BC407

Reporting: QuickViewer, InfoSet Query and SAP Query

SAP NetWeaver

Date _____

Training Center _____

Instructors _____

_____

Education Website _____

Participant Handbook

Course Version: 2006 Q3
Course Duration: 2 Day(s)
Material Number: 50084721

An SAP course - use it to learn, reference it for work

Internal Use SAP Partner Only

Copyright

Trademarks

Disclaimer

Internal Use SAP Partner Only

About This Handbook

This handbook is intended to complement the instructor-led presentation of this course, and serve as a source of reference. It is not suitable for self-study.

Typographic Conventions

American English is the standard used in this handbook. The following typographic conventions are also used.

Type Style	Description
Example text	Words or characters that appear on the screen. These include field names, screen titles, pushbuttons as well as menu names, paths, and options. Also used for cross-references to other documentation both internal (in this documentation) and external (in other locations, such as SAPNet).
Example text	Emphasized words or phrases in body text, titles of graphics, and tables
EXAMPLE TEXT	Names of elements in the system. These include report names, program names, transaction codes, table names, and individual key words of a programming language, when surrounded by body text, for example SELECT and INCLUDE.
`Example text`	Screen output. This includes file and directory names and their paths, messages, names of variables and parameters, and passages of the source text of a program.
`Example text`	Exact user entry. These are words and characters that you enter in the system exactly as they appear in the documentation.
`<Example text>`	Variable user entry. Pointed brackets indicate that you replace these words and characters with appropriate entries.

Internal Use SAP Partner Only

Icons in Body Text

The following icons are used in this handbook.

Icon	Meaning
	For more information, tips, or background
	Note or further explanation of previous point
	Exception or caution
	Procedures
	Indicates that the item is displayed in the instructor's presentation.

Contents

Internal Use SAP Partner Only

Course Overview

In this course you will learn how to use the QuickViewer, SAP Query and InfoSet Query reporting tools

Target Audience

This course is intended for the following audiences:

- Query developers
- Query administrators

Course Prerequisites

Required Knowledge

- Basic knowledge of a programming language
- Experience with an SAP system

Course Goals

This course will prepare you to:

- Use the QuickViewer, SAP Query and InfoSet Query reporting tools

Course Objectives

After completing this course, you will be able to:

- Create and execute QuickViews
- Create and execute SAP Queries
- Create and execute InfoSet Queries
- Maintain InfoSets
- Administer users for SAP Queries and InfoSet Queries

SAP Software Component Information

The information in this course pertains to the following SAP Software Components and releases:

Internal Use SAP Partner Only

Unit 1

Overview

Unit Overview

This unit provides information that is valid for all reporting tools and that you will need for the whole course.

Unit Objectives

After completing this unit, you will be able to:

- Explain the principle according to which all reporting tools work

Unit Contents

Internal Use SAP Partner Only

Lesson: Overview

Lesson Overview

This lesson provides information that is valid for all reporting tools and that you will need for the whole course.

Lesson Objectives

After completing this lesson, you will be able to:

- Explain the principle according to which all reporting tools work

Business Example

You want to familiarize yourself with the principle according to which all reporting tools work.

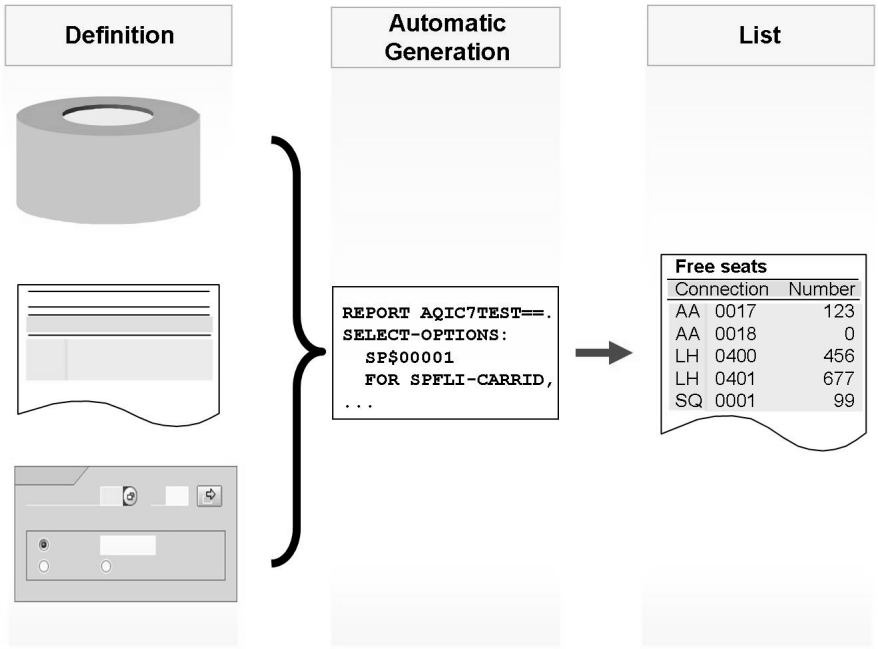


Figure 1: Reporting Tools: Basic Principle

The reporting tools dealt with in this course (QuickViewer, SAP Query and InfoSet Query) work according to the same basic principle.

The user must:

- Specify the **data basis** (for example, the tables from which the data is to be read)
- Determine the **appearance** of the list
- Define a **selection screen** for predefining data

The thing that is actually executed is a report (that is, an ABAP program) created (generated) by the system. Therefore, a reporting tool is also described as a report generator.

The user does not require any ABAP knowledge to be able to use the reporting tools.

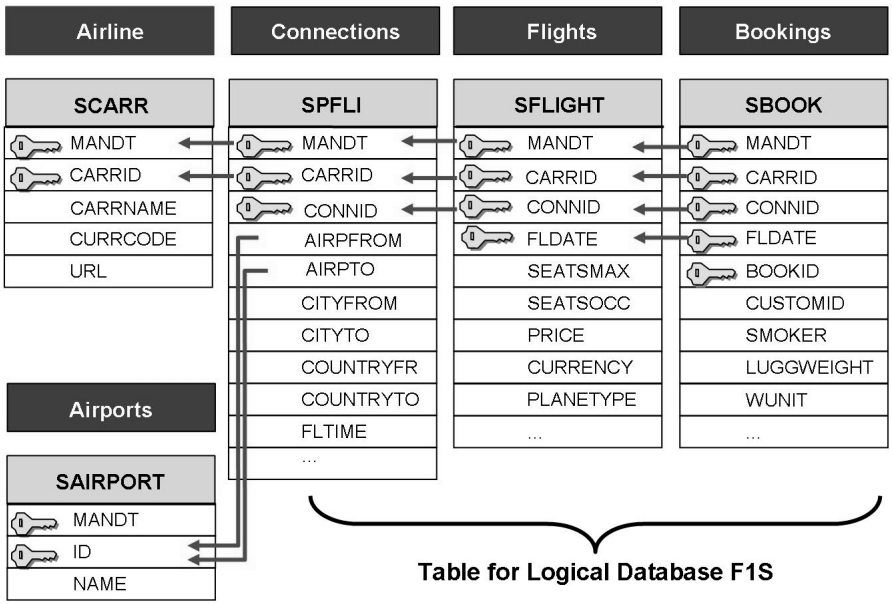


Figure 2: Tables for the Flight Data Model

The most important table fields used in this course and their meaning:

SCARR

CARRID	Airline ID
CARRNAME	Airline name
CURRCODE	Local currency of airline

SPFLI

Internal Use SAP Partner Only

CARRID	Airline ID
CONNID	Connection code
AIRPFROM, AIRPTO	Airport of departure, airport of destination
CITYO, CITYFROM	Start location, destination

SFLIGHT

CARRID, CONNID	See SPFLI
FLDATE	Flight Date
SEATSMAX, SEATSOCC	Maximum number of seats, actual number of seats occupied (this applies to economy class)
PRICE	Basic flight price
CURRENCY	Currency

SBOOK

CARRID, CONNID, FLDATE	See SFLIGHT
BOOKID	Booking number
CUSTOMID	Customer number

Lesson Summary

You should now be able to:

- Explain the principle according to which all reporting tools work

Internal Use SAP Partner Only

Unit Summary

You should now be able to:

- Explain the principle according to which all reporting tools work

Unit 2

QuickViewer

Internal Use SAP Partner Only

Unit Overview

In this unit, you will learn how to use the QuickViewer to create and execute queries.

Unit Objectives

After completing this unit, you will be able to:

- Use the QuickViewer to generate lists
- Use the SAP List Viewer

Unit Contents

Lesson: QuickViewer

Lesson Overview

In this lesson, you will learn how to use the QuickViewer to create and execute queries.

Lesson Objectives

After completing this lesson, you will be able to:

- Use the QuickViewer to generate lists
- Use the SAP List Viewer

Business Example

You have been asked to use the QuickViewer to create and execute queries.

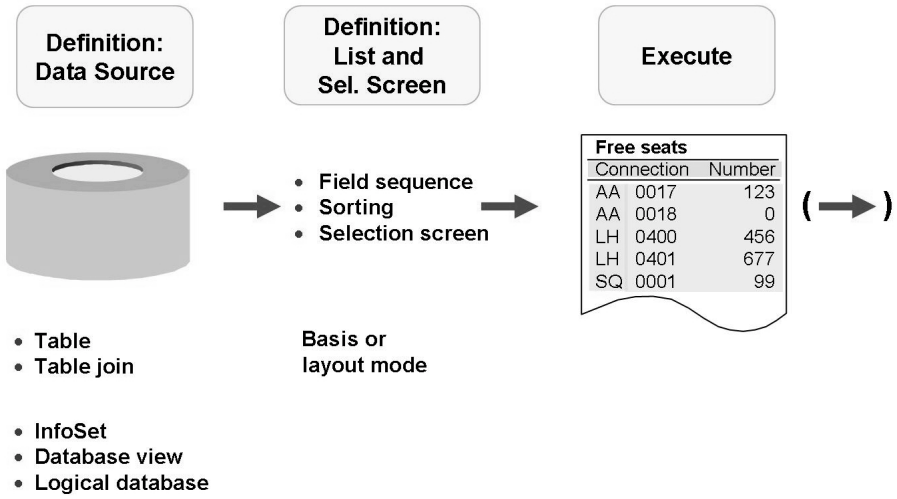


Figure 3: QuickViewer: Basic Principles

The QuickViewer is a tool for developing ad hoc reports that is new in SAP R/3 4.6A. There are three steps to using the QuickViewer: **Defining data bases, structuring lists/selection screens** and **executing these**.

There are different types of data source for you to choose from:

- In the simplest case, you access a table
- You can also connect several tables for your QuickView in a table join.
- If you use several tables regularly with the same connection in different QuickViews or Queries, you can connect these centrally. InfoSets, data base views from the ABAP dictionary or logical databases are all possible sources.
- You can structure the list and selection screen in two different ways: Using the basic mode (described in this unit) or the more complex layout mode (described in the *SAP Query* unit).

You do not have to do anything apart from define the QuickView to execute it, not even save it.

You can edit the list you created, for example, you can forward it to a word processing system or save it with the list definition to display it again later.

QuickViewer: Create

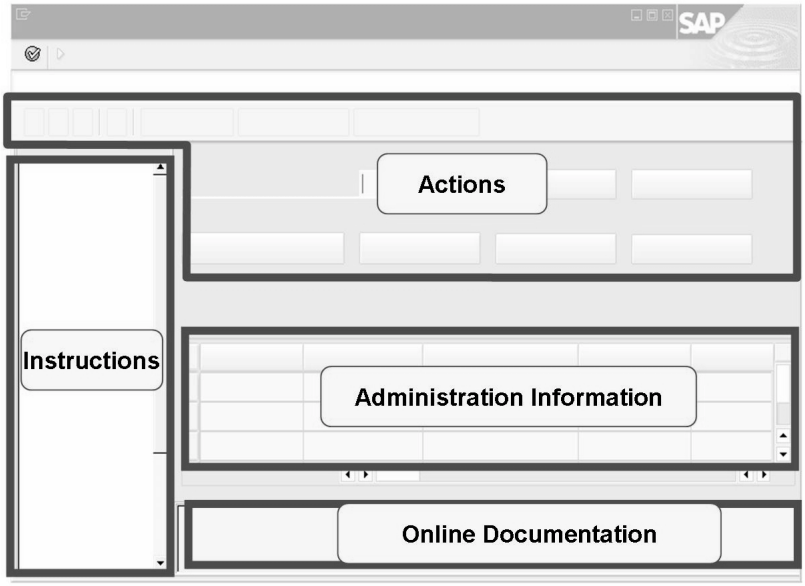


Figure 4: QuickViewer: Initial Screen

You start the QuickViewer either by calling transaction SQVI or using the menu path *System → Services → QuickViewer*.

The initial screen is made up of four screen areas:

- In the instructions, you can see a **short introduction** to the QuickViewer with links to the online documentation
- You use the **pushbuttons** to reach the various editing modes of the quick viewer, for example, the display or maintain mode.
- In the **administration area**, you can see a list of QuickViews that you created yourself, along with information such as the title or the data basis.
- Finally, in the lower right area, you see links to the most important **help topics**.

You can change the width of the instructions area by clicking on the separating wall with your right mouse button and, keeping the right mouse button pressed down, dragging the wall to the required position. You can change the height of the online documentation area in the same way.

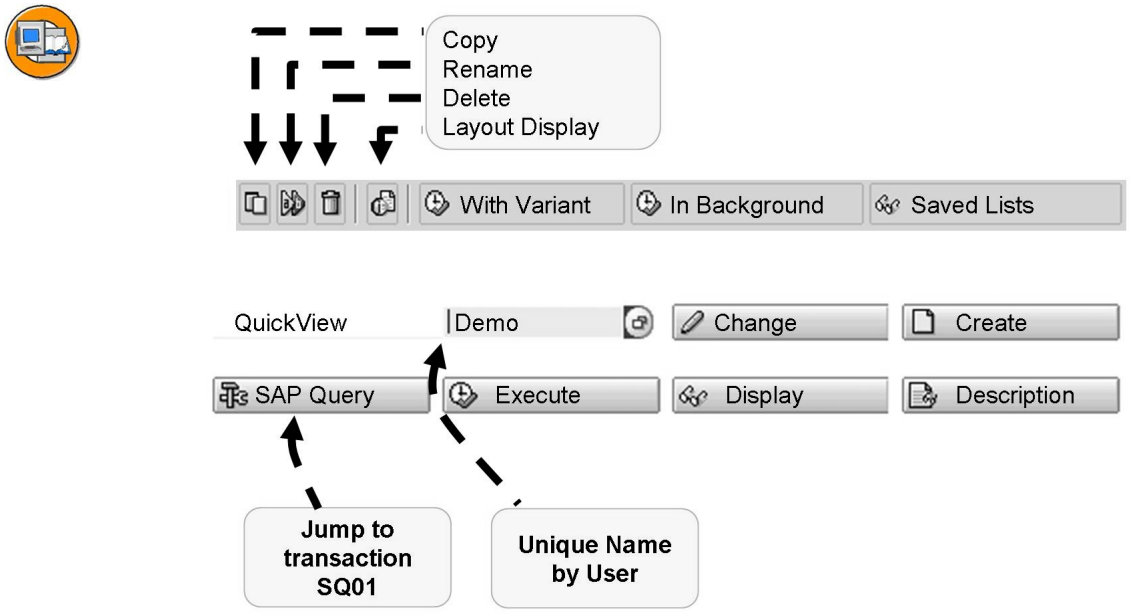


Figure 5: Actions

You can choose the **processing type** for your QuickView using the menu paths or the pushbuttons shown. To do this you have to either enter the name of the QuickView in the input field or choose a QuickView from the list by highlighting it in the selection column.

Each user creates their own **user-specific QuickViews**, which cannot be seen by other users (and therefore cannot be copied). However, you can convert a QuickView into a SAP Query.

Since QuickViews are not connected to the change and transport system, a customer namespace is not required. You can therefore **choose any name that you wish** – only the usual restrictions for query objects apply: Letters, hyphens and underscores only.

You can execute QuickViews directly, with a variant (that is, by predefining the selection screen) or in the background.

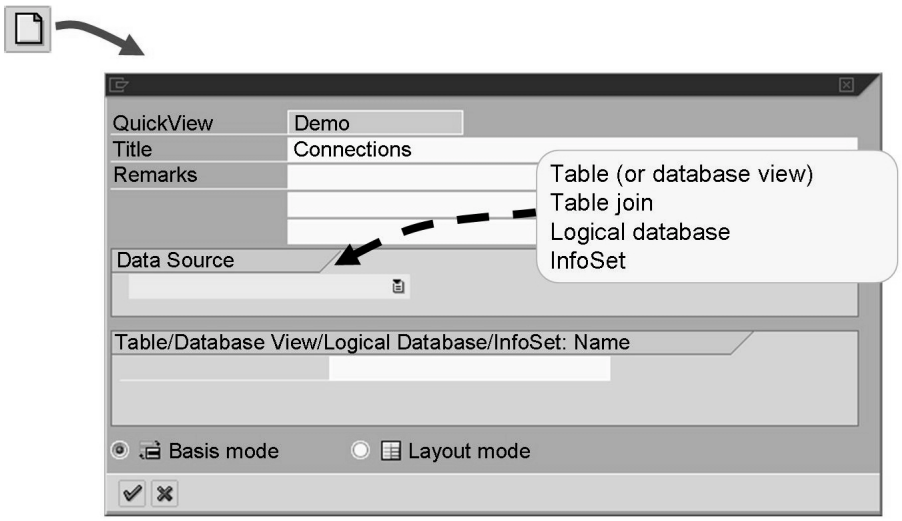


Figure 6: Create a QuickView

When you create a QuickView (for example, using the pushbutton), a dialog box appears.

The text that you enter in the *Title* field appears not only in the initial screen overview, but also as the title of the QuickView when you execute it later.

You must specify a data source in order to generate a QuickView. (See above for details).

You can access the dataset specified by the data source and add any number of fields to the selection screen or list from the dataset.

You can use the basic or layout mode to define the selection screen and list. You use the layout mode for QuickViews in the same way as the layout mode for SAP Queries. The procedure is described in the SAP Query unit.

