled like any other table and, as such, has the same relationships to other objects as described for ***Table***. You can create, view, alter, and delete a view definition. The view definition determines the relationship between ***View*** and ***Table*** and between ***Column*** and ***Column***.

Relationship between a view and other objects:

1.	View	Contains	Column			
2.	View	Uses	Snapshot			
3.	View	Uses	Synonym			
4.	View	Uses	Table			
5.	View	Uses	View6.	Easycommand	Uses	View
7.	Easyform	Uses	View			
8.	Module	Uses	View			
9.	Qpcommand	Uses	View			
10.	Qpquery	Uses	View			
11.	Querycommand	Uses	View			
12.	Snapshot	Uses	View			
13.	Synonym	Refers	View			
14.	User	Uses	View			

1. A view table comprises one or more columns; a view column is uniquely assigned to a view table (1-N).

2. A view table can refer to one or more snapshots; a snapshot can be used by a view table (N[0]-M[0]).

3. A view table can refer to a number of synonyms; a synonym can be used by a view table (N[0]-M[0]).

4. A view table refers to one or more tables; a table can be used by a view table (N[0]-M).

5. A view table can refer to one or more view tables; a view table can be used by a view table (N[0]-M[0]).

6. An EASY command can use a number of view tables; a view table can be used by an EASY command (N[0]-M[0]).

7. An EASY form can be used by a number of view tables; a view table can be used in an EASY form (N[0]-M[0]).

8. A module can use a number of view tables; a view table can be used by a number of modules (N[0]-M[0]).

9. A QueryPlus command can use a number of view tables; a view table can be

used by a number of QueryPlus commands (N[0]-M[0]).

10. A QueryPlus query can use a number of view tables; a view table can be used by a number of QueryPlus queries (N[0]-M[0]).

11. A QUERY command can use a number of view tables; a view table can be used by a number of QUERY commands (N[0]-M[0]).

12. A snapshot can refer to a number of view tables; a view table can be used by a number of snapshots (N[0]-M[0]).

13. A synonym can refer to a view table; a number of synonym definitions can exist for a view table (N[0]-1[0]).

14. A user can use a number of view tables; a view table is used by one or more users (N-M[0]).

See also

Displaying a View

Creating a View

Deleting a View

Renaming a View

Managing Access Privileges for Views

Displaying a View

There are two options for displaying a view table:

- From the table list, as a list of columns by positioning on the desired view table and selecting the **Relships/Object Usage/Columns** function.

- From the view list, as a view definition by selecting the **Object/Design** function or clicking on the 📝 toolbar button. Views that were not created with DOMAIN are displayed as SQL statements.

Creating a View

In DOMAIN, a view is created with the same **drag-and-drop** mechanisms that are used in Microsoft Access.

Follow the steps below to define a view:

- Select one or more tables.
- Select the columns from the base tables.
- Name the view columns.
- Determine the sorting criterion.
- Define the selection criteria for the columns.

Terminate view definition using the **Object/Save** function. You can now specify the name of the view.

See also

Selecting One or More Tables

Selecting the Columns From the Base Tables

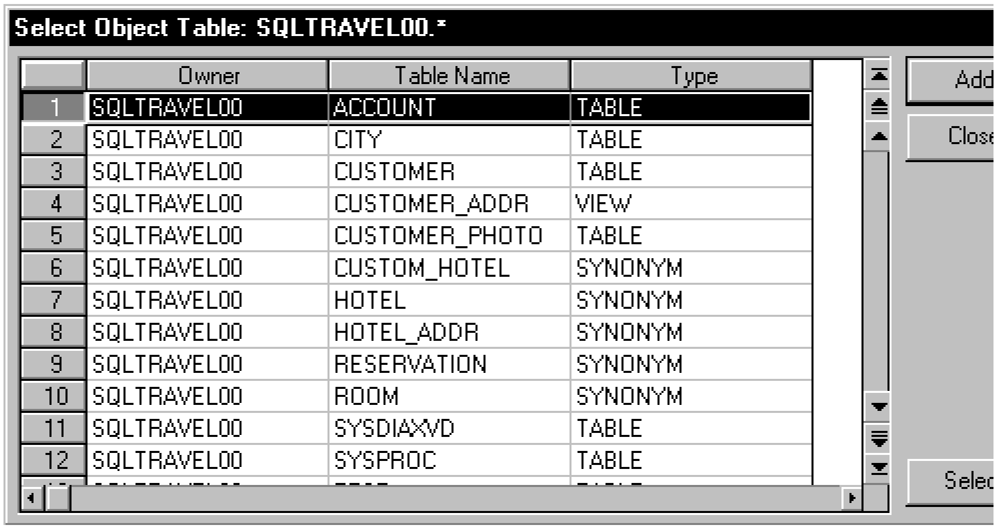
Naming the View Columns

Determining Sorting Criteria

Defining Selection Criteria for the Columns

Selecting One or More Tables

When you select the **Object/New** function or click on the 🗋 toolbar button in the main menu or from the view list, the empty window for defining the view table is displayed followed by the list of all own tables. From the table list, you must then select the tables on which the view is to be based.



	Owner	Table Name	Type
1	SQLTRAVEL00	ACCOUNT	TABLE
2	SQLTRAVEL00	CITY	TABLE
3	SQLTRAVEL00	CUSTOMER	TABLE
4	SQLTRAVEL00	CUSTOMER_ADDR	VIEW
5	SQLTRAVEL00	CUSTOMER_PHOTO	TABLE
6	SQLTRAVEL00	CUSTOM_HOTEL	SYNONYM
7	SQLTRAVEL00	HOTEL	SYNONYM
8	SQLTRAVEL00	HOTEL_ADDR	SYNONYM
9	SQLTRAVEL00	RESERVATION	SYNONYM
10	SQLTRAVEL00	ROOM	SYNONYM
11	SQLTRAVEL00	SYSDIAXVD	TABLE
12	SQLTRAVEL00	SYSPROC	TABLE

Transfer the tables to the view definition window by clicking on the **Add** command button. If the standard table list provided does not meet your requirements, you can use the **Select** command button to form another table list. If a view definition already exists, you can change the view definition by transferring additional tables from the table list. Click on the **Close** command button to terminate table selection.