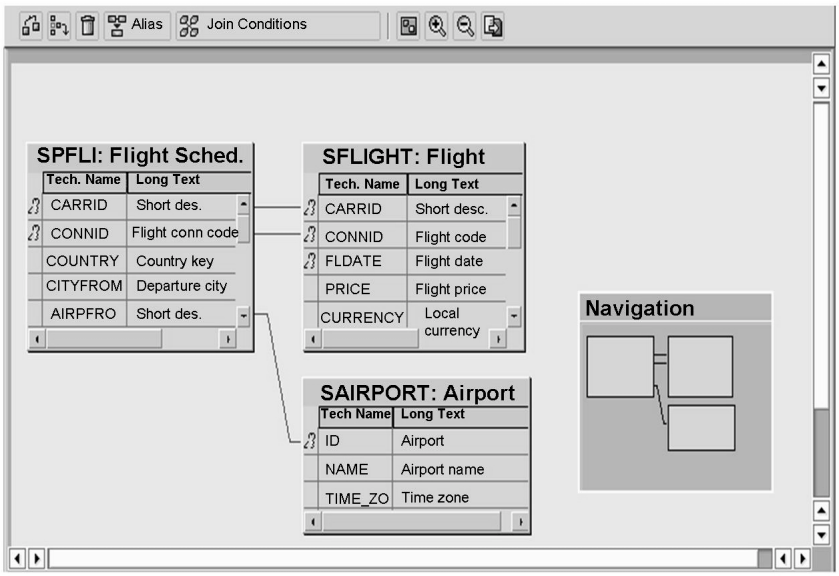


Figure 7: Join Definition

If you specify a table join as a data source, you have to define the join. Before you define the selection screen and list, another screen appears automatically for you to define the join. The join that you define there is specific to a QuickView. This means that it cannot be used more than once (unlike a database view, an InfoSet or a logical database).

You define the table join graphically. For example, you use the pushbutton to insert the required tables. You have to specify the links (join conditions) between the tables, but you can choose for the system to insert a standard proposal. The system evaluates information from the ABAP Dictionary (in the same domain).

A join condition between two tables is filled when both of the joined fields have the same contents.

You can only include tables once for each join. However, you can enter an alias name for a table, using which you can include the table again. You then connect alias tables in the same way as normal tables.

You can only include transparent tables in a join and not pool or cluster tables.

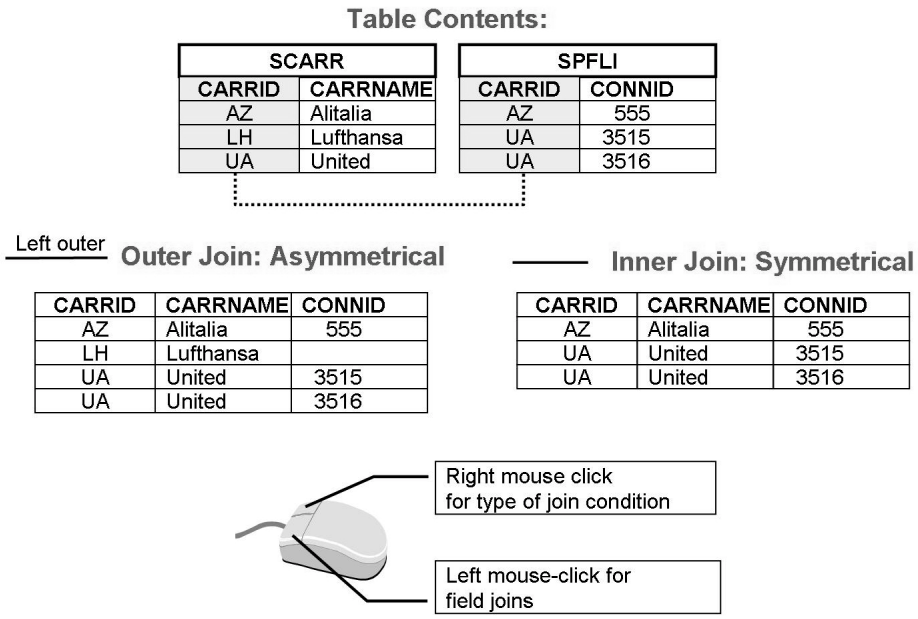


Figure 8: Defining Joins

The resulting quantity depends on the type of join:

For **inner joins**, only data records that have join conditions are displayed in the tables. For example, an inner join between tables *SCARR* and *SPFLI* has at least one connection (flight) from the *SPFLI* table in the target quantity for each airline of *SCARR*.

For **outer joins** (also called left outer joins), data records that do not have any join conditions are also included in the left-hand table. If you, for example, join the tables *SCARR* and *SPFLI* using the *CARRID* field with an outer join logic, the target quantity can contain airlines with or without flights.

If you add a table to a table that is already included, an inner join is defined first. If you want to use an outer join, you click with the right mouse button on the connection and then on *Left Outer Join*. In the dialog box that follows, you can set up an outer join as the join type. This is then displayed using the addition *left outer join*.

Since an outer join can have an asymmetrically structured target quantity, the sequence of the tables is important. The table displayed on the left of the graphic is treated as the left-hand table in a join on a database level.

You can create joins between two fields using drag & drop (dragging while keeping the left mouse button held down).

To delete a join between two fields, click with the right mouse button on the series line and choose *Delete Link*.

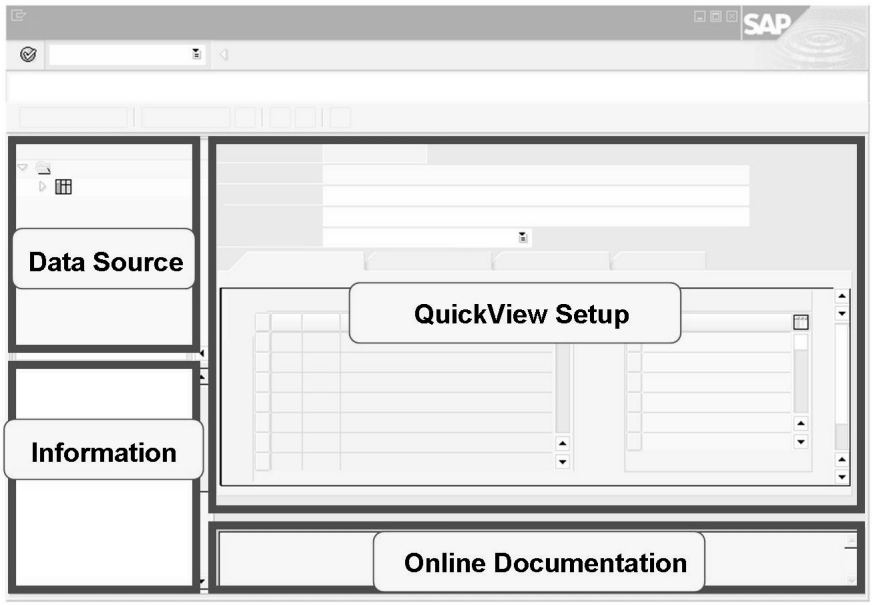


Figure 9: Basic Mode: Principle Structure

In basic mode, the screen is divided into four areas.

- The available fields (data source) are displayed to the left in tree form.
- Further information on how to work in the basic mode displays in the lower left window.
- In the upper-right screen area, you maintain the title and comments as well as control the type of output (for example, *classic list* or *SAP List Viewer*). This is also where you define the list structure, set the sort sequence and determine the selection criteria.
- You can branch to the online documentation from the lower right window.

You can change the width and height or the four areas by clicking on the relevant separating wall and, keeping the mouse button pressed down, dragging the wall to the required position.

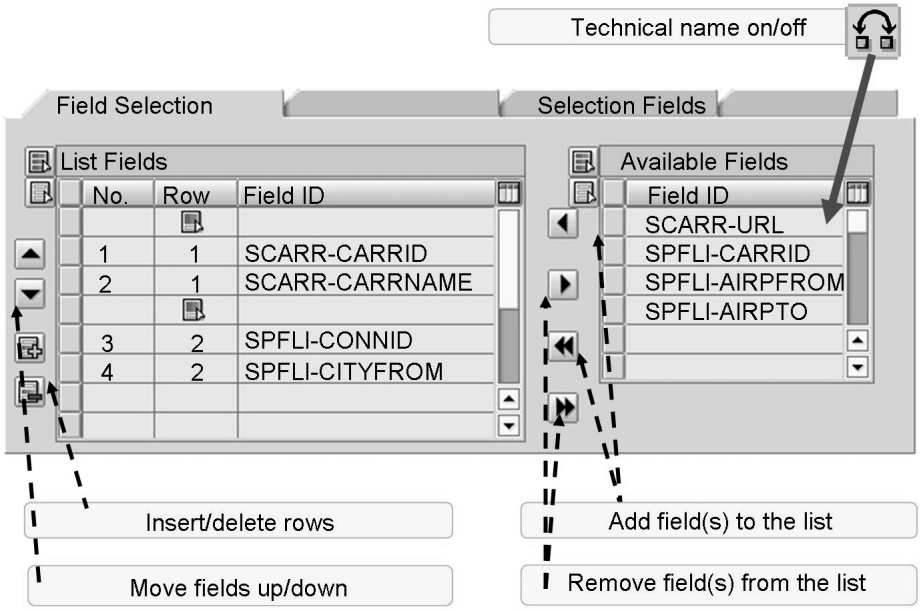


Figure 10: Structuring a QuickView in the Basic Mode

You can structure your QuickView using two table controls. Select the fields you want in your list in the right table control (“Available Fields”) and use the transfer functions to move them to the left table control (“List Fields”).

You can also control how many lines the list should have in the left table control (*List fields*). To insert a line break before a field, put your cursor in the field in the left table control and click on the pushbutton with the green plus sign at the bottom left. To remove the line break, you put the cursor in the left table control on the line that you want to remove the break from and confirm this by clicking on the red minus sign.

You proceed in the same way with selection fields by highlighting the required fields in the right table control (“Possible Selection Fields”) and transferring them to the left control (“Selection Fields”).

If you do not define a selection screen or do not make any entries in the selection screen when you execute the QuickView, a dialog box appears which asks you to specify an upper limit for the number of data records.

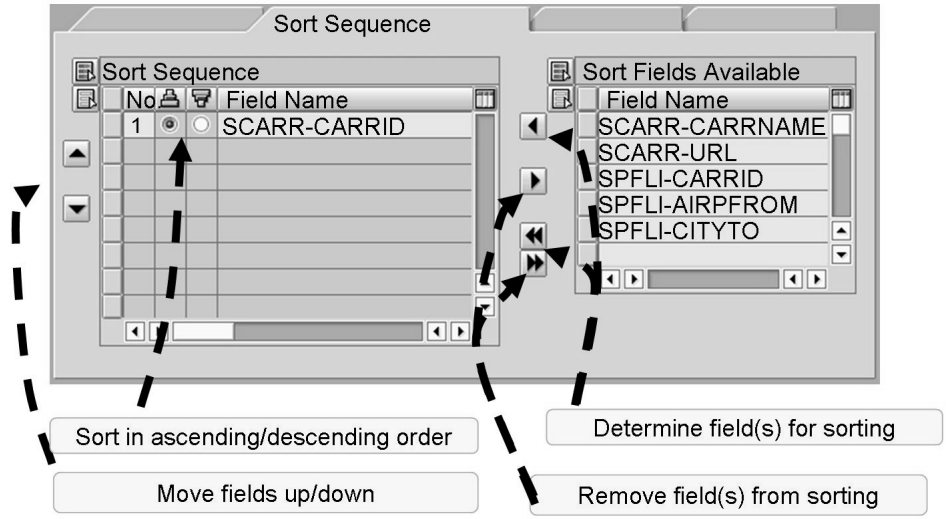


Figure 11: Determining the Sort Sequence

You determine the sort sequence on the second tab in the same way as the list fields, that is, you select the required fields from the right table control (the list of data source fields) and transfer them to the left table control.

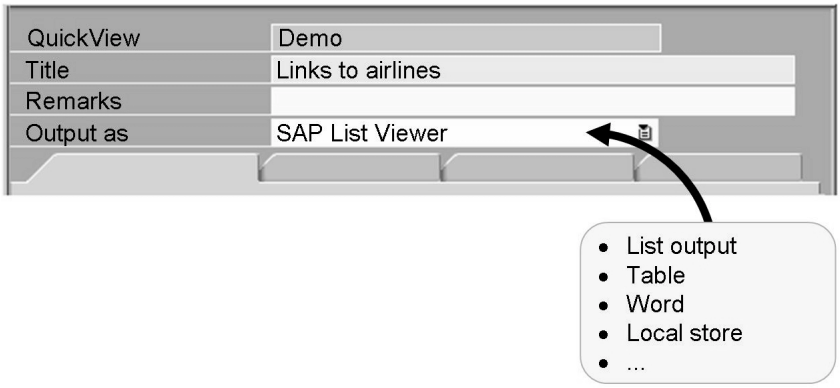


Figure 12: Output Options

You can define the type of list output used in two ways:

- You can find a selection of the most important output options in the *Output as* dropdown box in the upper-right area of the basic mode screen.
- You see a complete list of all output options when you execute the QuickView on the selection screen.

There are three main output options:

- Output within the SAP system
- Direct transfer to an external tool (for example, MS Word or MS Excel)
- Direct saving (for example, as a local file).

The most important output options in the SAP system are:

- **SAP List Viewer** (previously the ABAP List Viewer): See the section later in this unit.
- **Table**: This list is displayed in a table control. It contains interactive functions such as sorting or displaying in the SAP List Viewer.
- **List output (ABAP list)**: The result is displayed in a classic list. This form of output is especially suited to hierarchical lists (that is, lists with multiple lines). You can also save the displayed list and display it from the initial screen of the QuickViewer at a later date with the same data, without having to make your data selections again. The list output also has various interactive functions.

For more types of output, see the online documentation: *SAP Library → ABAP Workbench → SAP Query → QuickViewer → Output Options*.

QuickViewer: Execute

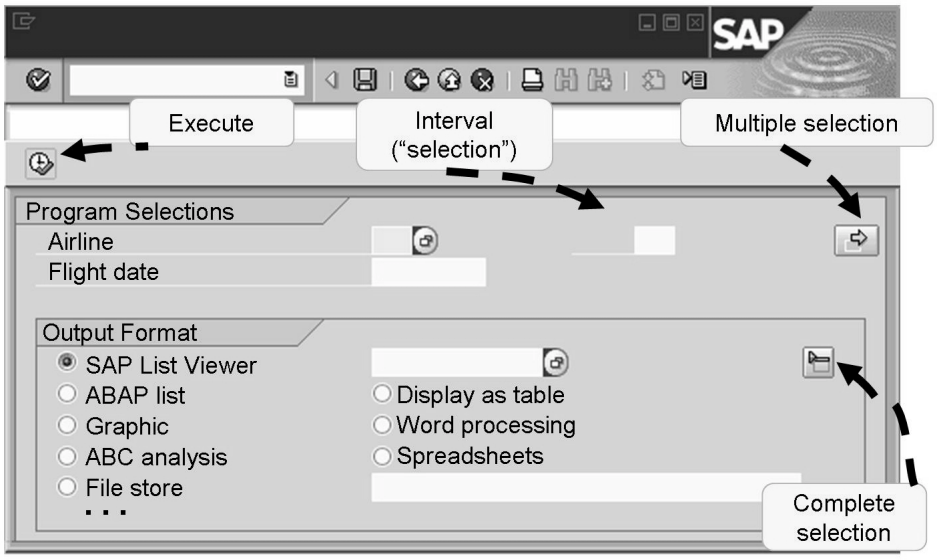


Figure 13: The Selection Screen

Reports, including QuickViews, normally send selection screen on which the user can preselect data. This has two aims: To make the list more clear since it only contains data that is actually required and also to minimize data selection, which leads to improved performance.

On the selection screen you can have simple input fields (parameters) or complex (selections) as well as radio buttons and checkboxes.

In the upper area of the selection screen ("**Program Selections**") when you execute a QuickView, you see the fields that you filled when you created the QuickView. In the lower screen area, you can decide on one of the possible output forms. This does not have to be the same one as you set in the QuickView definition.

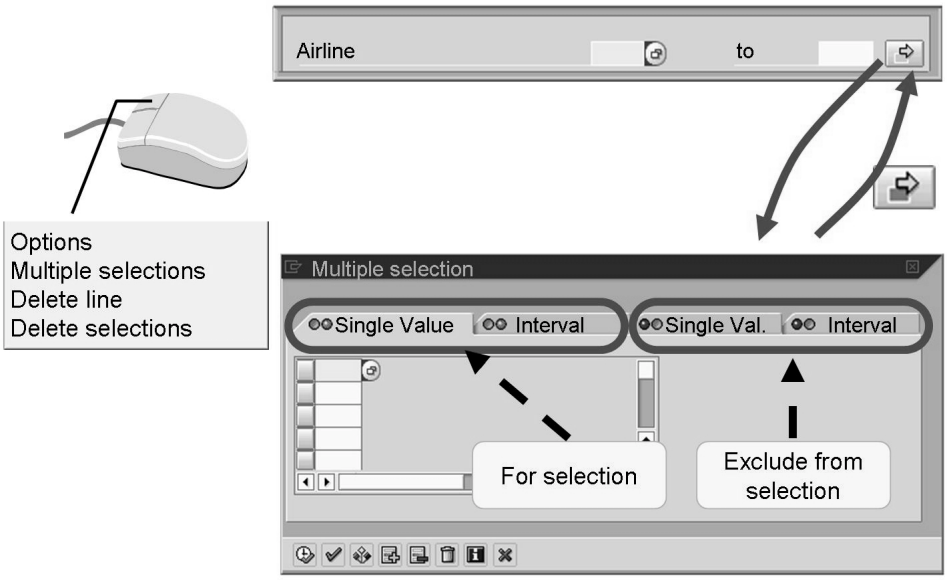


Figure 14: Selection Screen: Selections

Selections give you the options of entering complex, multiple selections. You can enter one or more values or intervals. You can also exclude single values or intervals from the selection. If multiple selections are made, the color of the arrow changes from white to green.

The context menu (right mouse button) allows you to enter comparative operators ("Options"), to enter several values, to delete a single value or interval ("Delete Lines") or to delete all entries.

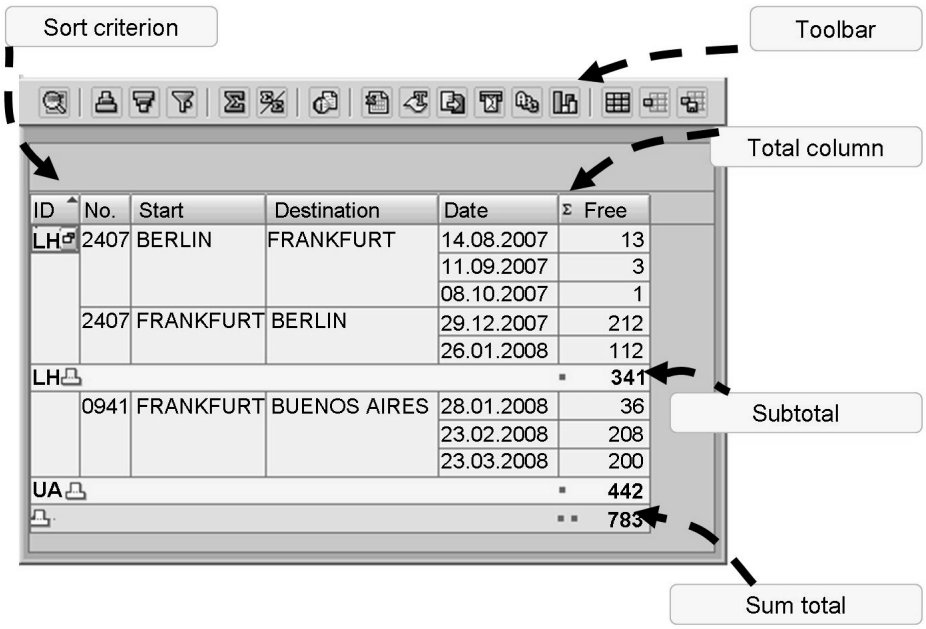


Figure 15: SAP List Viewer: Overview

The SAP List Viewer (previously the **A**BAP **L**ist **V**iewer or ALV) is the most flexible display option and is therefore the default setting in the QuickViewer. You can control the extensive functions either by clicking with the mouse within the display or using the toolbar.

You can save column settings in a layout. You can then select these settings in the selection screen.

None of the functions changes the data selection - they merely affect the display.

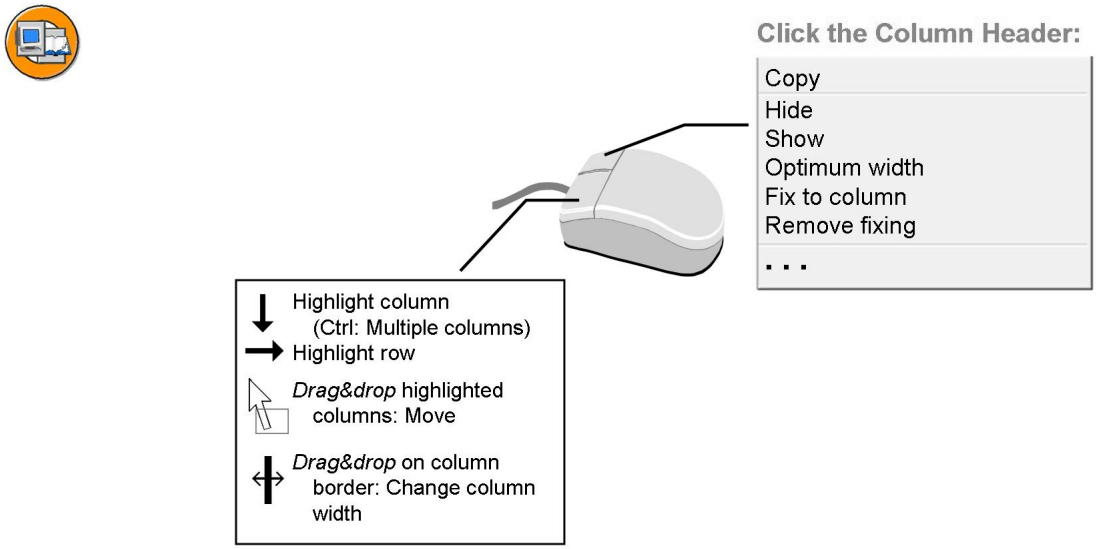


Figure 16: SAP List Viewer: Using the Mouse

Functions of the left mouse button:

- You **highlight columns** by clicking with the left mouse button on the title of a column. You highlight several columns by holding down the control key on the keyboard at the same time. To undo the highlighting you have to click on the relevant column again.
- You **highlight lines** by clicking on the line.
- You move selected columns by holding down the left mouse button and dragging the column to the required position (**Drag&Drop**).
- You change the **column width** by holding down the left mouse button and dragging a column margin in the header line to the required position (Drag&Drop).

The mouse pointer changes according to the function you are using.

Important functions of the right mouse button (**context menu**) when clicking on a column header:

- You can **hide** selected columns for a clearer layout. By choosing *Show*, you can choose in a dialog box to display the required columns again.
- If you choose the *Optimum Width* function, all of the columns displayed are adjusted to their optimum width.
- You can define a highlighted column as the last fixed column: When you scroll across, only columns to the right of the highlighted column are scrolled through. This is particularly useful is you have a wide list in which not all columns can be displayed on the screen at once. It means that you can still see the most important columns all the time.

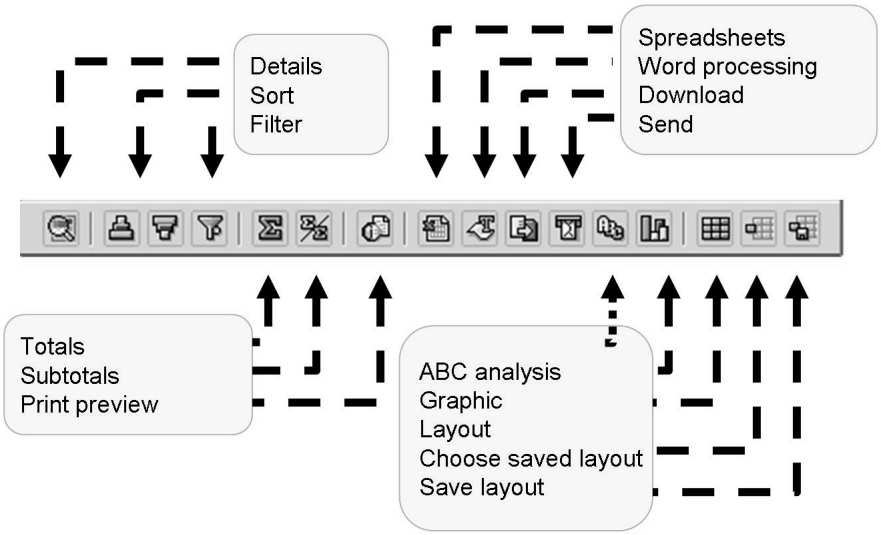


Figure 17: SAP List Viewer: Toolbar

In the SAP List Viewer toolbar, the following functions are available as standard:

If you click on Details a dialog box appears which shows all fields in the line that the cursor is on, including those that are hidden.

You can specify fields as sort criteria by highlighting one or several columns and then clicking on Sort in Ascending/Descending Order. If you have not selected a column, a dialog box appears in which you can control complex sorting with several sort hierarchies. Fields that you have defined as sort criteria have a small read triangle in the header line. Sorting by at least one field is a prerequisite for subtotals.

You can calculate totals for one or more selected numerical columns. You can use the “Subtotal” function to structure control level lists: Select the columns (non-numeric fields only) that you want to use and the corresponding control level totals are displayed. Columns that are totaled have a small sigma (Σ) in the header line.

When you print or download the list, the system always processes the complete list, not just the displayed screen sections.

You can save column properties (for example, sorting or hidden fields) in a layout and adjust the QuickView to your requirements again when you next execute it.

For further information about using the SAP List Viewer, see the online documentation: *SAP Library → Getting Started - Using SAP Software → Working with Lists → SAP List Viewer for SAP GUI.*

1. Select numerical column(s)
2. Calculate total(s)
3. Select non-numerical column(s) for control levels if necessary
4. Create subtotals
5. Instead of 3: Make entries in the dialog box:

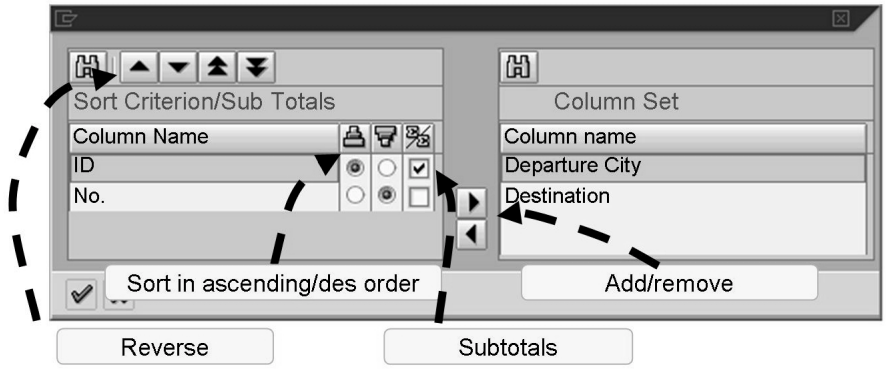


Figure 18: SAP List Viewer: Subtotals

To create subtotals in the SAP List Viewer, proceed as follows:

- Highlight the column(s) for which you want to create the totals.
- Click on the pushbutton with the sigma symbol. This makes the pushbutton for subtotals visible.
- If you do not require complex subtotals, highlight the columns for which you want to create subtotals. These must be non-numerical columns. When you then construct subtotals, the list is sorted in ascending order according to the fields in the highlighted column and subtotals are created for all highlighted columns.
- You can then create the subtotals by clicking on the corresponding pushbutton.
- For more complex subtotals, miss out step three. When you click on the subtotals pushbutton, a dialog box appears in which the columns that you want to use for sorting are shown on the left and the rest of the columns are shown on the right. You move columns from one side to the other by, for example, double-clicking on them. In the left-hand side you can determine for each column in the SAP List Viewer, whether you want to sort in ascending or descending order and whether you want to create subtotals. Moreover, you can change the order of the sort hierarchy by moving the using the pushbuttons with the arrows (top-left) to move the sort criteria up or down.

Exercise 1: Creating a QuickView with a Table

Exercise Objectives

After completing this exercise, you will be able to:

- Create a QuickView in the basis mode that uses a table as its data source

Business Example

You have been asked to create a QuickView based on a table.

Task:

Create a QuickView that displays a standard list of flight bookings.

1. Create a QuickView

   Create a QuickView and give it an appropriate name. Select Table as the data source and specify table SBOOK. Choose the basis mode.

2. Structure the list

   The following fields are to be displayed: *Airline Code, Flight Connection Number, Flight Date, Booking Number, Customer Number, Booking Date* and *Weight of Luggage*.

3. Sort the list

   Sort the list by *Airline Code*, in ascending order.

4. Selection screen

   You want to provide fields *Airline Code* and *Flight Connection Number* in the selection screen.

5. Execute

   Choose List Output (ABAP list) as the type of output.

Solution 1: Creating a QuickView with a Table

Task:

Create a QuickView that displays a standard list of flight bookings.

1. Create a QuickView

   Create a QuickView and give it an appropriate name. Select Table as the data source and specify table SBOOK. Choose the basis mode.

   a) Start the QuickViewer either by calling transaction SQVI or using the menu path *System → Services → QuickViewer*. Create a QuickView, for example, by entering a name of your choice and clicking on the Create button.

   Hint: There is no specific naming convention since QuickViews are user-dependent and cannot be transported.

   Select *Table* as the data source. Choose SBOOK as the table name. Select *Basis Mode* as the processing mode.

2. Structure the list

   The following fields are to be displayed: *Airline Code, Flight Connection Number, Flight Date, Booking Number, Customer Number, Booking Date* and *Weight of Luggage*.

   a) The list is structured in the right-hand area and on the *List Field Selection* tab.

   Select the fields specified in the exercises in the right table control *(Available Fields)*. Use the arrows between the table controls to add or remove fields from the QuickView. Use *Column Left* to add the selected fields to the list (*Fields in list* table control).

3. Sort the list

   Sort the list by *Airline Code*, in ascending order.

   a) Go to the next tab: *Sort Sequence*. In the right table control (*Available Sort Fields*), highlight the *Airline Code* field. Use *Column Left* to include the field in the list (*Selected Sort Sequence* table control).

4. Selection screen

Continued on next page

You want to provide fields *Airline Code* and *Flight Connection Number* in the selection screen.

a) Now switch to the *Selection Fields* tab. Select the fields specified in the exercises in the right table control (*Possible Selection Fields*). Use *Column Left*, to add the selected field to the selection screen (*Selection Fields* table control).

5. Execute

Choose List Output (ABAP list) as the type of output.

a) Above the tab page, you can choose the output form: *Display as List*. To execute, choose Execute (shortcut: F8).

Hint: You do not need to save the QuickView to execute it.

Exercise 2: Optional Exercise: Creating a QuickView with a Table Join

Exercise Objectives

After completing this exercise, you will be able to:

- Create a QuickView that uses a table join as its data source
- Use the SAP List Viewer

Business Example

Create a QuickView that displays information about flight connections, the airports used and the occupancy of the flights in the SAP List Viewer.

Task 1:

Create a QuickView

1. Create a QuickView and give it an appropriate name. Choose Table Join as the data source. Choose the basis mode.

Task 2:

Define the data source: Table join

1. First define a table join over two tables. The tables are SPFLI and SAIRPORT. Implement the links using table fields SPFLI-AIRPFROM and SAIRPORT-ID.
2.

Caution: When you define a table join you can only specify a table once.

Therefore, create an alias table with the name SAIRPORT2 that refers to table SAIRPORT.

3. Now link the alias table with table SPFLI using fields SPFLI-AIRPTO and SAIRPORT2-ID.
4. Add a fourth table: SFLIGHT. Join the fields SPFLI-CARRID with SFLIGHT-CARRID and SPFLI-CONNID with SFLIGHT-CONNID.

Hint: Use the suggestions made by the join condition.

Continued on next page

Task 3:

Structure the list

1. The following fields are to be displayed: *Airline Code, Flight Connection Number, Departure Airport Code and Name, Arrival Airport Code and Name, Flight Date, Occupied Seats*.

Hint: So that you give the airport names for the departure airport and arrival airport that are the same as their identifiers, use the technical names.

Task 4:

Selection Screens

1. You want to provide fields *Airline Code* and *Flight Connection Number* in the selection screen.

Task 5:

Execute

1. Choose the SAP List Viewer (=ABAP List Viewer) as the output type. Create a total for the occupied seats. Create a subtotal for each airline. Hide the column with the airport identifier.

Solution 2: Optional Exercise: Creating a QuickView with a Table Join

Task 1:

Create a QuickView

1. Create a QuickView and give it an appropriate name. Choose Table Join as the data source. Choose the basis mode.

   a) Start the QuickViewer either by calling transaction SQVI or using the menu path *System → Services → QuickViewer*.

   Create a QuickView and give it an appropriate name. Choose *Table Join* as the data source and *Basis Mode* as the processing mode. Enter a suitable title (mandatory field entry).

Task 2:

Define the data source: Table join

1. First define a table join over two tables. The tables are SPFLI and SAIRPORT. Implement the links using table fields SPFLI-AIRPFROM and SAIRPORT-ID.

   a) After the dialog box for determining the title and data source, you reach the screen for defining joins. Choose *Insert table* to add table SPFLI as the first table. Now add table SAIRPORT. The system links the two tables through fields SPFLI-AIRPFROM and SAIRPORT-ID

2.

   Caution: When you define a table join you can only specify a table once.

   Therefore, create an alias table with the name SAIRPORT2 that refers to table SAIRPORT.

   a) Define an alias table by choosing *Alias table*. Now choose *Create* in the subsequent dialog box. Enter SAIRPORT as the table name and enter the alias name SAIRPORT2.

Continued on next page

3. Now link the alias table with table SPFLI using fields SPFLI-AIRPTO and SAIRPORT2-ID.

   a) Now add the alias table as the third table in the table join. The system links SAIRPORT-ID and SAIRPORT2-ID. You do not want this link – you have to delete it. To do this, position the cursor on the joining line and click the right mouse button. Choose *Delete Link* from the menu. Now drag and drop (left mouse button) to add a join condition between fields SAIRPORT-ID2 and SPFLI-AIRPTO.

4. Add a fourth table: SFLIGHT. Join the fields SPFLI-CARRID with SFLIGHT-CARRID and SPFLI-CONNID with SFLIGHT-CONNID.

Hint: Use the suggestions made by the join condition.

   a) Add SFLIGHT as the fourth table. If the system cannot find a join, create the join manually by dragging the field SFLIGHT-CARRID to SPFLI-CARRID and SFLIGHT-CONNID to SPFLI-CONNID. Alternatively, click on *Propose Join Condition* and enter SFLIGHT as the first table and SPFLI as the second.

Hint: You can position the tables anywhere on the screen (Drag&Drop).

   The green arrow in the standard toolbar (shortcut: F3) takes you to the basis mode for structuring the list.

Task 3:

Structure the list

1. The following fields are to be displayed: *Airline Code, Flight Connection Number, Departure Airport Code and Name, Arrival Airport Code and Name, Flight Date, Occupied Seats*.

Hint: So that you give the airport names for the departure airport and arrival airport that are the same as their identifiers, use the technical names.

   a) Proceed as in the previous exercise, point 3.

Continued on next page

Internal Use SAP Partner Only

Task 4:

Selection Screens

1. You want to provide fields *Airline Code* and *Flight Connection Number* in the selection screen.

   a) Proceed as in the previous exercise, point 4.