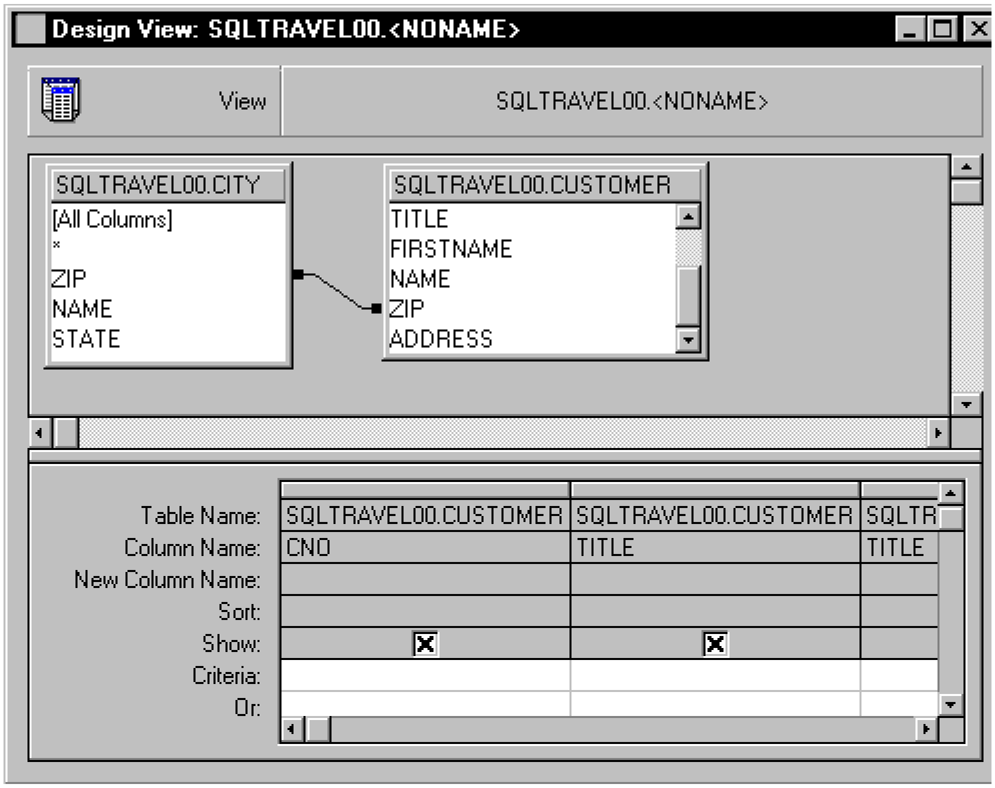


Table Name:	SQLTRAVEL00.CUSTOMER	SQLTRAVEL00.CUSTOMER	SQLTR
Column Name:	CNO	TITLE	TITLE
New Column Name:			
Sort:			
Show:	☒	☒	
Criteria:			
Or:			

From the view definition, the selection list of tables can be displayed at any time by using the **Edit/Add Table** function or the toolbar button.

Selecting the Columns From the Base Tables

The view definition window is divided into two sections. The top half contains the tables selected for the query. Here you can select the columns that are relevant for the view and join the tables to one another (see the section entitled 'The Defining Selection Criteria for the Columns'). The bottom half contains a more detailed description of the view in the form of a table. Here you can determine the sequence, the view column names, if any, the sorting sequence, and other conditions for the individual columns (see the sections entitled 'Naming the View Columns' and 'The Defining Selection Criteria for the Columns').

To select a view column in a table, click on the column in the top half of the query description window and drag the mouse pointer to the desired column of the view description in the lower half of the window by holding down the mouse button.

All columns transferred to the view description are defined as visible view columns. For example, if you wish to use a column only to formulate an AND condition and do not want the column itself to appear as a view column, you can set the column to **invisible.** To do so, click on the command button under **Show** for the relevant column; the check mark then disappears and the column is set to **invisible**. To facilitate the formulation of AND or OR operations, you can transfer one and the same column to the view description a number of times.

LONG Columns in Views

If the view definition is to contain a LONG column, the view definition must be based on a single base table.

Deleting a View Column

To completely remove a column from the view description, click on the column and select the **Edit/Delete** function.

Naming the View Columns

The view columns can be named independently of the base tables by entering the desired name on the **New Column Name** line.

Determining Sorting Criteria

Unless otherwise specified, the selected view columns are unsorted. Click on the **Sort** cell to change the sorting sequence for the relevant column to **Ascending** order; click on it once again to change to **Descending** order. Click on the cell a third time to return to the **unsorted (No Sort)** status.

Defining Selection Criteria for the Columns

You can use the cells as of the **Criteria** line to formulate selection criteria for a column. This restricts the view to certain rows of the base tables. A selection criterion can have any length. You can select the **Edit/Zoom** function or click on the 🔍 toolbar button to activate a larger editing area for easier entry.

Note the following when formulating a selection criterion: if the selection criterion comprises a single value, you can enter the value directly. If a predicate is used in formulating the selection criterion, the comparison values for the column must be set in accordance with its data type (i.e., the comparison value for non-numerical columns must be enclosed in single quotation marks).

You can use AND and OR operations to link several selection criteria. AND operations must always be formulated on *one* criterion line; OR operations must always be formulated on *several* criterion lines.

The AND Operation

The criteria for all columns formulated on one line are interpreted as AND operations. A column can be used for an AND operation without being visible. To simplify the formulation of AND operations, you can use the same column a number of times in the view description.

The OR Operation

Criteria located on different lines are interpreted as OR operations. A column can be used for an OR operation without being visible. To simplify the formulation of OR operations, you can use the same column a number of times in the view description.

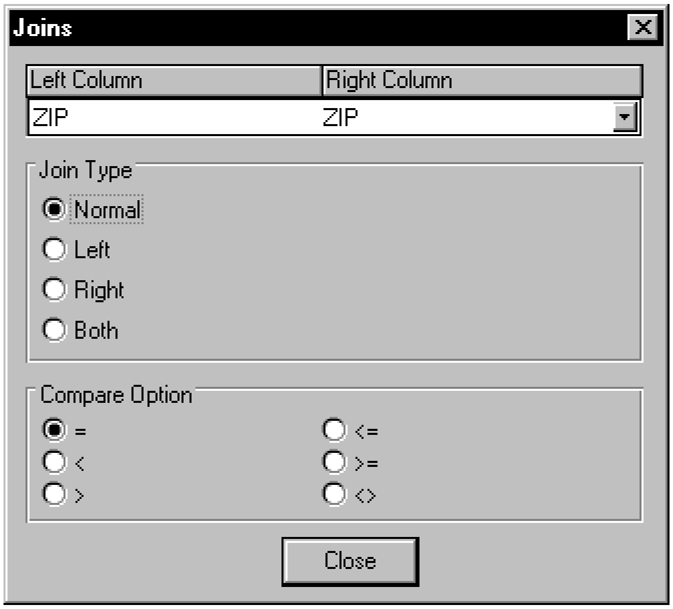
Using Predicates

You can use all the predicates described in the ADABAS Reference online help to formulate a selection criterion. These predicates are formulated as described in the section entitled 'Defining Selection Criteria for the Columns'. Subqueries can also be defined as a component of a predicate (see the ADABAS Reference online help). In this case, as well, you can use the **Edit/Zoom** function or the 🔍 toolbar button.

JOIN Predicate

The JOIN predicate is a special type of predicate that serves to link tables. To link two tables using a JOIN predicate, click on the column of the first table; holding down the mouse button, move the mouse pointer to the column of the second table and release the button. The link is displayed as a join line.

To specify a join operation more precisely, either select the **Options/Joins** function or double-click on the join line to display a list of all join definitions for this view. You can now define the join type and comparison operator for each join condition. To delete a join definition, click on the join line and select the **Edit/Delete** function.