Task 5:

Execute

1. Choose the SAP List Viewer (=ABAP List Viewer) as the output type. Create a total for the occupied seats. Create a subtotal for each airline. Hide the column with the airport identifier.

   a) Above the tab page, you can select the type of output: *SAP List Viewer (=ABAP List Viewer)*. To execute, choose Execute (shortcut: F8).

   All numerical fields (in this case: *Occupied Seats*) are totaled automatically. The small sigma sign (∑) in the column header indicates this. If necessary, scroll down to the bottom to see the totals. (You total numerical fields in the SAP List Viewer by highlighting the column and clicking on Totals).

   Select both columns with the airport codes by holding down the control key on the keyboard and clicking on the column headers. To hide the columns, click with the right mouse button on one of the two columns and choose *Hide* from the context menu.

Lesson Summary

You should now be able to:

- Use the QuickViewer to generate lists
- Use the SAP List Viewer

Unit Summary

You should now be able to:

- Use the QuickViewer to generate lists
- Use the SAP List Viewer

Unit 3

SAP Query

Unit Overview

In this unit, you will learn about all the options for SAP Query.

Unit Objectives

After completing this unit, you will be able to:

- Use SAP Query to create basic lists, statistics and ranked lists
- Use the interactive functions of these lists
- Save query lists and call them up again

Unit Contents

Lesson: SAP Query

Lesson Overview

In this lesson, you will learn about all the options for SAP Query.

Lesson Objectives

After completing this lesson, you will be able to:

- Use SAP Query to create basic lists, statistics and ranked lists
- Use the interactive functions of these lists
- Save query lists and call them up again

Business Example

You have been asked to use SAP Query to create a query that is comprised of several sublists (a basic list and several statistics).

Introduction to XML

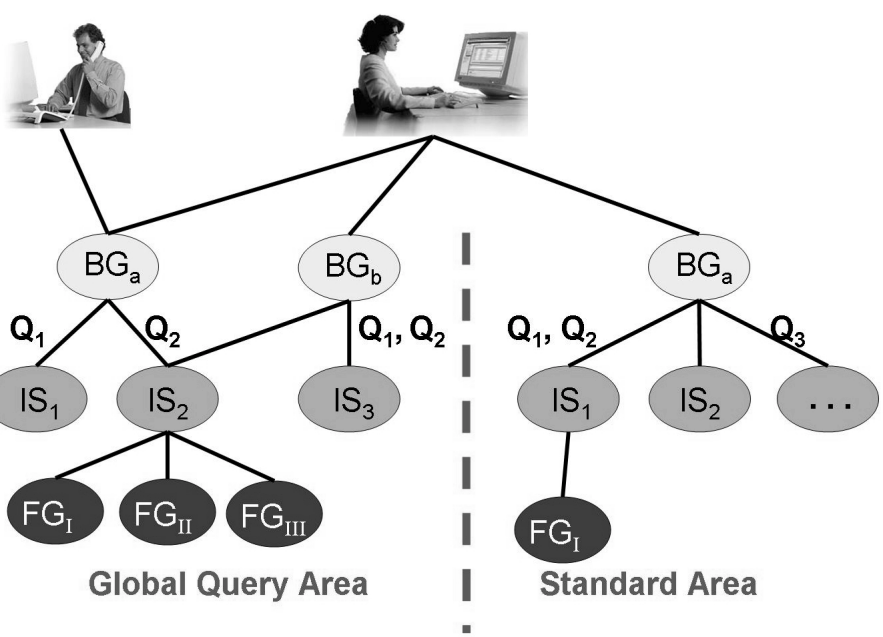


Figure 19: Organizational Environment of the SAP Query

SAP Query has an organizational environment that enables you to give specific user authorizations:

Internal Use SAP Partner Only

Users can create and start queries only if they belong to at least one **user group**. A user can belong to several user groups.

User groups have **InfoSets**. The InfoSet determines the database tables or the fields of those tables which a query can access. You can allocate an InfoSet to several user groups.

The individual fields of the InfoSet are divided into **field groups** for a better overview.

Queries are always created for a specific user group and a specific InfoSet. Users have access to all the queries allocated to their user group.

Finally there are two query areas. A query area covers a number of query objects that are internally complete and consistent - this means objects with the same name but with a different meaning can exist in the two different query areas.

- In the **global query area**, all query objects (user groups, InfoSets, queries) are **cross**-client and have an object catalog entry, that is, an automatic connection to the transport system.
- In the **standard area**, all query objects are client-**dependent** and are not connected automatically to the transport system.

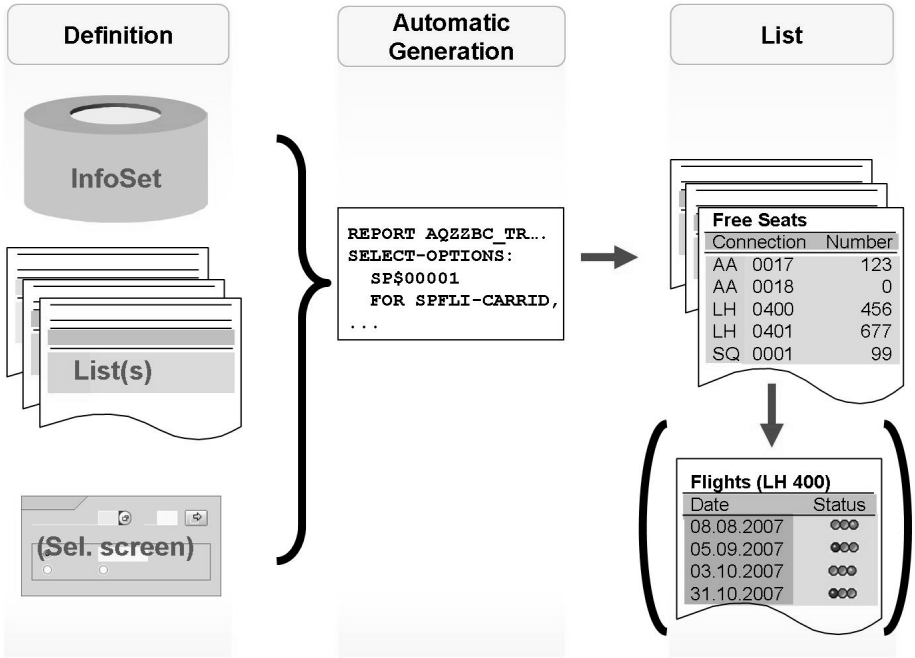


Figure 20: SAP Query: Basic Principle

You create a SAP Query using the same basic principles and processes as in the QuickViewer.

- You select your data basis. Unlike in the QuickViewer, you can only use an InfoSet here. However, from a technical point of view, this can also contain any number of tables, a logical database, and so on
- Next you define the lists. You can have a maximum of one basic list or 18 summarized lists (statistics, ranked lists) for each query.
- If necessary, you can enhance the InfoSet selection screen.

When you start the query, an internal report generator creates a program that corresponds to the list definition. This program then reads the data, processes it, and outputs the data as a list. The program is called *AQmmbbbbbbbbbbbbqqqqqqqqqqqqq*. You can display the program name from the initial screen of transaction SQ01 by following the menu path *Query → More Functions → Display Report Name*.

The individual parts of the name have the following meanings:

- *mm* - Encoded client (standard area) or *ZZ* (global area)
- *bbbbbbbbbbbb* - Name of the user group (12 characters)
- *qqqqqqqqqqqqq* - Name of the query (14 characters)

Spaces in query program names are replaced with '='.

The SAP Query has the same interactive functions as the QuickViewer. You can also link several lists together.

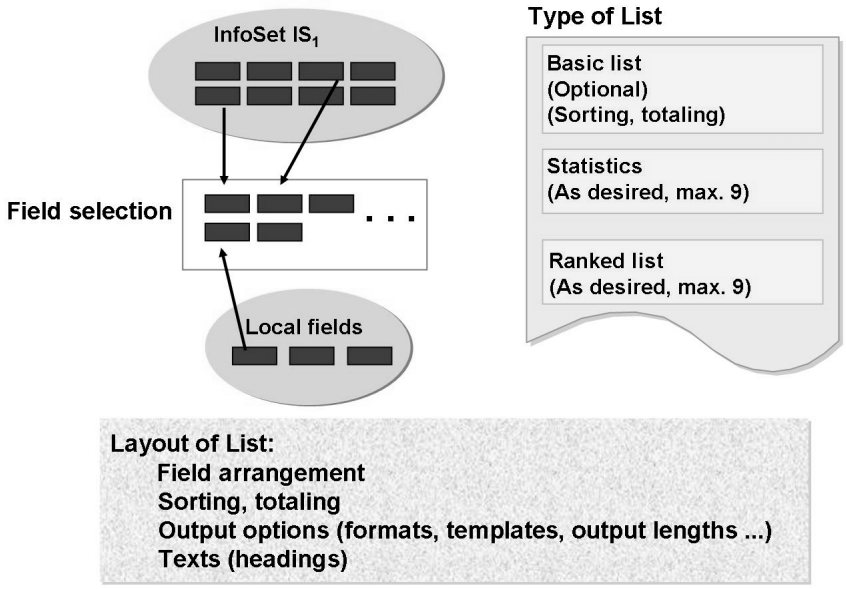


Figure 21: Defining an SAP Query

The SAP Query takes you through a sequence of screen fields in which, by

- Selecting (checkboxes)
- Assigning numbers (sequence, sorting and so on)
- Retrieving texts (headers, group level texts)

you determine the line structure and the list layout.

You can use the Query Painter to create basic lists graphically.

An SAP Query can have different lists:

- **Basic list**: Single line or multiline. You can compress multiline basic lists.
- **Statistics** and **ranked lists**: Compressed data records.
- You can combine different sublists, but with a maximum of one basic list, nine statistics and nine ranked lists.

You can also define local fields within a query, which means you can calculate new values from the data read (as a enhancement to the InfoSet).

Standard interactive functions are offered for a Query list. For example, you can process the query elsewhere (in MS Excel or an ABC analysis), display the query graphically (SAP Graphic), save the query or link it to another query (report/report interface).

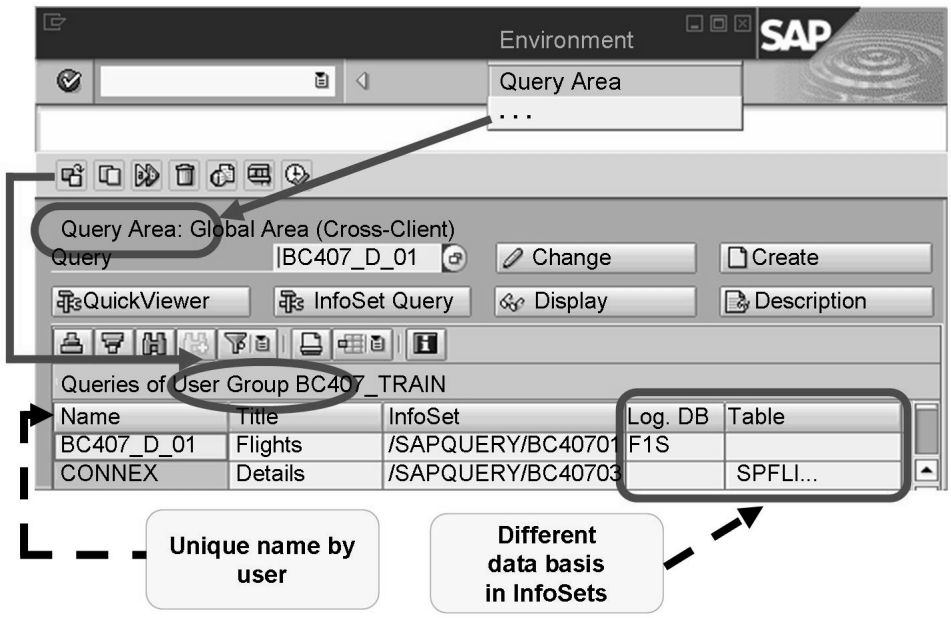


Figure 22: SAP Query: Initial Screen

Start SAP Query either in transaction SQ01or by choosing the SAP Query button on the initial screen of the QuickViewer.

Decide on the **query area** that you want to use: Menu *Environment → Query Area*. (If you always work in the global query area, you can set this in your user profile: Parameter *AQW* with the value *G*. You can go to your user profile by following the menu path *System → User Profile → Own Data*).

Specify one of the **user groups** to which you are assigned. To do this, choose *Edit → Change User Group* or choose the relevant pushbutton. (You can also set the user group in your user profile: The relevant parameter is *AQB*.)

Field Selection and Basic Lists

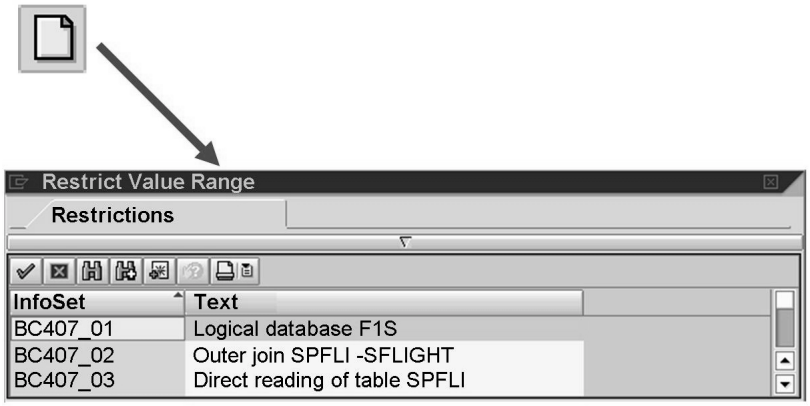


Figure 23: Creating an SAP Query

If you are creating a new query, you have to choose an InfoSet for the query. The system displays a list of all the InfoSets that are assigned to your selected user group.

Once you have chosen an InfoSet (for example, by double-clicking on it) for a query, you can no longer change it for that query since the InfoSet is the basis for data retrieval.

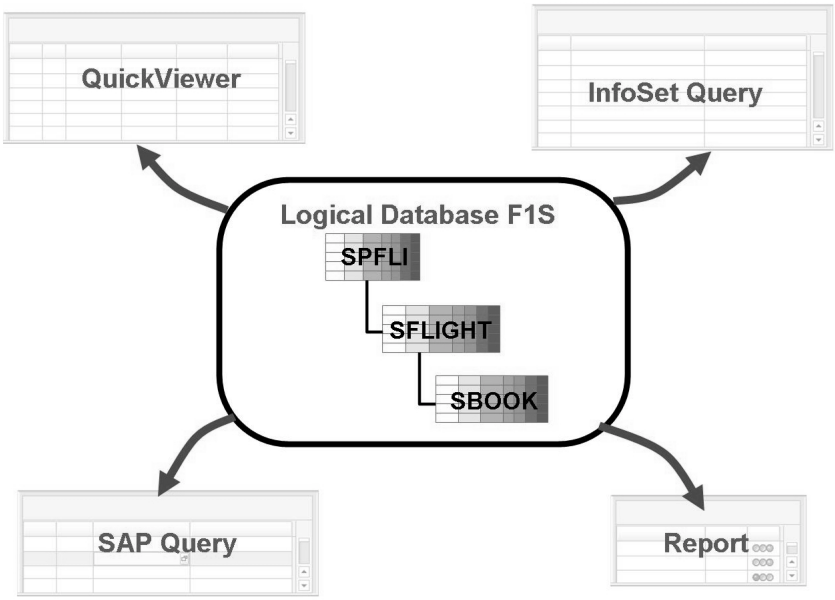


Figure 24: Insertion: Logical Databases

InfoSets can be based on **logical databases** (among other things).

A logical database is an ABAP program that reads **predefined data from the database** and makes it available to other programs (usually reports). A hierarchical structure determines the order in which the data is delivered to the programs. The programmer or user of the reporting tool decides which of the tables that are contained in the logical database he or she actually wants to use.

A logical database also provides the program with a **selection screen** that checks user entries and conducts error dialogs. If, for example, you use a logical database in an InfoSet only the part of the selection screen that matches the tables that are read is sent.

SAP NetWeaver 7.0 provides approximately 250 logical databases.

You maintain logical databases in transaction SE36. You can also find further information there: *Help → Application Help.*

F1S is the logical database used in BC4XX courses. It includes the tables *SPFLI* (connections), *SFLIGHT* (flights) and *SBOOK* (bookings).

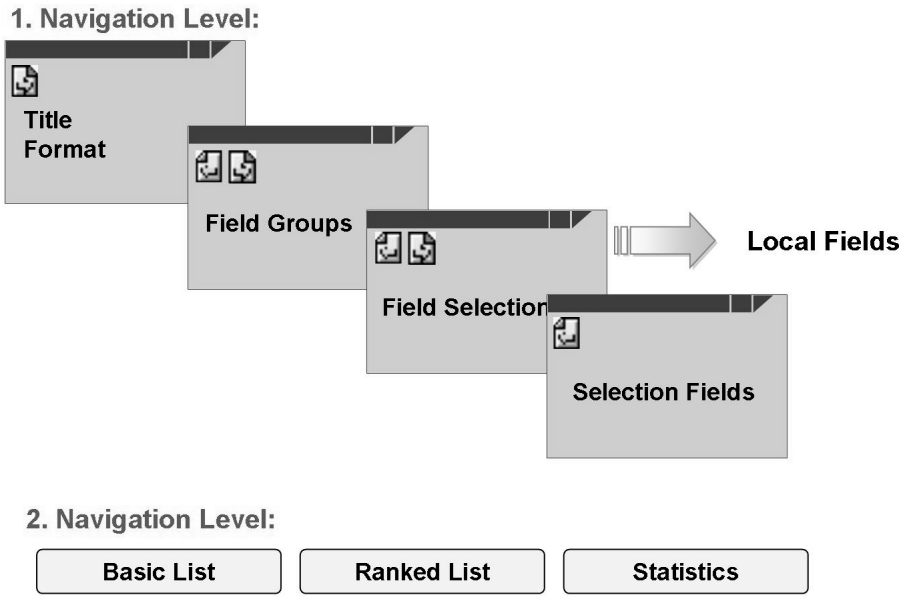


Figure 25: Navigating Between Screens

Once you have chosen an InfoSet when creating your SAP Query or once you have changed an existing SAP Query, a sequence of screens appear.

- **Title, Format**

  This is where you give your Query a title and determine the list format and output type.

- **Field groups**

  InfoSets are divided into field groups. These form logical groups of data.

- **Field selection**

  This is where you choose the data fields for your chosen field groups. If you require local fields, you also define them here. Field selection is mandatory for ranked lists and statistics. If you want to define only a basic list with the graphical query painter and do not need any local fields (to supplement the InfoSet), you can leave this step out.

- **Selection fields**

  You can specify additional selection fields, though this is optional. This extends the selection screen from the one defined in the InfoSet.

You can use the graphical **Query Painter** tool for basic lists. It is identical to the layout mode in the QuickViewer. You define statistics and ranked lists using a range of screen with input fields.

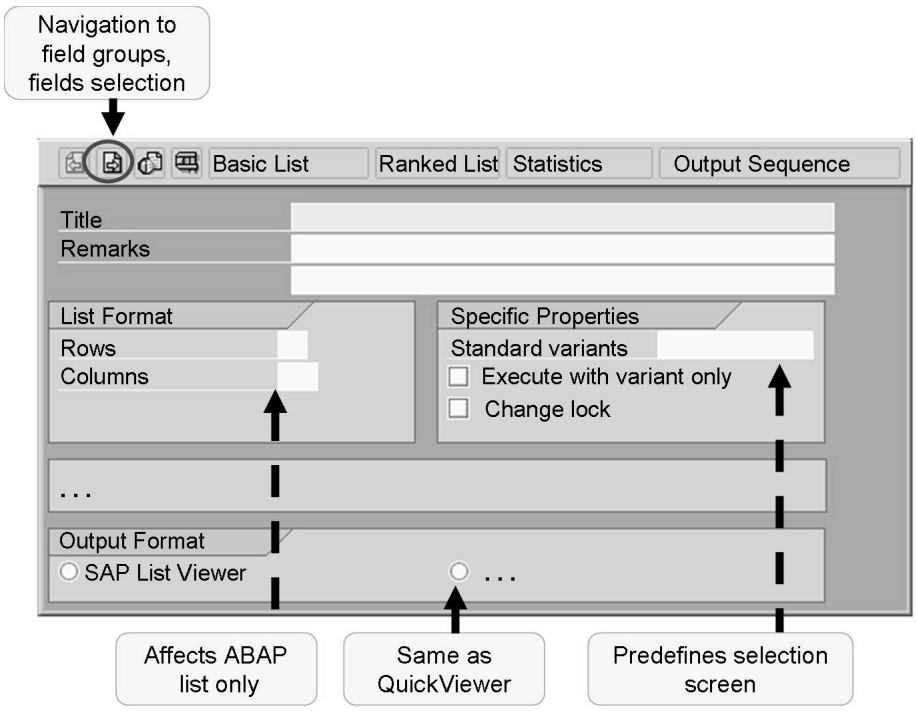


Figure 26: Title, Format

On the first screen you can make entries that affect the entire query, for example:

- Enter a title. The contents of this mandatory input field appear in the list.
- Determine the format of the list (list width and page length). This is only necessary if you are using an ABAP list as the output format.
- Choose the output format. As with the QuickViewer, there are three main output options: Output within the SAP system, transfer to an external tool or save. If you define a selection screen for your SAP Query, you can choose the output type when you run the query.
- Determine a standard variant. Variants are predefined values for the selection screen that the user can create according to his or her requirements. If you enter a standard variant, the query is always called with the corresponding predefined values in the selection screen.

For ranked lists and statistics, you now have to continue with the field group and field selection. If you want your query to have a basic list, it is not absolutely necessary to go to these screens; you can proceed with the basic list definition from this screen by, for example, clicking on Basic List.

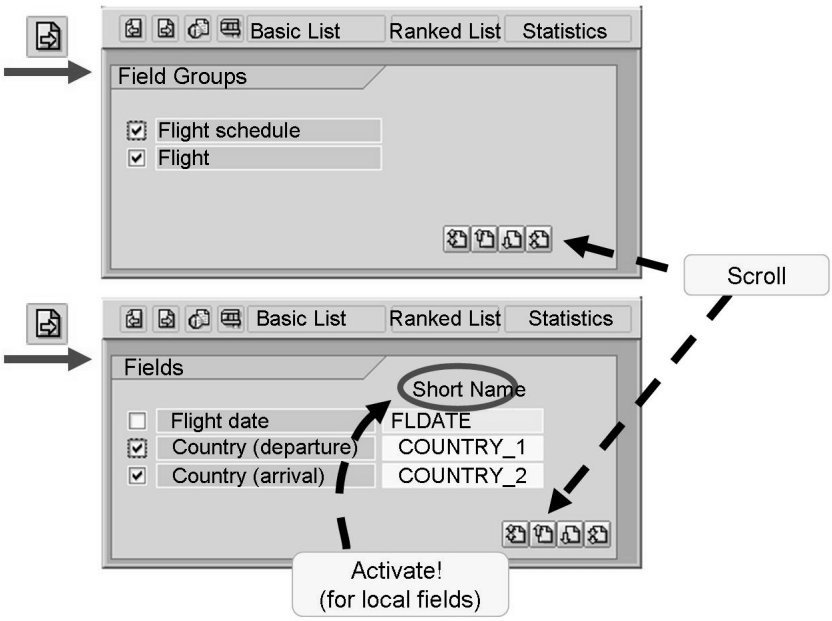


Figure 27: Field Groups and Fields

Field Groups

Choose the field group that you want to use and then navigate to the field selection.

Field selection

Select the fields that you want to have available in statistics, ranked lists or when defining local fields.

If you want to work with local fields, you have to activate the **short names** (menu *Edit → Short Names → Switch On/Off).* Enter the short name for all fields for which you need calculations.

A field with a short description is invoked by its name (in the example above, *FLDATE*), the contents of the field (for example, for the page headers) with &name (here, *&FLDATE*).

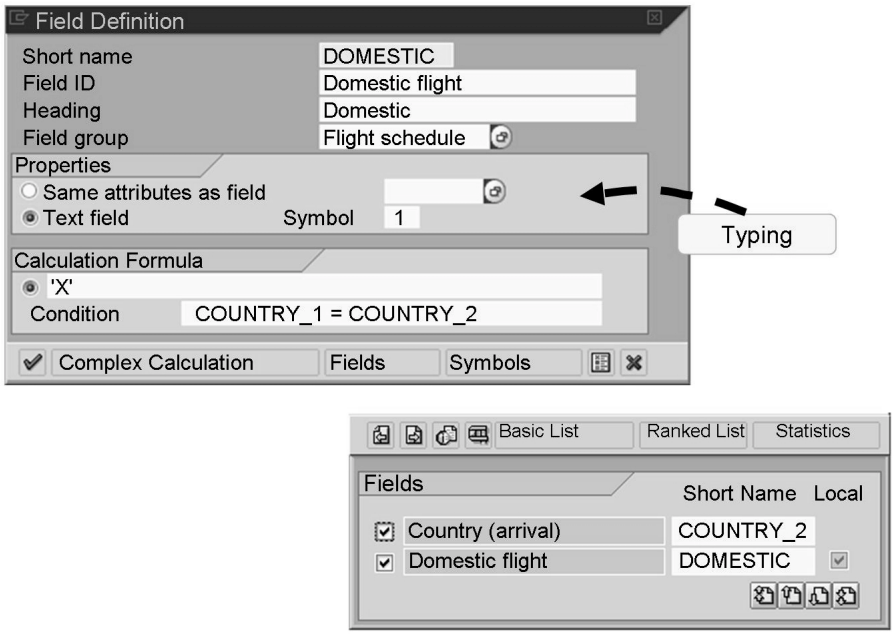


Figure 28: Defining Local Fields

By defining **local fields**, you can generate additional information from the fields that are available in an InfoSet. You can display them along with the "normal" fields of the source InfoSet.

Local fields are only recognized in the query in which they are defined. To create or change the fields, go to the field selection screen using *Edit → Local Fields → Create or Change*. Local fields always need to have a short name. They also have to be assigned to a field group (formally functional group).

To type local fields, you can refer to other fields that have a short description, that is, to other local fields in the same SAP Query. You can type local fields as text, calculation, date or time fields. You can also define the fields as symbols or icons.

Local fields are filled with calculation rules. In the simplest case, calculation rules consist of a single formula formed with normal mathematical rules and consisting of operands and operators. You can see which operators are allowed by clicking on *Complex Calculation*.

You can make the field value calculation condition-dependent. If the condition remains unmet, the field contains a default value. You can set multiple conditions: Click on Complex Calculation to do this.

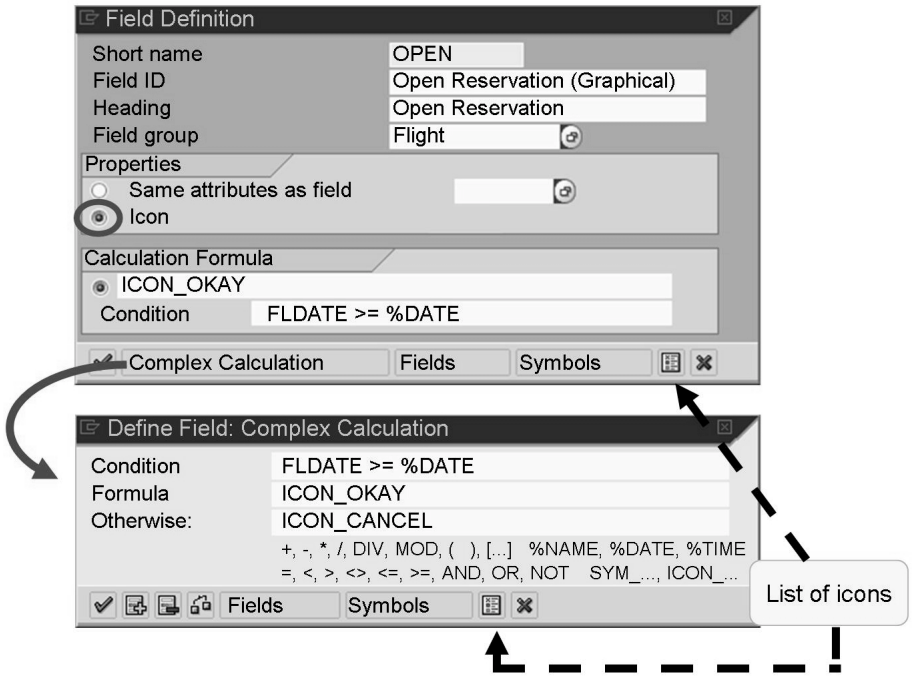


Figure 29: Local Fields: Example with Icon

To create local fields as icons, you have to type these as such in the attributes.

Enter the required name in the calculation formula. You can see which icons are available by clicking on the icon pushbutton: Then double-click on an icon in the list displayed to select it.

The following special fields are available for formulas:

- *%NAME*: Name of the user who executes the query
- *%DATE*: Current date when the query is executed
- *%TIME*: Time when the query is executed

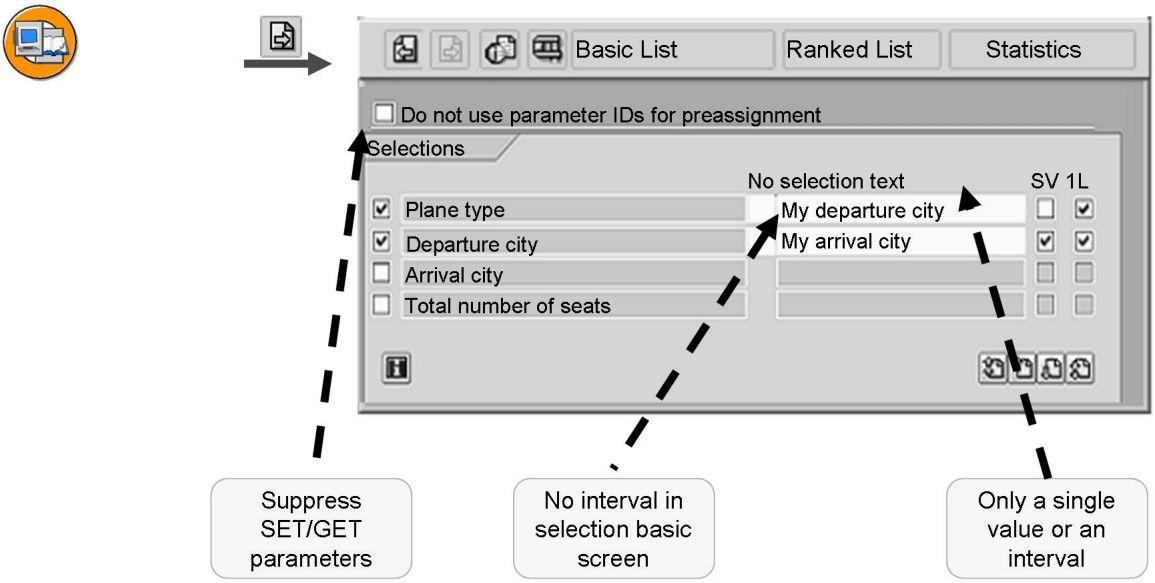


Figure 30: Selection fields

You can define all of the fields selected for the task as selection criteria. However, since selection fields are generally already available using the InfoSet, you should first check which selections are already delivered by the InfoSet.

Only make fields that are not already defined as selection fields using the InfoSet into additional selection fields.

Under certain circumstances, your additional selections may be automatically populated with the last input value of the user. This technique is called “inputting SET/GET parameters” and is set up in the ABAP Dictionary independent of the SAP Query. When you determine your selections, you can suppress this preassignment if you do not require it.

You can define an individual text that appears on the selection screen for each selection. If you do not specify a text, the system uses a standard text.

If you choose the property *SV* (single value), you can only enter one single value at first in the selection screen (however, the *Multiple selection* function is still available on your additional screen).

If you choose the property *1L* (single line), you can either enter a single value or an interval for the selection. The multiple selection function is no longer possible. If you combine *SV* and *1L*, you can only enter a single value without multiple selection.

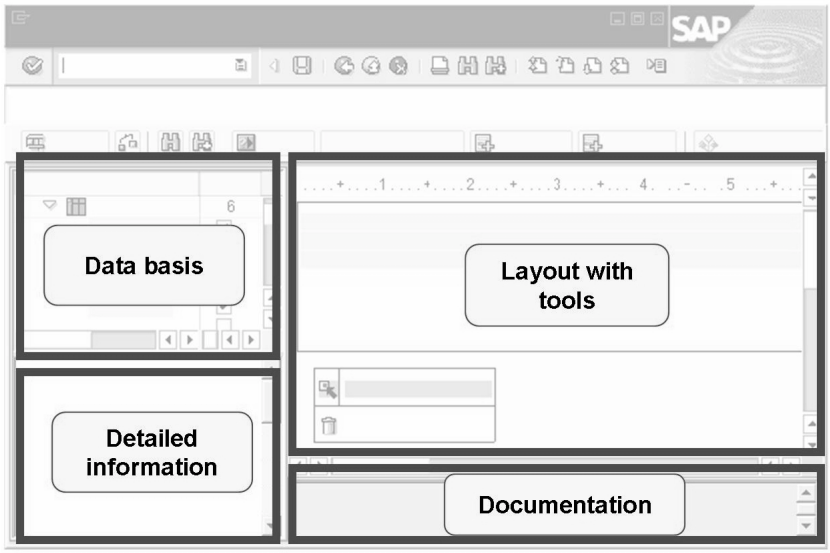


Figure 31: Basic List: Layout Mode (Query Painter)

You use the Query Painter to create basic lists. The Query Painter work area is divided into four sections. The available fields are displayed to the left in tree form. The list layout is displayed in the upper right. Information for the element which is currently active is displayed in the lower left area of the window. Links to documentation and any warnings that are output while formatting the list are displayed in the lower right section of the window.

You can sort the list according to individual fields. You can define control level processing for sorted fields. You can produce a total for each control level and count the fields in a control level.

You can set output formats for lines and individual fields.

You can set additional output options for each line (output dependent on the presence of another line, output of blank lines [before and after a line], page advance and output at the top of a page).

You can set output options for each field (output length, output position, output with edit screen).

You can enter your own column headers or use the system default column headers.

You can define a header and footer for each individual list.

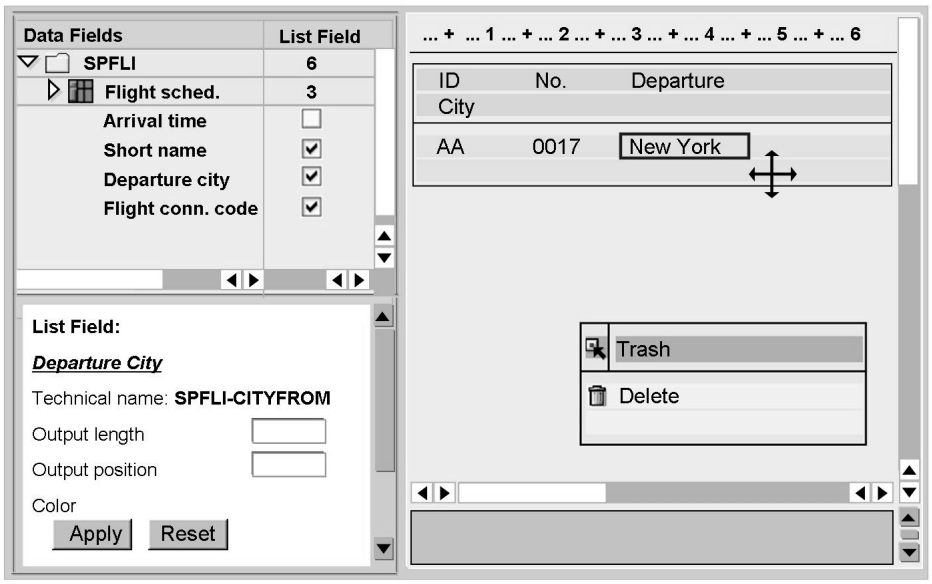


Figure 32: Constructing Basic Lists

Selecting a field in the upper left window automatically adds that field to the list (appended at the end of the current line). The individual fields are represented by field values. Sample data records are read from the source. If this is not possible, field values are simulated. The structure of the layout determines the structure of the subsequent list - that is, it contains the order of the fields, the headers, the colors, totals lines, and so on. To display the list structure for multiline hierarchy lists, several sample records are read.