Joins

Left Column	Right Column
ZIP	ZIP

Join Type

● Normal
○ Left
○ Right
○ Both

Compare Option

● = ○ <=
○ < ○ >=
○ > ○ <>

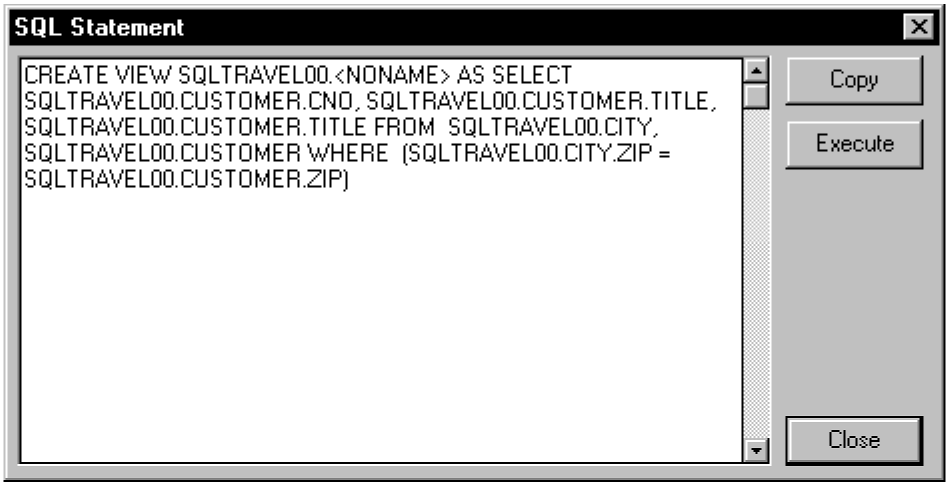
Options When Defining Views

Under the **Options/Object Options** menu item, you can set the **Check Option** for the defined view (see the ADABAS Reference online help).

View Definition as SQL Statement

If the qualifying condition is so complicated that it is difficult to define it using the means described here, it is possible to complete the definition on the SQL level. Select the **Options/SQL Statement** function to access an editing window with the SQL statement for the view definition appears. To complete the SQL statement, you can add the WHERE qualification to it. Then execute the SQL statement for defining the view. Before executing it, replace the token <NONAME> with the name of the view to be defined.

However, DOMAIN does not display the qualification thus defined.



```
CREATE VIEW SQLTRAVEL00.<NONAME> AS SELECT
SQLTRAVEL00.CUSTOMER.CNO, SQLTRAVEL00.CUSTOMER.TITLE,
SQLTRAVEL00.CUSTOMER.TITLE FROM  SQLTRAVEL00.CITY,
SQLTRAVEL00.CUSTOMER WHERE  (SQLTRAVEL00.CITY.ZIP =
SQLTRAVEL00.CUSTOMER.ZIP)
```

Deleting a View

To delete a view table, select it in the table list or view list and activate the **Object/Delete** function. You will then be asked to confirm the deletion.

Renaming a View

To rename a view table, select it in the table list or view list and activate the **Object/Rename** function. A dialog box then appears in which you must enter the new name.

Managing Access Privileges for Views

The management of access privileges for view tables is the same as for base tables (see section 'Managing Access Privileges for Tables').

Domain

The ***Domain*** object has a key position. It describes the value range for table columns. Domains permit data elements to be defined centrally and used in any tables, programs, and forms. Thus, you can introduce a list of data elements for an area of application.

Relationships between a domain and other objects:

1. Column Refers Domain2. Module Uses Domain3. User
Owns Domain

1. A column can be defined by a DOMAIN; a DOMAIN can be used for a number of column definitions (N[0]-1).
2. A module can be used for a number of DOMAIN definitions; a DOMAIN definition can be used in a number of modules (N[0]-M[0]).
3. A user can own several DOMAIN definitions; a DOMAIN definition is owned by exactly one user (1-M[0]).

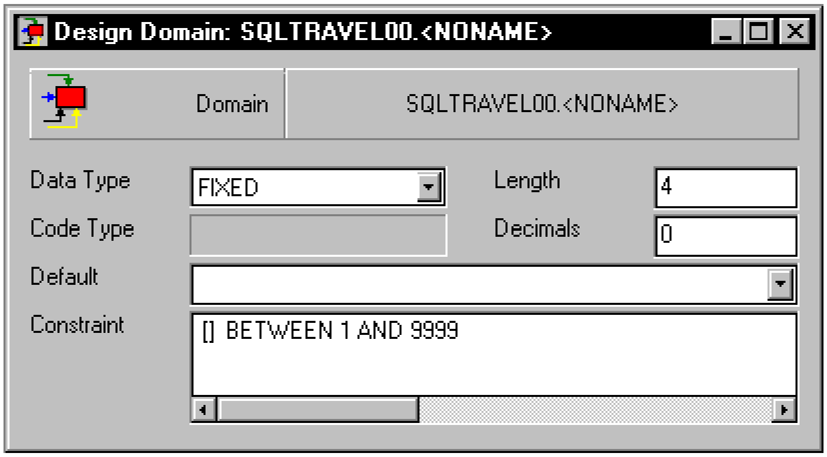
See also

Creating a Domain

Deleting a Domain

Creating a Domain

To create a domain, select the **Object/Design** function or click on the □ toolbar button from the main menu or from a domain list to change to design mode.



Design Domain: SQLTRAVEL00.<NONAME>

Domain SQLTRAVEL00.<NONAME>

Data Type	FIXED	Length	4
Code Type		Decimals	0
Default			
Constraint	[] BETWEEN 1 AND 9999		

The same attributes are used to describe a domain as were used for a column: **Code, Data Type, Length, Decimals,** and **Default.**

As with the table constraint, you can formulate a logical expression in the ***Constraint*** box. When a table is defined, the expression can contain column names from the table; when a domain is defined, however, a position indicator with the format **[ ]** (left bracket, right bracket) is used instead. The position indicator represents the column to which the domain definition is assigned. The constraint within the domain definition cannot refer to more than one column.

Create the domain using the **Object/Save** function while specifying the domain name.

You can also begin by transferring an existing domain definition to an empty definition window using **Object/Copy**, modify it as desired, and then save it under a different domain name using the **Object/Save** function.
You cannot modify domains subsequently; i.e., the **Object/Save** function is not available in design mode for an existing domain.

Deleting a Domain

To delete one or more domains, select them in the domain list and activate the **Object/Delete** function. You will then be asked to confirm the deletion.

User

The ***User*** object describes an ADABAS user or ADABAS usergroup. ***User*** occurs in relationships with a number of objects. The relationships are described by access privileges.

Relationships between a user and other objects:

1.	User	Owns	Dbfunction
2.	User	Owns	Domain
3.	User	Owns	User
4.	User	Uses	Column
5.	User	Uses	Dbprocedure
6.	User	Uses	Program
7.	User	Uses	Qpcommand
8.	User	Uses	Qpexcellink
9.	User	Uses	Qpquery
10.	User	Uses	Qpwordlink
11.	User	Uses	Querycommand
12.	User	Uses	Snapshot
13.	User	Uses	Table
14.	User	Uses	View

1. A user (DBA) can be the owner of a number of DB functions; a user has a unique owner (1-M[0]).

2. A user (DBA) can be the owner of a number of domains; a domain has a unique owner (1-M[0]).

3. A user (DBA) can be the owner of a number of users; a user has a unique owner (1-M[0]).

4. A user can have access privileges for a number of table columns; a table column is used by one or more users (N-M[0]).

5. A user can have execute privileges for a number of DB procedures; a DB procedure is used by one or more users (N-M[0]).

6. A user can have access privileges for a number of programs (for execution or copying); a program is used by one or more users (N-M[0]).

7. A user can have call privileges for a number of QueryPlus commands; a QueryPlus command is used by one or more users (N-M[0]).

8. A user can have usage privileges for a number of QueryPlus ExcelLinks; a QueryPlus ExcelLink is used by one ore more users (N-M[0]).

9. A user can have call privileges for a number of QueryPlus queries; a QueryPlus query is used by one or more users (N-M[0]).

10. A user can have usage privileges for a number of QueryPlus WordLinks; a QueryPlus WordLink is used by one or more users (N-M[0]).

11. A user can have call privileges for a number of QUERY commands; a QUERY command is used by one or more users (N-M[0]).

12. A user can use a number of snapshots; a snapshot can be used by one or more users (N-M[0]).

13. A user can use a number of tables; a table can be used by one or more users (N-M[0]).

14. A user can use a number of view tables; a view table can be used by one or more users (N-M[0]).

See also

User List

Defining a User

Defining a Usergroup

Defining Users in a Group

Defining a User Like Another User

Changing a User Definition

Deleting a User

Changing Owners of Users And Usergroups

Access Privileges for Tables

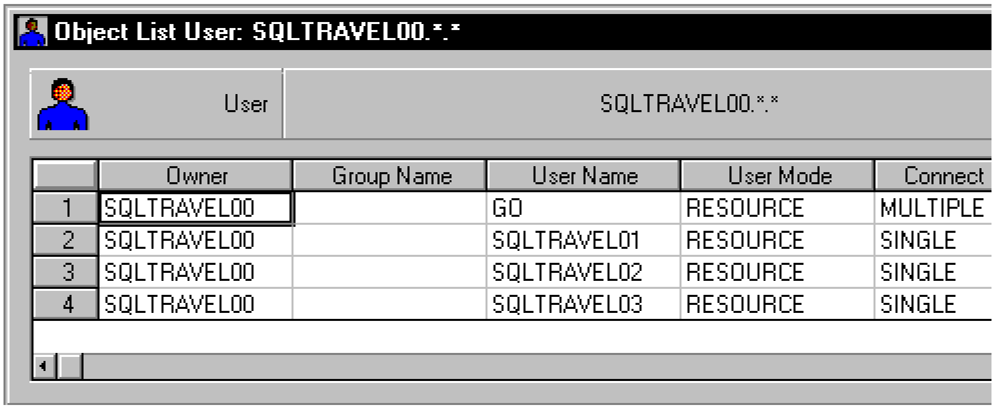
Execute Privileges for DB Procedures

Statistical Information About the Object User

User List

As with all other objects, you can access the user list by selecting the **Object/List All Objects**, **Object/List Own Objects** or **Object/Select Objects** function or by clicking on the ,
, or
toolbar button from the main menu. A list of all users, usergroups, and members is then displayed.

The example shows a list of all users that have been created by the user SQLTRAVEL00 (there are no usergroups).



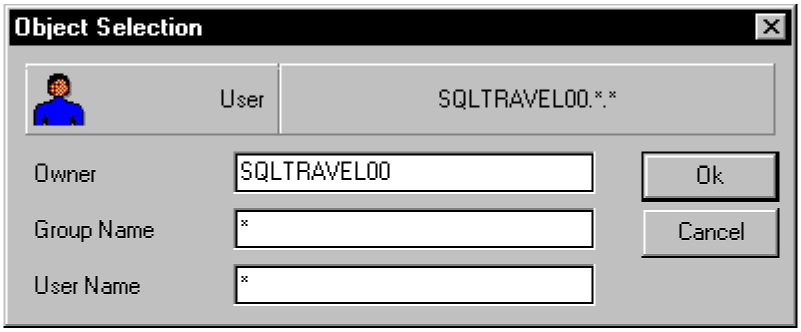
Object List User: SQLTRAVEL00.*.*

User SQLTRAVEL00.*.*

	Owner	Group Name	User Name	User Mode	Connect
1	SQLTRAVEL00		GO	RESOURCE	MULTIPLE
2	SQLTRAVEL00		SQLTRAVEL01	RESOURCE	SINGLE
3	SQLTRAVEL00		SQLTRAVEL02	RESOURCE	SINGLE
4	SQLTRAVEL00		SQLTRAVEL03	RESOURCE	SINGLE

In the case of single users, only the **User Name** box is used; in the case of usergroups, only the **Group Name** box is used. **User Name** and **Group Name** are both used in the case of members of usergroups. The meaning of the three-part name at the top of the window is **Owner.Group Name.User Name**.

When you use the **Object/Select Objects** function or click on the toolbar button, a dialog box is displayed for selecting the user list directly.



User SQLTRAVEL00.*.*

Owner: SQLTRAVEL00

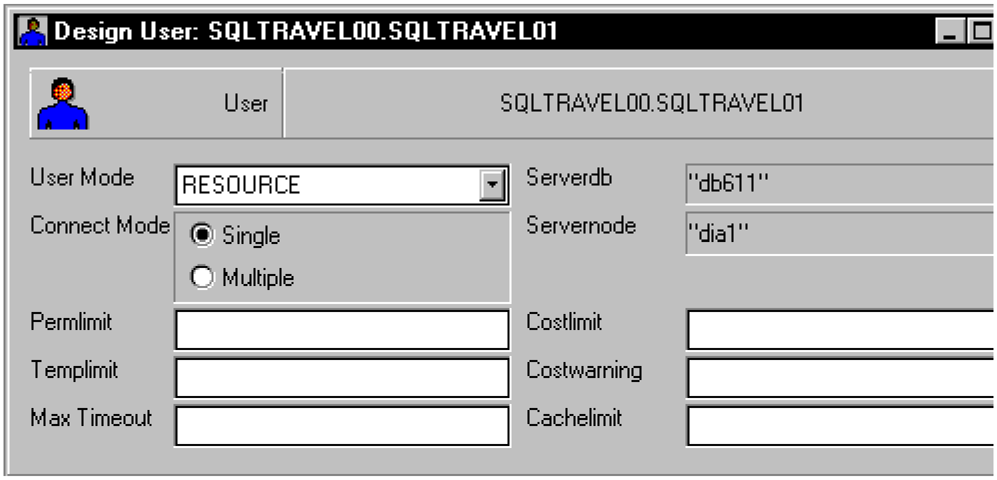
Group Name: *

User Name: *

You can specify the three names in the dialog box as search strings. For example, you can obtain a list of all single users by clearing the **Group Name** box or a list of usergroups without their members by clearing the **User Name** box. You can also obtain a list of all users, usergroups, and user-group members by entering an asterisk ('*') in the **Owner** box.

Defining a User

When you select the **Object/New/User** function from the user list, the definition window is displayed; the **Usermode** box is predefined and **Connect Mode='Single'** is selected.



Design User: SQLTRAVEL00.SQLTRAVEL01

User SQLTRAVEL00.SQLTRAVEL01

User Mode: RESOURCE

Connect Mode: ● Single ○ Multiple

Serverdb: "db611"

Servernode: "dia1"

Permlimit:

Templimit:

Max Timeout:

Costlimit:

Costwarning:

Cachelimit:

You can now define the attributes (**Usermode**, **Permlimit**, **Templimit**, **Costwarning**, **Costlimit,** and **Timeout**) for the new user. A combo box is used to make an entry in the **Usermode** box. Only those user modes that are currently available are offered.