You can also use **Drag&Drop** to edit the list. Example: You want to change the **field sequence**. To do this, highlight the field you want to move with the mouse (a border appears), click and hold the left mouse button (the cursor changes), drag the field to the desired location, and release the mouse button. You delete a field in the same way by dragging it to the trash.

To change a **field output length**, select the field and drag the left or right-hand border to the required position.

You can also change the output position and output length with entries in the lower left window. Choose *Apply* to apply your values to the list structure.

There are also **tools** available for designing the list. You can change the arrangement of the tools with Drag&Drop (by clicking on the tool title). You change the size in the same way as the output length for fields.

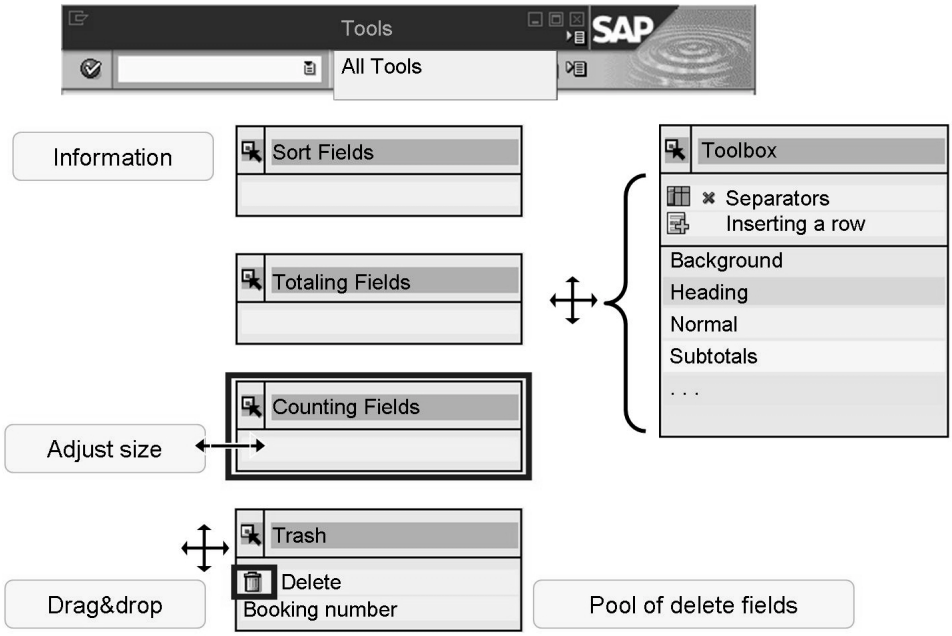


Figure 33: Query Painter Tools

In the upper right area of the Query Painter, you can hide tools, for example, using the menu *Tools → All Tools On/Off*.

To get **information** about the tools, click on the icon in the upper left-hand corner.

You can delete fields that you have had in your list by using **Drag&Drop** to move them to the trash. You can also move fields to the totaling, counting or sort fields using Drag&Drop. To remove the fields again, drag the trash can icon onto them.

You can drag list fields that you put in the trash can earlier back to the list from the trash.

You use the toolbox for column separators, line breaks and field colors:

- To insert **separators** between fields in a list row, drag the table icon from the toolbox to the lines. To delete the separator, drag the red cross onto it.
- If you drag the plus icon from the toolbox onto a **line**, a new line is added beneath it. If you want to delete a line, the line cannot contain any fields. Drag the trash can onto the line that you want to delete.
- To highlight individual fields in color, drag the required **color** from the toolbox onto the field.

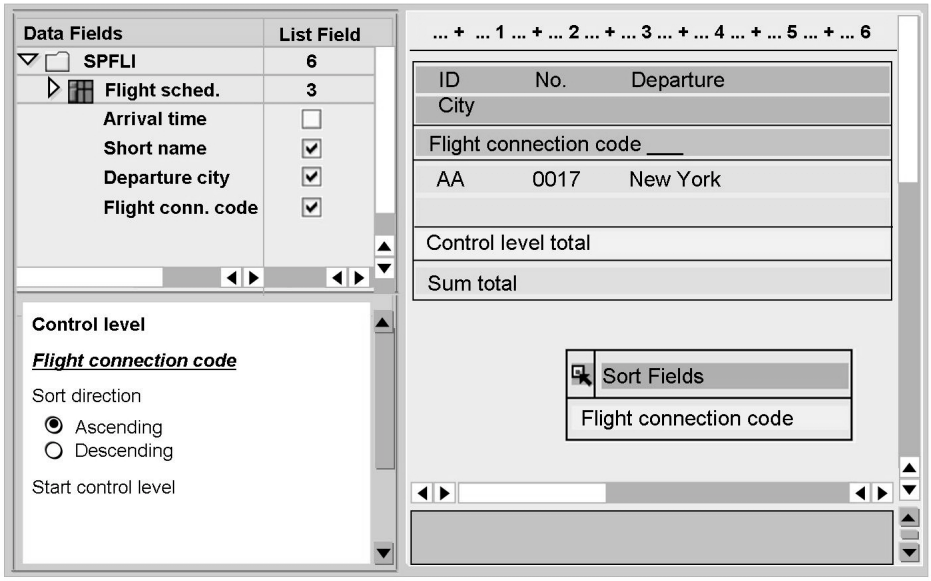


Figure 34: Basic List: Control Level Processing

If you choose a standard list (ABAP list) as the type of output, you can determine the control level in the Query Painter. To do this, you have to determine the **sort fields**. You can define the sort sequence in either ascending or descending order separately for each field. To create a sort field, drag a field from the list to the *Sort* tool (Drag&Drop).

You can define control levels with or without a total at the end of the group level (subtotal). You can change the text accompanying the subtotals. Put the cursor in the field to edit the text.

If you are totaling using a field, the total is displayed in the same column as the field, which means it has the same output length. Accordingly, the output length may be too short and result in a text overflow (an asterisk appears in the first position of the value). To prevent totals overflowing, you can simply increase the output length of the field you wish to total.

You can output blank lines or force a page break before the control levels are output. You do this in the lower left area.

You can show and change header and footer texts for control levels.

The system automatically distributes currencies for totals in foreign currencies.

You can create control levels not only in the Query Painter, but also when executing in the SAP List Viewer. This is more flexible since you can create as many sort orders and intermediate stages as you wish for each query definition (and save them as a SAP List Viewer layout, if required).

Figure 35: Query Painter: Headers and Footers

You can define header and footer lines for the *ABAP list* output format. To do this, click on the relevant pushbutton.

If you require more than one line, drag the plus icon to the existing line, as usual.

To delete a line, drag the trash can onto it.

The following variables are available in headers and footers:

Variables in Headers and Footers

Field name	Description
&%NAME	User name
&%DATE	Date
&%TIME	Time
&%PAGE	Page number

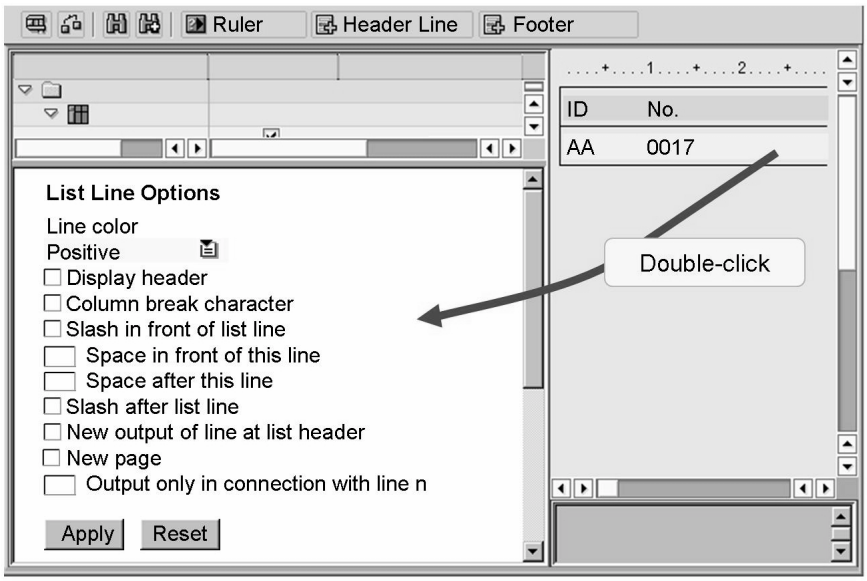


Figure 36: Query Painter: Line Options

If you double-click on a space in a line in the Query Painter that does not already contain a field, the **line options** are displayed in the lower left screen area.

You reach the **list options** (width and borders) by clicking on the ruler.

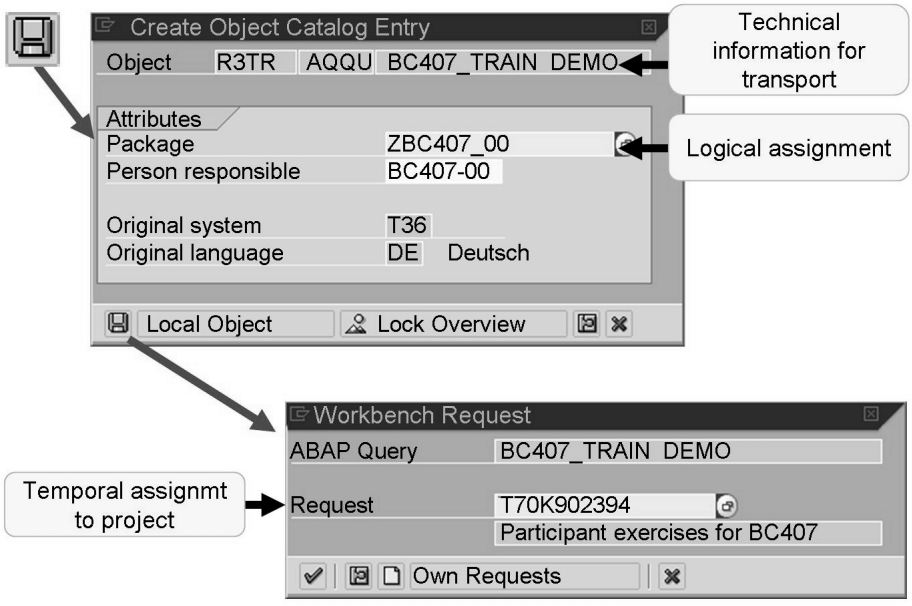


Figure 37: Saving Queries in the Global Query Area

The first time that you save a query object in the global query area, you are asked to specify a **package**. This is a **logical collection** of programming objects, for example, queries, other programs or database table definitions. You can, for example, assign all programming objects that relate to flight reservations to the same package.

Assigning a package is a prerequisite for a transport, for example, from a development system to a test or a productive system. If you choose *Local Object* (which corresponds to the pseudo-package $TMP), this prerequisite is not fulfilled.

You create packages in transaction SE80. They have to be in the customer namespace (begin with Z or Y).

As well as the logical collection of query objects, you also require a temporal assignment to a **workbench request**. All objects in an workbench request are transported from one system to the next at the same time.

Every project team member has a task. This task is a subcategory of a **workbench request**.

Project leaders create the workbench requests and their corresponding tasks in the Transport Organizer (transaction SE09).

Each programming object has exactly one package but usually belongs to different workbench requests (at different points in time).

Query objects in the standard area are not connected automatically to the transport system. Refer to the *RSAQR3TR* report documentation.

Statistics and Ranked Lists

Existing Data Records

Airline	Code	Date	Booking	Weight	Unit
AA	0017	24.12.2007	1	12	lbs
AA	0017	24.12.2007	2	19	lbs
AA	0017	24.12.2007	3	8	lbs
AA	0017	25.12.2007	1	13	lbs
AA	0017	25.12.2007	2	18	lbs
LH	0400	06.11.2007	1	8	kg
LH	0400	06.11.2007	2	17	kg

Statistics:

Airline	Code	Date	Weight
AA	0017	24.12.2007	20
AA	0017	25.12.2007	14
LH	0400	06.11.2007	25

Airline	Code	Weight
AA	0017	34
LH	0400	25

- Compressed data records
- Totaling of all numerical fields Conversion to comparable unit
- Up to 9 statistics for each query

Figure 38: Statistics

Statistics return data records in compressed form. They group several data records together in the same key and create **one** information line for the key. All non-numerical fields automatically belong to the key. If the output format is the *Standard List*, the key fields are highlighted in color.

Column totals are calculated for all numerical fields.

For each query, you can define up to nine statistics separately or as an addition to basic lists or ranked lists.

If you work with different currency or quantity fields within statistics, you have to enter a reference currency or a reference unit for each field, so that the system can convert it into that currency or unit. If you choose the *List* output format, all conversions are displayed at the end. In the event of an error, the system logs any conversions that did not take place. In addition, the system highlights the currency or quantity fields affected within the statistics.

You can also define subtotal lines to appear within statistics. If you compress the statistics, the system displays only the subtotal lines and the grand total.

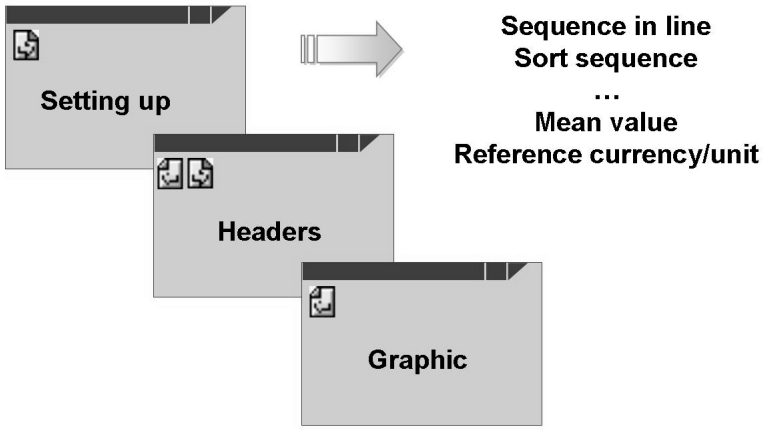


Figure 39: Creating Statistics: Sequence

You can only create statistics when you have selected field groups and their required fields.

You create statistics in three steps:

- Define the list layout, for example, the field sequence or the intermediate stages.
- Define the page header and footer.
- Then determine how a graphic that the user creates interactively is to appear, for example, as a bar chart.

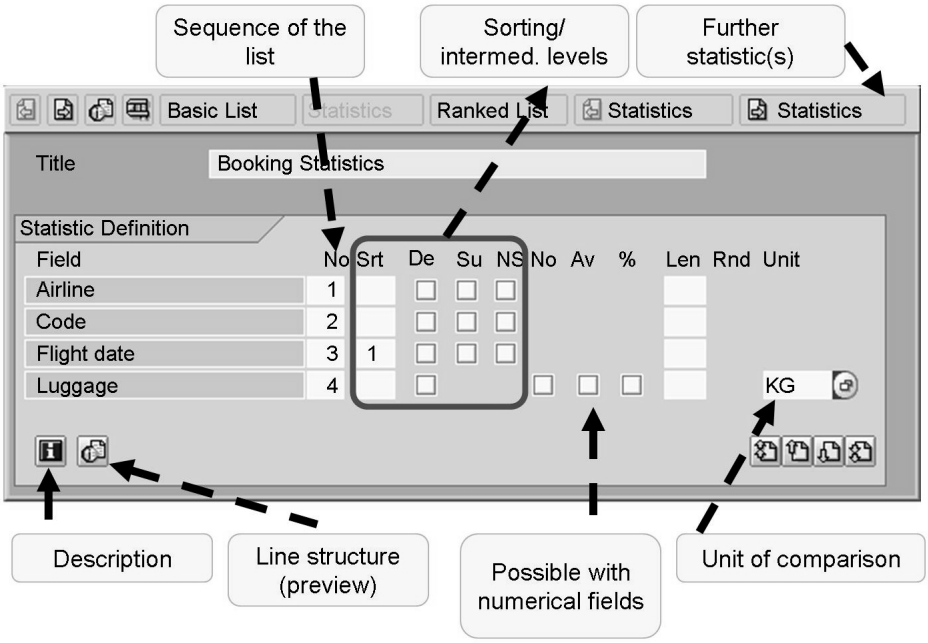


Figure 40: Creating Statistics

Statistics are defined in the same way as ranked lists - not graphically, but alpha-numerically.

Enter a unique **title** – each SAP Query can have up to nine statistics.

Then enter at least the **sequence** of the required fields. By pressing the enter key, the lines are arranged on the screen in the correct sequence for orientation purposes. If you click on the Line Layout button, a preview is shown.

Fields are displayed in their default **length**. You can change this to a different length if you wish.

You can sort a statistic by the values of the key column (the value of the non-numerical fields). You determine the sequence of the sort criteria by entering a numeral. If you want the **sorting** to be in descending order, you have to check the relevant box.

You can define **intermediate levels** for non-numerical fields by which you want to sort.

For numerical fields, you can display the **mean value**, **percentage distribution** and **number** of data records read to the list.

Airline	Code	Date	Weight	Free
AA	0017	25.12.2007	140	143
AA	0017	24.12.2007	200	96
LH	0400	06.11.2007	250	97

Airline	Code	Date	Weight	Free
AA	0017	25.12.2007	140	143
LH	0400	06.11.2007	250	97
AA	0017	24.12.2007	200	96

As with statistics, but:
- Sorting by a numerical criterion
- Fixed number of list lines

Figure 41: Ranked List

Ranked lists are special types of statistics. Data records that belong to a key are grouped together in one line in the list and the numerical values belonging are totaled. However, they are always sorted based on **one** numerical value. This value is referred to as the **ranked list criterion**. The system outputs only a certain number of places. Ranked lists are particularly appropriate for questions such as, "Which are the ten flight connections with the highest revenue?"

For each query, you can define up to nine ranked lists separately or as an addition to basic lists or statistics.