As a rule, you will select only the **Usermode**; the other attributes are usually defined only in exceptional cases. For a description of the meaning of the **Permlimit**, **Templimit**, **Costwarning**, **Costlimit,** and **Timeout** attributes, refer to the ADABAS Reference online help.

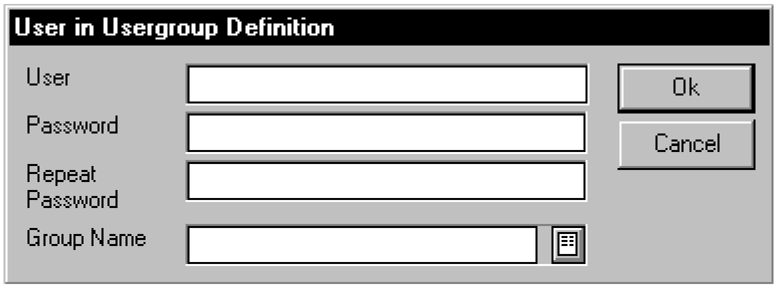
To create the user, select the **Object/Save** function and enter the user name and password in the superimposed dialog box. As a precaution, you must enter the password twice.

Defining a Usergroup

A ***Usergroup*** is defined in the same way as a user. When you select the **Object/New/Usergroup** function, the definition window is displayed. The window for defining usergroups is the same as that for users except that when you select the **Object/Save** function, a dialog box appears in which you must enter the name of the usergroup only. The members of the usergroup are defined separately using the **Object/New/User in Group** function.

Defining Users in a Group

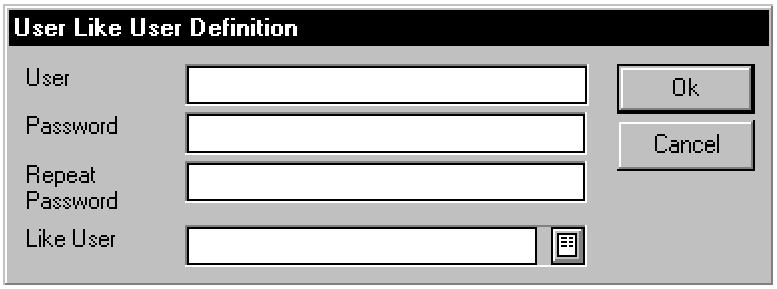
When you select the **Object/New/User in Group** function, a dialog box is displayed in which you must enter the username, password, and usergroup. All other user attributes have already been defined for the usergroup.



When you click on the ▤ button, DOMAIN helps you to fill in the **Usergroup** box by displaying a list of all usergroups. Select the desired group to transfer it from the list.

Defining a User Like Another User

When you select the **Object/New/User Like** function, a dialog box is displayed in which you must specify the username, password, and **Like User**.



DOMAIN provides you with a list of users when you click on the ▤ button. The user list contains all the user names that you are authorized to view. It can contain both single users and members of groups.

Changing a User Definition

In design mode for an existing user definition, you can subsequently perform the following actions:

- Change the **Usermode**.

- Change the **Connectmode** from **Single** to **Multiple** or vice versa.

- Change the password using the **Alter/Password** function. A dialog box is then displayed in which you must enter the new password twice.

- Change the settings for **Permlimit**, **Templimit**, **Costwarning**, **Costlimit**, and **Timeout**.

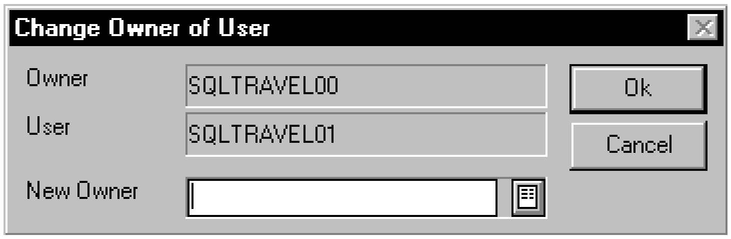
Deleting a User

To delete a user, click on the user entry in the user list and activate the **Object/Delete** function. Deleting a user also deletes all objects owned by this user. In addition, all existing relationships between this user and other objects are deleted; consequently, it may take some time to complete this function.

Changing Owners of Users And Usergroups

The owner of a user or usergroup can be changed either by DBAs who are owners of user definitions or by the system DBA using the **Object/Change Owner** function. This function is available in design mode.

In the resulting window, you can change the owner of the current user.



Change Owner of User

Owner SQLTRAVEL00

User SQLTRAVEL01

New Owner

The system DBA is authorized to transfer not only his own users but also those of other DBAs to another user (also a DBA). DBAs can transfer only their own users to another DBA; for this reason, the **Owner** box for a DBA is preset to his or her own name and cannot be changed.

Access Privileges for Tables

The access privileges are defined when the ***Table*** object is defined.

From the user list or from the user/usergroup definition window, you can determine a user's or usergroup's access privileges for tables using the **Relships/Uses Relships** function and the **User Uses Table** and **User Uses Column** relationships.

Execute Privileges for DB Procedures

Execute privileges are defined for the ***DB Procedure*** object.

From the user list or from the user/usergroup definition window, you can determine a user's or usergroup's execute privileges for DB procedures using the **Relships/Uses Relships** function and the **User Uses DB Procedure** relationships.

Statistical Information About the Object User

Statistical information about the selected user is displayed by selecting the **Object/Statistic** menu item or by clicking on the 123 button when you are in the ***User*** list.

DB Procedure

A ***DB Procedure*** is a special procedure executed in the database server. Like SQL statements, DB procedures can be called from any user process. They serve to combine a number of SQL statements. DB procedures are used for the following reasons:

- To improve performance, since a ***DB Procedure*** is processed by the database server, thus reducing communication overhead.

- To simplify programming, since complex sequences of SQL statements can be replaced by a single ***DB Procedure*** call.

- To simplify granting privileges, since it is not necessary to grant privileges for the database objects accessed beyond the call privilege for a ***DB Procedure***.

Relationships between a DB procedure and other objects:

1.	Dbprocedure Uses	Refers Dbprocedure	Module2.	Module	Calls	Dbprocedure3.	User

1. A DB procedure is implemented by a specific module; a module can be used as a DB procedure (0-1).

2. A module can call a number of DB procedures; a DB procedure can be called by a number of modules (N[0]-M[0]).

3. A user can have execute privileges for a number of DB procedures; a DB procedure can be used by one or more users (N-M[0]).

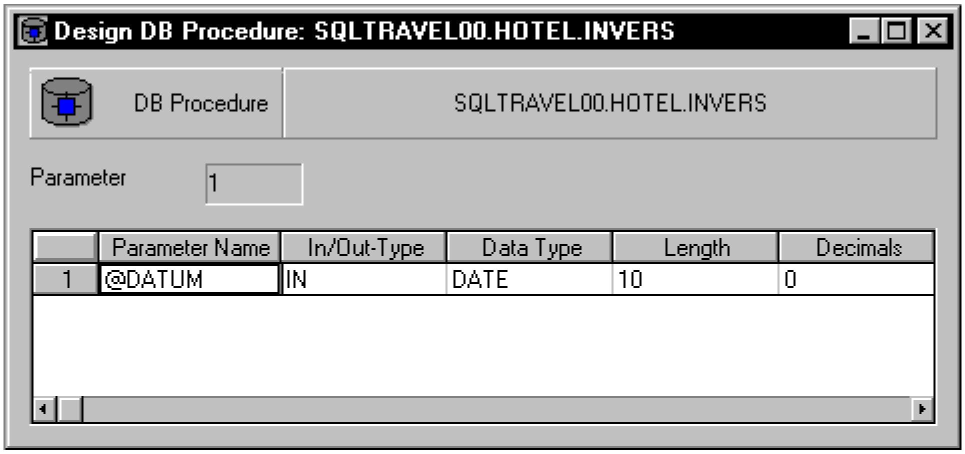
See also

Displaying a DB Procedure

Managing Execute Privileges for DB Procedures

Displaying a DB Procedure

To display a ***DB Procedure***, single-click on the desired procedure in a list of DB procedures and select the **Object/Design** function or click on the toolbar button. The list of all parameters is then displayed.



Design DB Procedure: SQLTRAVEL00.HOTEL.INVERS

DB Procedure: SQLTRAVEL00.HOTEL.INVERS

Parameter: 1

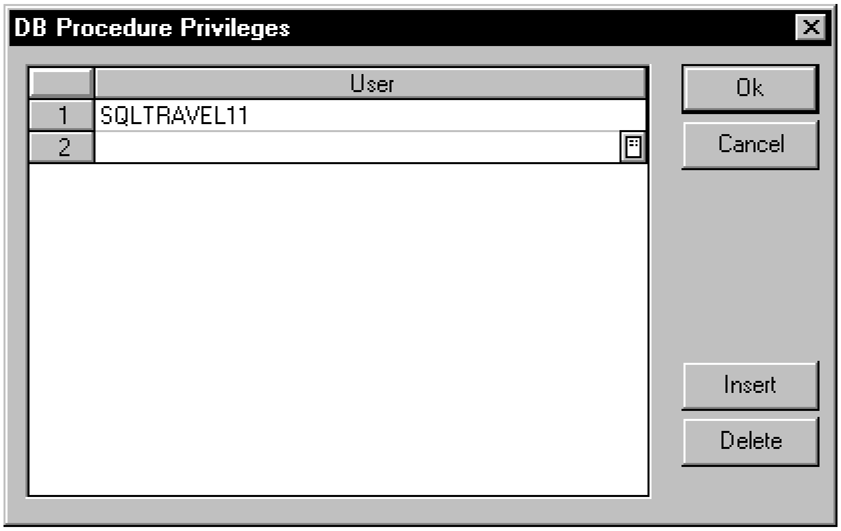
	Parameter Name	In/Out-Type	Data Type	Length	Decimals
1	@DATUM	IN	DATE	10	0

Managing Execute Privileges for DB Procedures

To activate privilege editing for the DB procedure, select the **Relships/Privileges** function from the ***DB Procedure*** list or the ***DB Procedure*** window. The privileges determine which users have execute privileges for the DB procedure and describe the relationship between users and the DB procedures.

A list of all users who have execute privileges for this ***DB Procedure*** is displayed. If you are the owner of the procedure, you can add additional users to the list.

DOMAIN allows you to select users from a list of existing users (▤ toolbar button).



	User
1	SQLTRAVEL11
2	▤

To cancel execute privileges, delete the line containing the privilege definition (**Delete**) and terminate the definition using **Ok**.

DB Function

A ***DB Function*** is a user-specific function that can be used in SELECT statements on the selected columns or in the WHERE condition. It is executed in the database server.

Relationships between a DB Function and other objects:

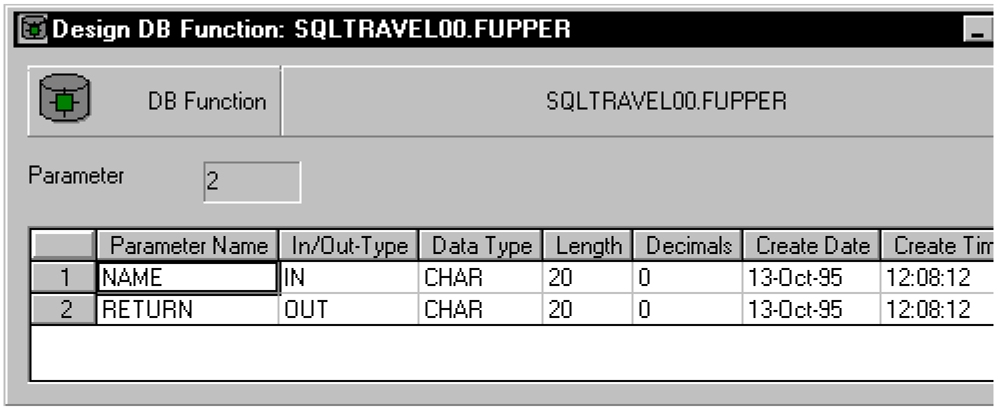
1. Dbfunction Refers Module2. User Owns Dbfunction

1. A DB Function is implemented by a specific module; a module can be used as a DB Function (0-1).

2. A user can be the owner of a number of DB Functions; a DB Function is owned by exactly one owner (1-M[0]).

DB Functions can be listed like ***DB Procedures***. In design mode, all parameter definitions of the ***DB Function*** are displayed. The RETURN parameter identifies the type of the result value produced by the ***DB Function***.

DB Functions can only be deleted by their owners and can be used by all other users.



Design DB Function: SQLTRAVEL00.FUPPER

DB Function SQLTRAVEL00.FUPPER

Parameter 2

	Parameter Name	In/Out-Type	Data Type	Length	Decimals	Create Date	Create Tir
1	NAME	IN	CHAR	20	0	13-Oct-95	12:08:12
2	RETURN	OUT	CHAR	20	0	13-Oct-95	12:08:12

Snapshot

The ***Snapshot*** object is defined in the same way as the view table. The data of a ***Snapshot*** exists in the form of a (read-only) copy that is typically located on a different computer from the base tables. Basically, ***Snapshots*** are used decentrally in order to access partial datasets from productive applications at defined times. ***Snapshots*** are updated explicitly using the REFRESH statement (see the ADABAS Reference online help).

Relationship between a snapshot and other objects:

1.	Snapshot	Contains	Column
2.	Snapshot	Uses	Synonym
3.	Snapshot	Uses	Table
4.	Snapshot	Uses	View
5.	Easycommand	Uses	Snapshot
6.	Easyform	Uses	Snapshot
7.	Module	Uses	Snapshot
8.	Qpcommand	Uses	Snapshot
9.	Qpquery	Uses	Snapshot
10.	Querycommand	Uses	Snapshot
11.	Synonym	Refers	Snapshot
12.	User	Uses	Snapshot
13.	View	Uses	Snapshot

1. A snapshot comprises one or more columns; a snapshot column is uniquely assigned to a snapshot (1-N).

2. A snapshot can refer to a number of synonyms; a synonym can be used by a snapshot (N[0]-M[0]).

3. A snapshot refers to one or more tables; a table can be used by a snapshot (N[0]-M).

4. A snapshot can refer to a number of view tables; a view table can be used a snapshot (N[0]-M[0]).

5. An EASY command can use a number of snapshots; a snapshot can be used by an EASY command (N[0]-M[0]).

6. An EASY form can use a number of view tables; a view table can be used in an EASY form (N[0]-M[0]).

7. A module can use a number of snapshots; a snapshot can be used by a number of modules (N[0]-M[0]).

8. A QueryPlus command can use a number of snapshots; a snapshot can be used by a number of QueryPlus commands (N[0]-M[0]).