The rules for conversions of currency and quantity fields also apply to ranked lists.

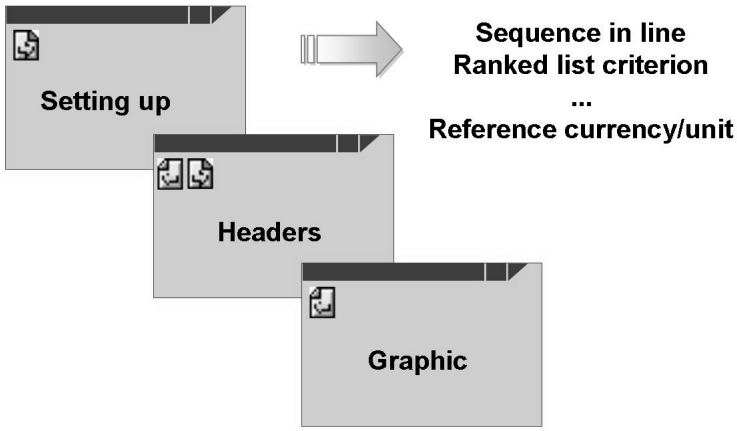


Figure 42: Creating Ranked Lists

You create ranked lists in three steps, just like with statistics.

- Define the list layout, for example, the field sequence or the intermediate stages. You also have to determine the ranked list criterion here.
- Define the page header and footer.
- Then determine how a graphic that the user creates interactively is to appear, for example, as a bar chart.

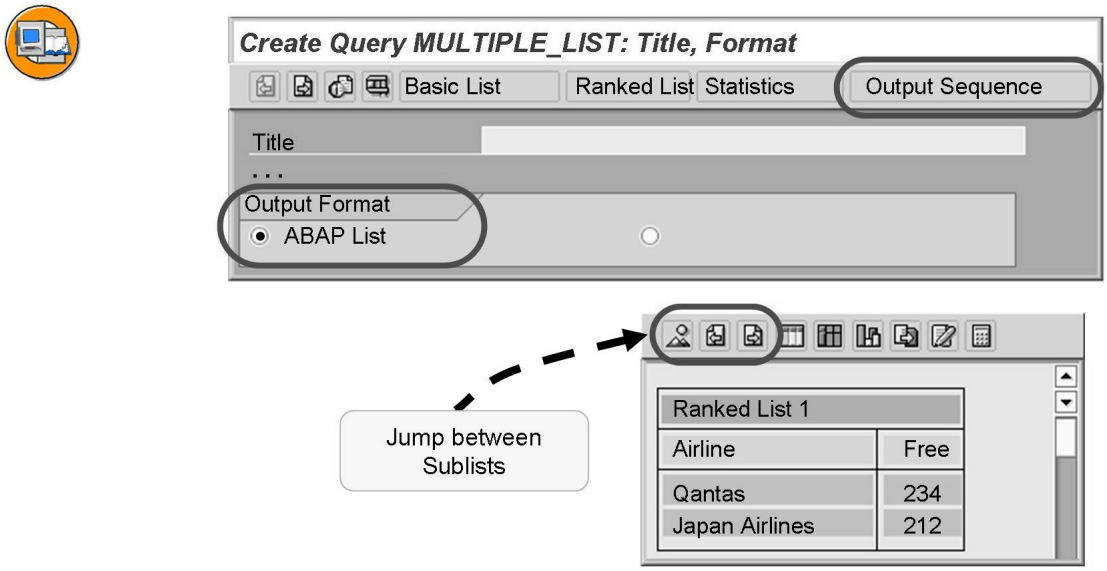


Figure 43: Sublist Combinations

If you want to display several lists within a query definition, you have to choose *ABAP List* as the output format. You can set the output sequence using the pushbutton on the same screen in which you also define the title, list width, and so on.

When you execute the query, all lists are displayed one below the other in the sequence you specified. You reach the lists either by using the scroll bar or by choosing *List Overview, Previous Section* and *Next Section*.

You can use the interactive functions with sublist combinations. The list is returned on which the cursor is placed.

Further Processing

SAP Query: Further Processing

Figure 44: Further Processing: Interactive List Display Functions

List Overview: If your list consists of several partial lists, for example a basic list, two statistical lists and a ranked list, you can display the partial lists individually. The sublists can also be printed separately.

Selections: Information about entries on the selection screen.

Drilldown functions: for expanding and collapsing the list.

Totaling: Totals for numeric fields.

Report/report interface (RRI): You can call queries from this interface (receiver) and call up other reports (sender).

Table display: The list is displayed as a table control or using the SAP List Viewer. You can also display lists with multiple lines. The different lines are summarized in one line.

File storage, private storage: For saving data as a file on the presentation server or in a private store.

Word processing and table calculation: Transfer to MS Word or Excel, for example.

Graphic: Displays list information using SAP Presentation Graphics.

ABC analysis, EIS (Executive Information System).

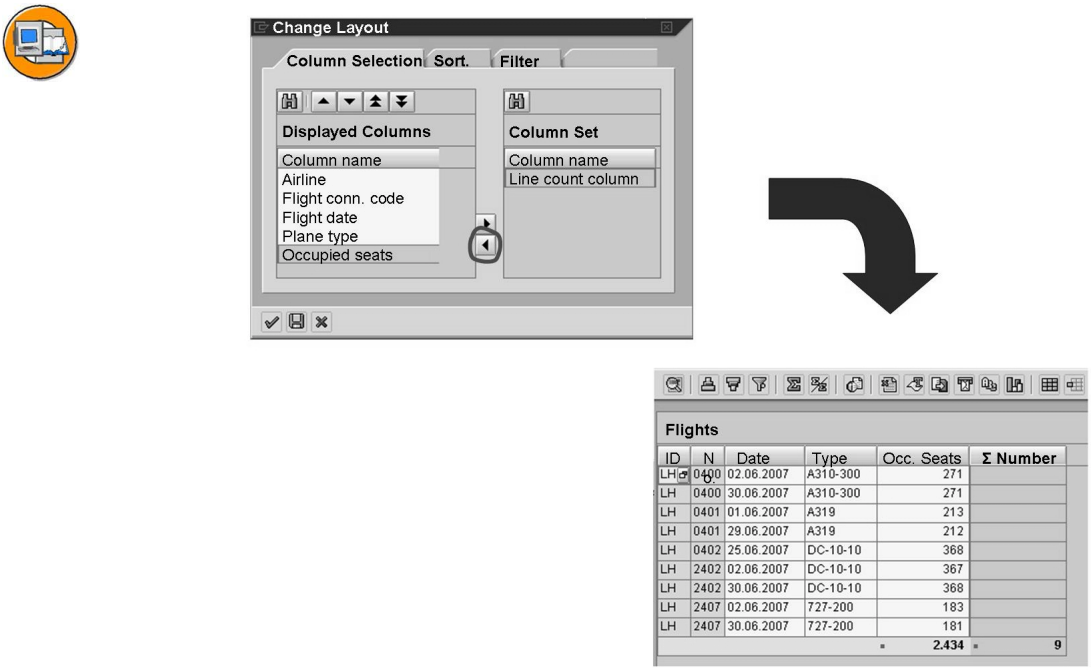


Figure 45: Display Counter Field in SAP List Viewer

As of SAP Web Application Server 6.20, you can display the number of issued records (counter field) in SAP List Viewer. The *column for number of lines* is always available in the column set of the SAP List Viewer, however it is always hidden at the start of the display.

You can display this column by choosing the *Show* function.

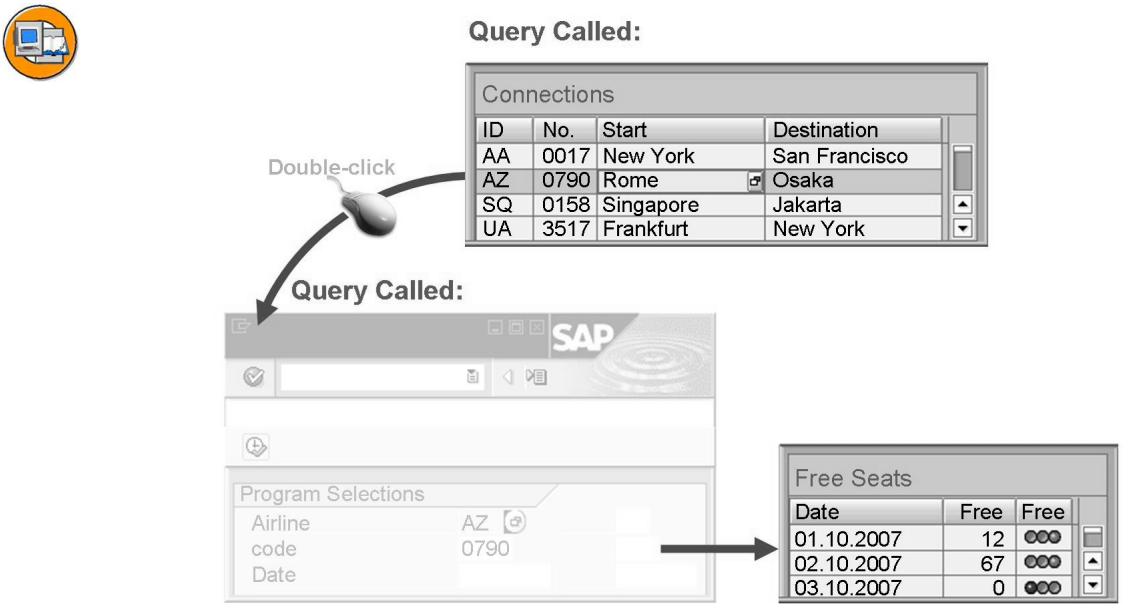


Figure 46: Report/Report Interface (RRI)

Within the SAP system, you can call up different reports from different applications using the report/report interface (RRI). The term *report* is a superordinate term for queries, ABAP reports and transactions.

The user double-clicks on a line in a query. A second query is called, the selection screen (not visible to the user) is filled with matching data from the first query and the list is displayed immediately. More precisely, all fields in the selection screen of the query that was called and all fields of the lines that were executed with a double-click are transferred to the selection screen of the second query. The fields of sender and receiver queries are assigned using the technical properties (data element equivalence). For more details about data transfer, see SAP note 50629.

You can connect several reports to a query. When double-clicking on a list (output format ABAP list or SAP List Viewer), the user is asked which report is to be called up.

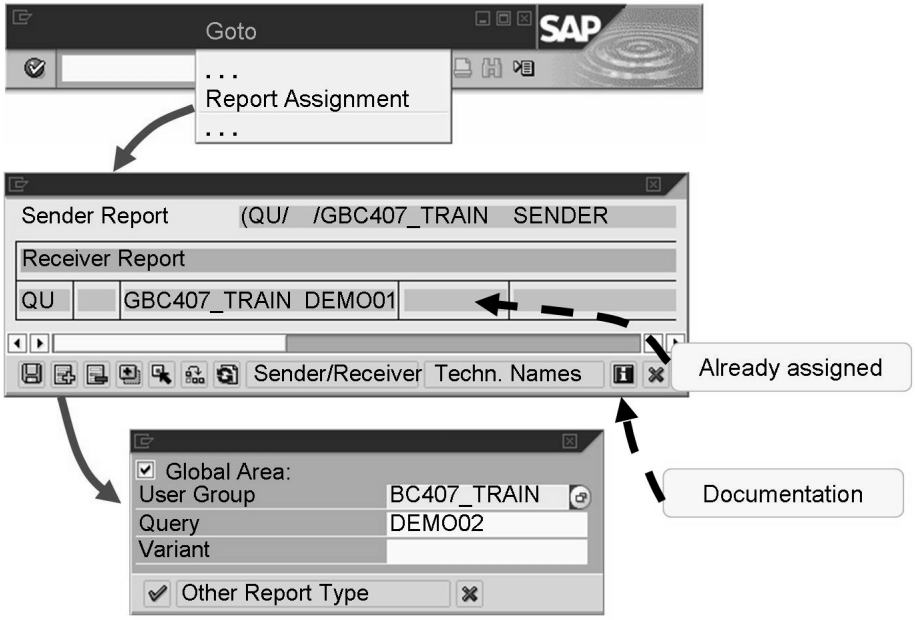


Figure 47: Report/Report Interface: Assigning Reports

To add a report to a SAP Query, choose *Goto → Report Assignment*. On the next screen, you get an overview of the reports that have already been assigned. You use the plus pushbutton to get to the next input template, in which you enter the query that you want to add. Press the enter key to go back to the overview list. Then choose *Save*.

You reach documentation on adding, deleting, moving, and so on, by clicking on the help button.

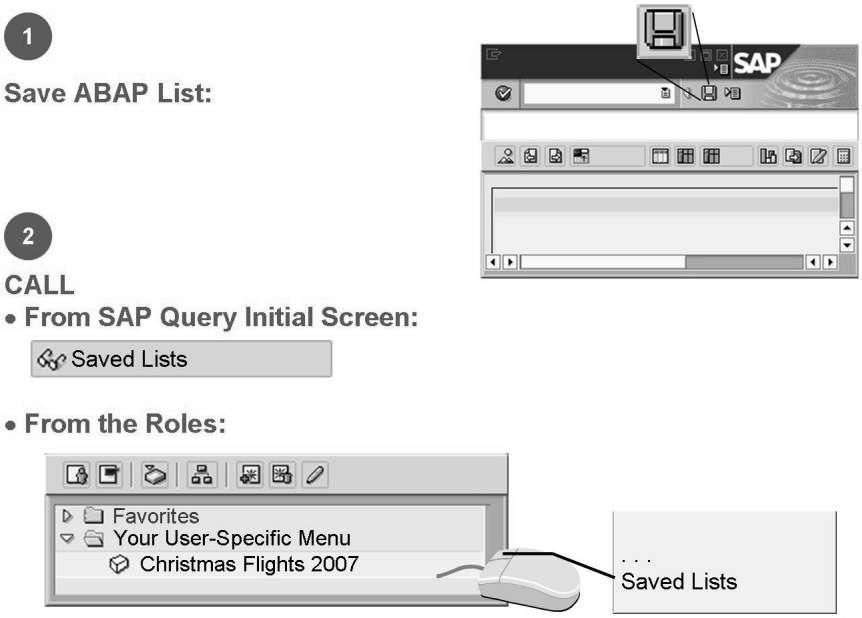


Figure 48: Saving Lists

You can save ABAP lists created with a query together with the query so that you can display these saved lists again at a later date. To **save** from the list display, choose *List* → *Save* or choose the Save pushbutton. Along with the name that you entered, the date and time are saved automatically.

Subsequent display of a saved list does not require database access to retrieve data. Such a display is therefore much quicker than restructuring the list by executing the query again.

Saving a list stores the list itself and additional information. This means that you are able to use most of the interactive functions (for example, the SAP List Viewer, download, but not the report/report interface) in the saved list.

You call the lists from the initial SAP Query screen (transaction SQ01) by following the menu path *Goto* → *Saved Lists* or by choosing the relevant pushbutton.

If a query is included in a **role**, all saved lists are automatically transferred to the role and can be displayed from there (using the context menu). To include a query in a role, see the documentation for transaction PFCG.

In the saved list, you can see the entries made in the selection screen when the list was executed. Choose *Selections*.

Exercise 3: Basic Lists and Local Fields

Exercise Objectives

After completing this exercise, you will be able to:

- Create a multiline (hierarchical), basic list with local fields for a query

Business Example

Your task: Create an SAP Query that shows flight connections and free seats.

Model solution: BC407_S_01 in the global query area, user group BC407_LEARN

Task 1:

Create the query

1. Check whether you are in the **global query area** and go there if necessary.

   Create **Query BC407##_01** for user group **BC407_LEARN**. ## stands for your (two-digit) group number.

   As the data basis, choose **InfoSet /SAPQUERY/BC407_01**.

   When you save your query (after step 1-2 at the earliest), enter **package ZBC407_##**.

Task 2:

Maintain the query properties

1. Maintain the title and comments for the query and set the column width to 90 columns. Choose *ABAP List* as the output format.

Task 3:

Select the field groups

1. You need the first two field groups *Flight Schedule* and *Flight*.

Task 4:

Create local fields

1. Assign short names for the fields *Occupied seats* (*OCC*) and *Maximum Capacity* (MAX).

Continued on next page

2. In field group *Flights*, create a local field, *Free Seats* with short name *FREE* and header *Free*. The field should have the same attributes as the *Maximum Capacity* field.
3. Calculate the number of free seats for this field as the difference between the maximum capacity and the number of occupied seats.

Task 5:

Create a multiline basic list in the Query Painter (layout mode).

1. **Line 1** *Airline Code, text: Airline* (this is the airline name that is determined automatically)*, flight connection number.*

   Line 2: *Departure and arrival cities, departure and arrival times, distance.*

   Line 3: *Flight date, occupied seats, free seats, maximum capacity, airfare.*

Task 6:

Structure the list

1. Output the list with frames.
2. Select the following colors:

   Line 1: Color *Heading (intensified)* with an empty line before the line

   Line 2: Color *Heading*

   Line 3: Color *Normal*

   Field *Free seats*: Color *Positive*
3. Adjust the output lengths for the following fields: *occupied seats, free seats* and *maximum number of seats* to 8 characters and the *airfare* (including currency) to 20 characters.
4. Change the heading of the field *Maximum Capacity* to Max.

Task 7:

Save and test the query

1. If you have not yet saved the query, save it now (in package ZBC407_##).

Solution 3: Basic Lists and Local Fields

Task 1:

Create the query

1. Check whether you are in the **global query area** and go there if necessary.

   Create **Query BC407##_01** for user group **BC407_LEARN**. ## stands for your (two-digit) group number.

   As the data basis, choose **InfoSet /SAPQUERY/BC407_01**.

   When you save your query (after step 1-2 at the earliest), enter **package ZBC407_##**.

   a) Start SAP Query in transaction SQ01. Make sure that you are in the global query area: This was displayed in the initial screen. Otherwise, follow the menu path *Environment → Query Areas* to go to the global work area.

   Set the user group to BC407_LEARN: Menu path *Edit → Other User Group*. Create a query named BC407##_01 (## stands for the group number).