9. A QueryPlus query can use a number of snapshots; a snapshot can be used by

a number of QueryPlus queries (N[0]-M[0]).

10. A QUERY command can use a number of snapshots; a snapshot can be used by a number of QUERY commands (N[0]-M[0]).

11. A synonym can refer to a snapshot; a number of synonym definitions can exist for a snapshot (N[0]-1[0]).

12. A user can use a number of snapshots; a snapshot is used by one or more users (N-M[0]).

13. A snapshot refers to one or more tables; a table can be used by a snapshot (N[0]-M).

A ***Snapshot*** can be handled like any other table and, as such, has the same structure and relationships to other objects as described for ***Table*** (see section 'Table'). The procedure for creating, viewing (**Relships/Object Usage/Colunns** function**)**, and deleting a ***Snapshot*** definition (see the section entitled 'View') is the same as for a view definition. The ***Snapshot*** definition determines the relationship between ***Snapshot*** and ***Table*** and between ***Column*** and ***Column***.

Module

A ***Module*** is a compilation unit and is therefore written in a specific programming language. It can, for example, be a C, a COBOL, or an ADABASIC module.

A ***Module*** uses reports, forms, and tables and can call other modules.

Relationships between modules and other objects:

1.	Module	Calls	Dbprocedure
2.	Module	Calls	Module
3.	Module	Uses	Column
4.	Module	Uses	Domain
5.	Module	Uses	Querycommand
6.	Module	Uses	Snapshot
7.	Module	Uses	Synonym
8.	Module	Uses	Table
9.	Module	Uses	View
10.	Dbfunction	Refers	Module
11.	Dbprocedure	Refers	Module
12.	Program	Contains	Module
13.	Trigger	Refers	Module

1. A module can call a number of DB procedures; a DB procedure can be called by a number of modules (N[0]-M[0]).

2. A module can call additional modules; a module can be called by a number of modules (N[0]-M[0]).

3. A module can use a number of table columns; a table column can be used in a number of modules (N[0]-M[0]).

4. A module can use a number of DOMAIN definitions; a DOMAIN definition can be used in a number of modules (N[0]-M[0]).

5. A module can call a number of QUERY commands; a QUERY command can be called by a number of modules (N[0]-M[0]).

6. A module can use a number of snapshots; a snapshot can be used by a number of modules (N[0]-M[0]).

7. A module can use a number of synonyms; a synonym can be used by a number of modules (N[0]-M[0]).

8. A module can use a number of tables; a table can be used by a number of modules (N[0]-M[0]).

9. A module can use a number of view tables; a view table can be used by a number of modules (N[0]-M[0]).

10. A DB function is implemented by a specific module; a modulce can be used as a DB function (0-1).

11. A DB procedure is implemented by a specific module; a module can be used as a DB procedure (0-1).

12 A program contains one or more modules; a module is uniquely assigned to a program (1-N).

13. A trigger is implemented by a specific module; a module can be used as a trigger (0-1).

The ***Program***, ***Module***, ***QUERY Command***, ***EASY Command,*** and ***EASY Form*** objects are all basically handled in the same way. Since these objects are created outside DOMAIN, their functionality within DOMAIN is mainly limited to displaying relationships to other objects.

See also

Deleting Modules

Deleting Modules

To delete one or more modules from the module list, select the relevant module(s) and activate the **Object/Delete** function. You are then asked to confirm the deletion in a dialog box.

Program

The ***Program*** object designates a unit to which the ***Module*** object is subordinate. A program comprises a number of modules. Users can have various access privileges for programs. A program is uniquely identified by its owner and its program name.

Relationships between a program and other objects:

1. Program Contains Module2. User Uses Program

1. A program contains one or more modules; a module is uniquely assigned to a program (1-N).

2. A user can have privileges for calling or copying one or more programs. Each program has a unique owner (1-N[0]).

Any program and relationship lists can be generated for the ***Program*** object. A program cannot be deleted explicitly from DOMAIN.

QUERY Command

The ***QUERY Command*** object designates the command stored in *QUERY*. The names of the owner and command are always required for its identification.

Relationships between a QUERY Command and other objects:

1.	Querycommand	Uses	Column
2.	Querycommand	Uses	Snapshot
3.	Querycommand	Uses	Synonym
4.	Querycommand	Uses	Table
5.	Querycommand	Uses	View
6.	Module	Uses	Querycommand
7.	User	Uses	Querycommand

1. A QUERY command uses one or more table columns; a table column can be used by a number of QUERY commands (N[0]-M).

2. A QUERY command can use a number of snapshots; a snapshot can be used by a number of QUERY commands (N[0]-M).

3. A QUERY command can use a number of synonyms; a synonym can be used by a number of QUERY commands (N[0]-M).

4. A QUERY command uses one or more tables; a table can be used by a number of QUERY commands (N[0]-M).

5. A QUERY command can use a number of view tables; a view table can be used by a number of QUERY commands (N[0]-M).

6. A module can call a number of QUERY commands; a QUERY command can be called by a number of modules (N[0]-M[0]).

7. A user can call a number of QUERY commands; a QUERY command is called by a number of users (N-M[0]).

Any command and relationship lists can be generated for the ***QUERY Command*** object. ***QUERY Commands*** can be deleted or modified only with the aid of the end user tool *QUERY*.

EASY Command

The ***EASY Command*** object designates the named request in *EASY*. The names of the owner, command, table owner, and table are always required for its identification.

Relationships between an EASY Command and other objects:

1.	Easycommand	Uses	Column
2.	Easycommand	Uses	Snapshot
3.	Easycommand	Uses	Synonym
4.	Easycommand	Uses	Table
5.	Easycommand	Uses	View

1. An EASY command uses one or more table columns; a column can be used by a number of EASY commands (N[0]-M).

2. An EASY command can use a number of snapshots; a snapshot can be used by a number of EASY commands (N[0]-M).

3. An EASY command can use a number of synonyms; a synonym can be used by a number of EASY commands (N[0]-M).

4. An EASY command uses one or more tables; a table can be used by an EASY command (N[0]-M).

5. An EASY command can use a number of view tables; a view table can be used by a number of EASY commands (N[0]-M).

Any command and relationship lists can be generated for the ***EASY Command*** object. ***EASY Commands*** can be deleted or modified only with the aid of the end user tool *EASY*.

EASY Form

The ***EASY Form*** object designates your own request form in *EASY*. The names of the owner, table, form owner, and form are always required for its identification.

Relationships between an EASY Form and other objects:

1.	Easyform	Uses	Column
2.	Easyform	Uses	Snapshot
3.	Easyform	Uses	Synonym
4.	Easyform	Uses	Table
5.	Easyform	Uses	View

1. An EASY form uses one or more table columns; a column can be used by a number of EASY forms (N[0]-M).

2. An EASY form can use a number of snapshots; a snapshot can be used by a number of EASY forms (N[0]-M).

3. An EASY form can use a number of synonyms; a synonym can be used by a number of EASY forms (N[0]-M).

4. An EASY form uses one or more tables; a table can be used by one EASY form (N[0]-M).

5. An EASY form can be used by a number of view tables; a view table can be used by a number of EASY forms (N[0]-M).

Any object and relationship lists can be generated for the ***EASY Form*** object. ***EASY Forms*** can be deleted or modified only with the aid of the end user tool *EASY*.

QueryPlus Command

A ***QueryPlus Command*** object is a command created with the aid of the Windows tool *QueryPlus*. In *QueryPlus*, as in the case of *QUERY*, any SQL statement can be stored as a command and made available to other users for calling. A ***QueryPlus Command*** is identified by its owner and name (see the section entitled 'QueryPlus' in the OfficePlus online help).

Relationships between a QueryPlus Command and other objects:

1.	Qpcommand	Uses	Column
2.	Qpcommand	Uses	Snapshot
3.	Qpcommand	Uses	Synonym
4.	Qpcommand	Uses	Table
5.	Qpcommand	Uses	View
6.	Qpexcellink	Uses	Qpcommand
7.	Qpwordlink	Uses	Qpcommand
8.	User	Uses	Qpcommand

1. A QueryPlus command uses one or more table columns; a table column can be used by a number of ***QueryPlus commands*** (N[0]-M).

2. A QueryPlus command can use a number of snapshots; a snapshot can be used by a number of QueryPlus commands (N[0]-M[0]).

3. A QueryPlus command can use a number of synonyms; a synonym can be used by a number of QueryPlus commands (N[0]-M[0]).

4. A QueryPlus command uses one or more tables; a table can be used by a number of QueryPlus commands (N[0]-M).

5. A QueryPlus command can use a number of view tables; a view table can be used by a number of QueryPlus commands (N[0]-M[0]).

6. An ExcelLink can link one or more QueryPlus command objects to an Excel object; a QueryPlus command can be used by one or more Excel objects by using an ExcelLink of the same name (N[0]-M[0]).
7. A WordLink can link one or more QueryPlus command objects to a WinWord document; a QueryPlus command can be used by one or more WinWord documents by using a WordLink object of the same name (N[0]-M[0]).

8. A user can call a number of QueryPlus commands; a *QueryPlus command* is called by a number of users (N-M[0]).

Any object and relationship lists can be generated for the ***QueryPlus Command*** object. ***QueryPlus Commands*** can be deleted or modified only with the aid of *QueryPlus*.

QueryPlus Query

A ***QueryPlus Query*** object is a query created interactively with the aid of the Windows tool *QueryPlus*. In *QueryPlus*, as in the case of *QUERY*, any SQL statement can be saved as a command and made available to other users for calling. A ***QueryPlus Query*** is identified by its owner and name (see the section entitled 'QueryPlus' in the OfficePlus online help).

Relationships between a QueryPlus Query and other objects:

1.	Qpquery	Uses	Column
2.	Qpquery	Uses	Snapshot
3.	Qpquery	Uses	Synonym
4.	Qpquery	Uses	Table
5.	Qpquery	Uses	View
6.	Qpexcellink	Uses	Qpquery
7.	Qpwordlink	Uses	Qpquery
8.	User	Uses	Qpquery

1. A QueryPlus query uses one or more table columns; a table column can be used by a number of ***QueryPlus Queries*** (N[0]-M).

2. A QueryPlus query can use a number of snapshots; a snapshot can be used by a number of QueryPlus queries (N[0]-M[0]).

3. A QueryPlus query can use a number of synonyms; a synonym can be used by a number of QueryPlus queries (N[0]-M[0]).

4. A QueryPlus query uses one or more tables; a table can be used by a number of QueryPlus queries (N[0]-M).

5. A QueryPlus query can use a number of view tables; a view table can be used by a number of QueryPlusqQueries (N[0]-M[0]).

6. An ExcelLink can link one or more QueryPlus query objects to one Excel object; a QueryPlus query can be used by one or more Excel objects by using an ExcelLink of the same name (N[0]-M[0]).
7. A WordLink can link one or more QueryPlus query objects to a WinWord document; a QueryPlus query can be used by one or more WinWord documents by using a WordLink object of the same name (N[0]-M[0]).

8. A user can call a number of QueryPlus queries; a QueryPlus query is called by a number of users (N-M[0]).

Any object and relationship lists can be generated for the ***QueryPlus Query*** object. ***QueryPlus Query*** objects can be deleted or modified only with the aid of *QueryPlus*.

WordLink

A ***WordLink*** object is a link between a WinWord document and one (or more) ***QueryPlus Command*** or ***QueryPlus Query*** object(s) that is (are) created with the aid of the Windows tool *QueryPlus*. A ***WordLink*** object is identified by the name of the Word document (see the section entitled 'QueryPlus' in the OfficePlus online help).

Relationships between a WordLink and other objects:

1.	Qpwordlink	Uses	Qpcommand
2.	Qpwordlink	Uses	Qpquery
3.	User	Uses	Qpwordlink

1. A WordLink can link one or more QueryPlus command objects to a WinWord document; a QueryPlus command can be used by one or more WinWord documents by using a WordLink object of the same name (N[0]-M[0]).

2. A WordLink can link one or more QueryPlus query objects to a WinWord document; a QueryPlus query can be used by one or more WinWord documents by using a WordLink object of the same name (N[0]-M[0]).

3. A user can use a number of WordLink objects; a WordLink object can be used by a number of users (N-M[0]).

Any object and relationship lists can be generated for the ***WordLink*** object. ***WordLink*** objects can be deleted or modified only with the aid of *QueryPlus* (see the section entitled 'QueryPlus' in the OfficePlus online help).

ExcelLink

An ***ExcelLink*** object is a link between an Excel object and one (or more) ***QueryPlus Command*** or ***QueryPlus Query*** object(s) that is (are) created with the aid of the Windows tool *QueryPlus*. An ***ExcelLink*** is identified by the name of the Excel object (see the section entitled 'QueryPlus' in the OfficePlus online help).

Relationships between an ExcelLink and other objects:

1.	Excellink Uses	Uses Excellink	Qpcommand2. ExcellinkUses	Qpquery3.	User

1. An ExcelLink can link one or more QueryPlus command objects to an Excel object; a QueryPlus command can be used by one or more Excel objects by using an ExcelLink object of the same name (N[0]-M[0]).

2. An ExcelLink can link one or more QueryPlus query objects to one Excel object; a QueryPlus query can be used by one or more Excel objects by using an ExcelLink object of the same name (N[0]-M[0]).

3. A user can use a number of ExcelLink objects; an ExcelLink object can be used by one or more users (N-M[0]).

Any object and relationship lists can be generated for the ***ExcelLink*** object. ***ExcelLink*** objects can be deleted or modified only with the aid of *QueryPlus* (see the section entitled 'QueryPlus' in the OfficePlus online help).