   When you click on Create, you are asked to enter an InfoSet (the data basis) in another input template. Double-click on /SAPQUERY/BC407_01.

Task 2:

Maintain the query properties

1. Maintain the title and comments for the query and set the column width to 90 columns. Choose *ABAP List* as the output format.

   a) Maintain the title (mandatory field!) and list width in the initial screen.

Task 3:

Select the field groups

1. You need the first two field groups *Flight Schedule* and *Flight*.

   a) Choose *Next Screen* or follow the menu path *Goto → Next Screen*. You reach the Field Groups screen. Mark the top two *field groups*.

Continued on next page

Task 4:

Create local fields

1. Assign short names for the fields *Occupied seats* (*OCC*) and *Maximum Capacity* (MAX).

   a) Choose *Next Screen* or follow the menu path *Goto → Next Screen*. You reach the *Fields* screen.

   Activate the short names under menu *Edit → Short names* and enter short names for all specified fields.

2. In field group *Flights*, create a local field, *Free Seats* with short name *FREE* and header *Free*. The field should have the same attributes as the *Maximum Capacity* field.

   a) Under *Edit → Local Field*, create the required field. The field has to belong to the function group *Flight*. Enter the short name *MAX* in the field *Same attributes as field*.

3. Calculate the number of free seats for this field as the difference between the maximum capacity and the number of occupied seats.

   a) Define `MAX - OCC` as the calculation formula. Save the local field by pressing the enter key.

Task 5:

Create a multiline basic list in the Query Painter (layout mode).

1. **Line 1** *Airline Code, text: Airline* (this is the airline name that is determined automatically)*, flight connection number.*

   Line 2: *Departure and arrival cities, departure and arrival times, distance.*

   Line 3: *Flight date, occupied seats, free seats, maximum capacity, airfare.*

   a) Now choose the *Basic list* pushbutton. This starts the Query Painter.

   You can see the data fields available in the upper left area of the Query Painter. For the fields specified in the task, mark the *List Fields* checkbox. Make sure that you set up the list in the order in which you selected the fields. Otherwise, you have to sort the fields in the upper-right area again. Click on the field that you want to move and drag it to the required position.

   To insert a new line, click on the plus icon in the toolbox and drag it to an existing line.

   You can find the field *Text: Airline Code* under *Additional fields*.

Continued on next page

Task 6:

Structure the list

1. Output the list with frames.

   a) Click on the ruler. In the lower left area, you determine that the list is to have a border. Do not forget to choose *Apply*.

2. Select the following colors:

   Line 1: Color *Heading (intensified)* with an empty line before the line

   Line 2: Color *Heading*

   Line 3: Color *Normal*

   Field *Free seats*: Color *Positive*

   a) To change the colors of the lines, double-click on an area in the current line that does not contain a field. You then set the required color in the lower left area from the dropdown list. Do not forget to choose *Apply*.

3. Adjust the output lengths for the following fields: *occupied seats, free seats* and *maximum number of seats* to 8 characters and the *airfare* (including currency) to 20 characters.

   a) To change the attributes of a field, select the field. This displays the field and its details in the lower left window, and you can now change its attributes. Press *Apply* to activate your changes.

4. Change the heading of the field *Maximum Capacity* to Max.

   a) You change the header text of a field by double-clicking on it. This makes the text field ready for input.

Task 7:

Save and test the query

1. If you have not yet saved the query, save it now (in package ZBC407_##).

   a) When you save the query, enter ZBC407_## as the package. ## stands for your group number. In the next screen, click on *Own Requests* and select you request by double clicking on it.

      To test it, choose the Execute pushbutton (shortcut: F8).

Exercise 4: Basic Lists and Local Fields (optional)

Exercise Objectives

After completing this exercise, you will be able to:

- Define local fields as icons
- Sort a list

Business Example

Your task:

Copy your SAP Query, sort it and enhance it with a traffic-light icon.

Model solution

BC407_S_01_OPT in the global query area, user group BC407_LEARN

Task 1:

Copy the query

1. In the **global query area** for user group **BC407_LEARN**, copy your **Query BC407##_01** to **BC407##_01_OPT**. ## stands for your (two-digit) group number.

   Save the new query in **package ZBC407_##**.

Task 2:

Create local fields

1. In field group *Flights*, create a local field, *Free Seats* with short name FREE_ICON and header *Free Seats (graphically)*. Type the field as an icon.
2. Determine the icon for the field *FREE_ICON* using a complex calculation:

   ICON_GREEN_LIGHT (green light) when there are more than 100 free seats,

   ICON_GREEN_LIGHT (yellow light), when there are more than 10 free seats,

   ICON_RED_LIGHT (red light) otherwise.
3. Display the icon after the *FREE* field in the basic list.

Continued on next page

Task 3:

Determine sorting

1. Sort the list in descending order by *Airline Code* and *Flight Connection Number*. Do not include any subtotals.

   Test your query with airline code *SQ*.

Internal Use SAP Partner Only

Solution 4: Basic Lists and Local Fields (optional)

Task 1:

Copy the query

1. In the **global query area** for user group **BC407_LEARN**, copy your **Query BC407##_01** to **BC407##_01_OPT**. ## stands for your (two-digit) group number.

   Save the new query in **package ZBC407_##**.

   a) Start SAP Query in transaction SQ01. If you have not logged off since the last exercise, your query area and user group should still be set correctly.

   In the table control, select your query BC407##_01 and choose *Query → Copy*. On the next screen, enter the name of the copy for *To*: BC407_##_01_OPT. Then enter the package ZBC407_## and choose your task on the screen that follows.

Task 2:

Create local fields

1. In field group *Flights*, create a local field, *Free Seats* with short name FREE_ICON and header *Free Seats (graphically)*. Type the field as an icon.

   a) Branch to the maintenance of the local fields (see above). Create a local field with the name *FREE_ICON* and identify it as an *Icon* in the properties.

2. Determine the icon for the field *FREE_ICON* using a complex calculation:

   ICON_GREEN_LIGHT (green light) when there are more than 100 free seats,

   ICON_GREEN_LIGHT (yellow light), when there are more than 10 free seats,

   ICON_RED_LIGHT (red light) otherwise.

   a) Press the *Complex calculations* pushbutton. Enter `FREE > 100` under the first condition and enter *ICON_GREEN_LIGHT* under the corresponding formula. Enter `FREE > 10` under the second condition and enter *ICON_YELLOW_LIGHT* under the corresponding formula. For *Otherwise* enter *ICON_RED_LIGHT*.

Continued on next page

3. Display the icon after the *FREE* field in the basic list.

   a) In the Query Painter, mark the *List Fields* checkbox for the newly-defined field and position the field using Drag&Drop.

Task 3:

Determine sorting

1. Sort the list in descending order by *Airline Code* and *Flight Connection Number*. Do not include any subtotals.

   Test your query with airline code *SQ*.

   a) In the Query Painter, click on the *Airline Code* field and drag it to the *Sort Fields* tool.

      The subtotals are set automatically for sort levels for all numerical fields. To delete them, drag the trash can onto the heading of the subtotals.

      Proceed in the same way with field *Flight Connection Number*.

Exercise 5: Statistics

Exercise Objectives

After completing this exercise, you will be able to:

- Create a statistic

Business Example

Your task:

Create a statistic the gives you information about the luggage from flights.

Model Solution:

BC407_S_02 in the global query area, user group BC407_LEARN.

Task 1:

Create the query

1. In the **global query area**, create **query BC407##_02** for the user group **BC407_LEARN**. ## stands for your (two-digit) group number.

   As the data basis, choose the **InfoSet /SAPQUERY/BC407_05**.

   When you save your query (after step 1-2 at the earliest), enter **package ZBC407_##**.

Task 2:

Maintain the query properties

1. Maintain the title and the comments for the query. Choose *ABAP* List as the output format.

Task 3:

Select the field groups

1. You need all three field groups.

Task 4:

Field selection

1. You need *Airline Code, Flight Connection Number, Flight Date* and *Weight of Luggage*.

Continued on next page

Task 5:

Define the statistics

1. Choose *Airline Code, Flight Connection Number, Flight Date* and *Weight of Luggage* for the statistic, along with the mean value and the amount of luggage.

   Sort the statistic in descending order by luggage. Enter KG (kilogram) as the comparative unit.

   When you are testing, enter 10000 for *Number of data records to be read*.

Solution 5: Statistics

Task 1:

Create the query

1. In the **global query area**, create **query BC407##_02** for the user group **BC407_LEARN**. ## stands for your (two-digit) group number.

   As the data basis, choose the **InfoSet /SAPQUERY/BC407_05**.

   When you save your query (after step 1-2 at the earliest), enter **package ZBC407_##**.

   a) See the previous task.

Task 2:

Maintain the query properties

1. Maintain the title and the comments for the query. Choose *ABAP* List as the output format.

   a) See the previous exercise.

Task 3:

Select the field groups

1. You need all three field groups.

   a) See the previous exercise.

Task 4:

Field selection

1. You need *Airline Code, Flight Connection Number, Flight Date* and *Weight of Luggage*.

   a) In the *Fields* screen, mark the required fields.

Task 5:

Define the statistics

1. Choose *Airline Code, Flight Connection Number, Flight Date* and *Weight of Luggage* for the statistic, along with the mean value and the amount of luggage.

   Sort the statistic in descending order by luggage. Enter KG (kilogram) as the comparative unit.

Continued on next page

When you are testing, enter 10000 for *Number of data records to be read*.

a) Choose Define *Statistic*. In the input field *No.*, enter the following numbers:

Airline Code	1
Flight Connection Number	2
Flight Date	3
Weight of Luggage	4

For luggage, mark the fields *No.* and *Av* as well.

Sort the statistics in descending order by entering 1 for *Luggage* in the input field *Srt* and selecting *De*.

If you enter only the standard value of 100 for *Number of records to be read*, even though the system reads 100 records from the database, it may only issue one line in the statistics if the first flight already has 100 bookings or more.

Exercise 6: Ranked List (Optional)

Exercise Objectives

After completing this exercise, you will be able to:

- Create a ranked list
- Create a SAP Query with two lists
- Use the report/report interface (RRI)

Business Example

Enhance your SAP Query with a ranked list for reservation revenue and add another query using the RRI.

Model Solution:

BC407_S_02_OPT in the global query area, user group BC407_LEARN

Task 1:

Enhance the query

1. Enhance **query BC407##_02** from the previous exercise. You do not need to make a copy of it.

Task 2:

Field selection

1. In addition to the fields already selected, you need the *price of the booking in local currency*.

Task 3:

Define the ranked list

1. Create a ranked list. Use *Airline Code, Flight Connection Code* and *Price of booking in local currency* in the ranked list.

   You want to display the five connections with the highest revenues in the ranked list. Therefore define *Price of booking in local currency* as ranked list criterion. Choose USD (US dollars) as the reference unit.

Task 4:

Determine the list sequence

1. Specify that you want to display first the ranked list and then the statistic when the query is executed.

Continued on next page

Task 5:

Add a report

1. Add the query *BC407_CONNEX* from the global query area, user group *BC407_LEARN*.

Internal Use SAP Partner Only

Solution 6: Ranked List (Optional)

Task 1:

Enhance the query

1. Enhance **query BC407##_02** from the previous exercise. You do not need to make a copy of it.

   a) Process your solution to the last exercise further.

Task 2:

Field selection

1. In addition to the fields already selected, you need the *price of the booking in local currency*.

   a) In the *Fields* screen, mark the required fields.

Task 3:

Define the ranked list

1. Create a ranked list. Use *Airline Code, Flight Connection Code* and *Price of booking in local currency* in the ranked list.

   You want to display the five connections with the highest revenues in the ranked list. Therefore define *Price of booking in local currency* as ranked list criterion. Choose USD (US dollars) as the reference unit.

   a) Choose *Ranked List*. Determine the sequence of the fields in the same was as with the statistic.

      Define the *Number of Seats* as 10. As the ranked list criterion, mark *Price of booking in local currency* and enter USD as the reference unit in the input field *Unit*.

Task 4:

Determine the list sequence

1. Specify that you want to display first the ranked list and then the statistic when the query is executed.

   a) Go back to the first navigation level using *F3* (green arrow). Click on the *Output Sequence* button and enter the lists you have already defined, 2 and 1.

Continued on next page

Task 5:

Add a report

1. Add the query *BC407_CONNEX* from the global query area, user group *BC407_LEARN*.

   a) Choose *Goto → Report Assignment*

   Click on the plus icon. In the next screen, mark *Global Area* and enter the *User Group* and the *Query*.

   Confirm your entries by pressing the enter key and then Save.

Lesson Summary

You should now be able to:

- Use SAP Query to create basic lists, statistics and ranked lists
- Use the interactive functions of these lists
- Save query lists and call them up again

Unit Summary

You should now be able to:

- Use SAP Query to create basic lists, statistics and ranked lists
- Use the interactive functions of these lists
- Save query lists and call them up again

Unit 4

InfoSet Query

Unit Overview

In this unit, you will learn about all the options for the InfoSet Query.

Unit Objectives

After completing this unit, you will be able to:

- Use InfoSet Query to create basic lists, statistics and ranked lists

Unit Contents

Lesson: InfoSet Query

Lesson Overview

In this lesson, you will learn about all the options of InfoSet Query.

Lesson Objectives

After completing this lesson, you will be able to:

- Use InfoSet Query to create basic lists, statistics and ranked lists

Business Example

You have been asked to use InfoSet Query to create a query.

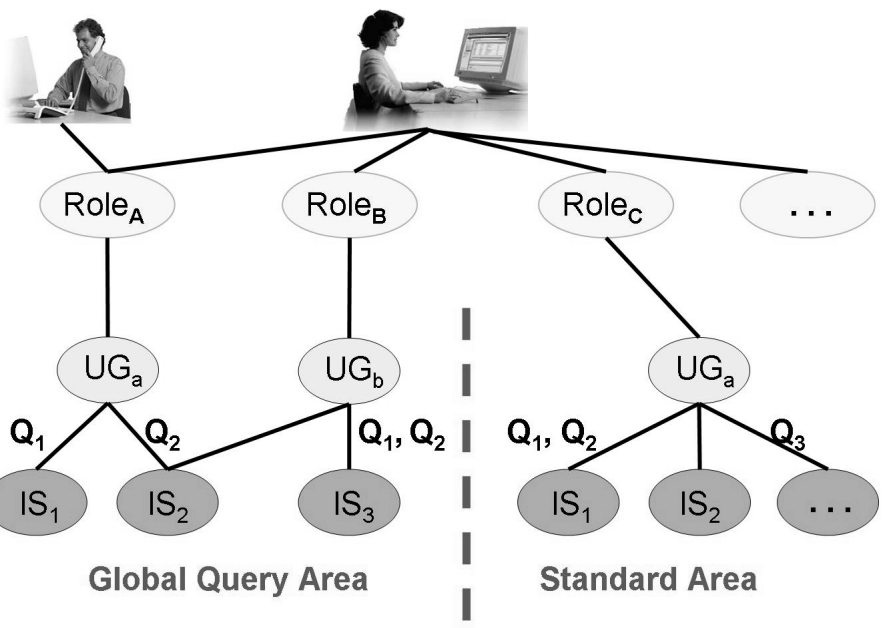


Figure 49: Organizational Environment of the InfoSet Query

In some aspects, the organizational environment of the InfoSet Query is identical to that of the SAP Query.

- The same **InfoSets** form the data basis, with which the queries are defined. They are divided up into **field groups** containing fields.
- The SAP Query and the InfoSet Query **user groups** are identical.
- Both queries have a global **query area** (with client-independent objects and automatic transport connection) and a standard area (with client-dependent objects and without automatic transport connection).

However, users are not assigned to user groups directly, but using a **role**. Each person can have several roles but each role can only have one user group.

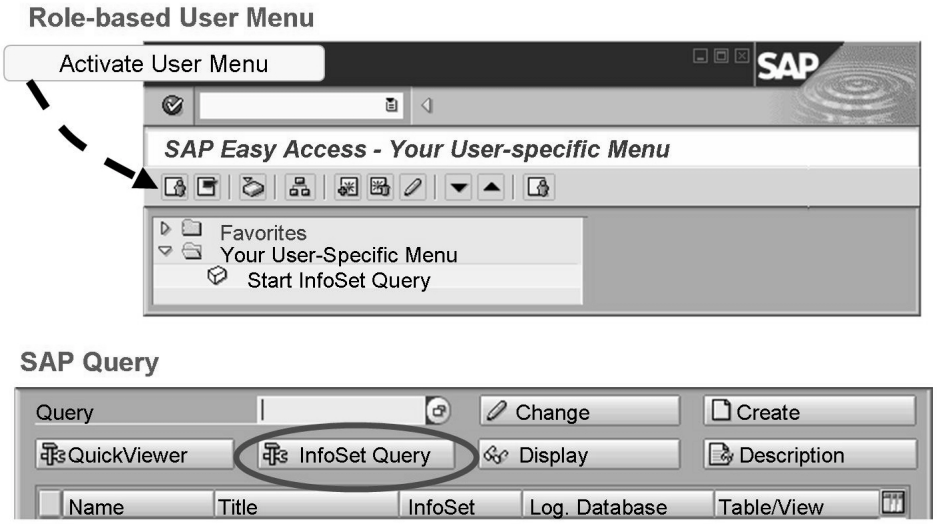


Figure 50: Starting the InfoSet Query

You can start InfoSet Query from role menus, from the SAP menu, or from SAP Query (SQ01).

Depending on your role:

- The query is created in the global query area or the standard area and
- maybe a query is used as a processing template or
- an InfoSet is specified as the data basis.

For details on role administration for the InfoSet Query, see the *User Administration* unit.

When you start SAP Query (SQ01), all queries are displayed, regardless of whether they were created as a SAP Query or an InfoSet Query. If you choose *InfoSet Query*, you can also edit all queries as InfoSet Queries. However, do not edit a SAP Query as an InfoSet Query if it contains the following objects: Several lists, a multiline basic list or local fields. If you use a query with one of these properties in the InfoSet Query, the system displays only single line lists without local fields. If you overwrite the query, these special SAP Query properties are lost.

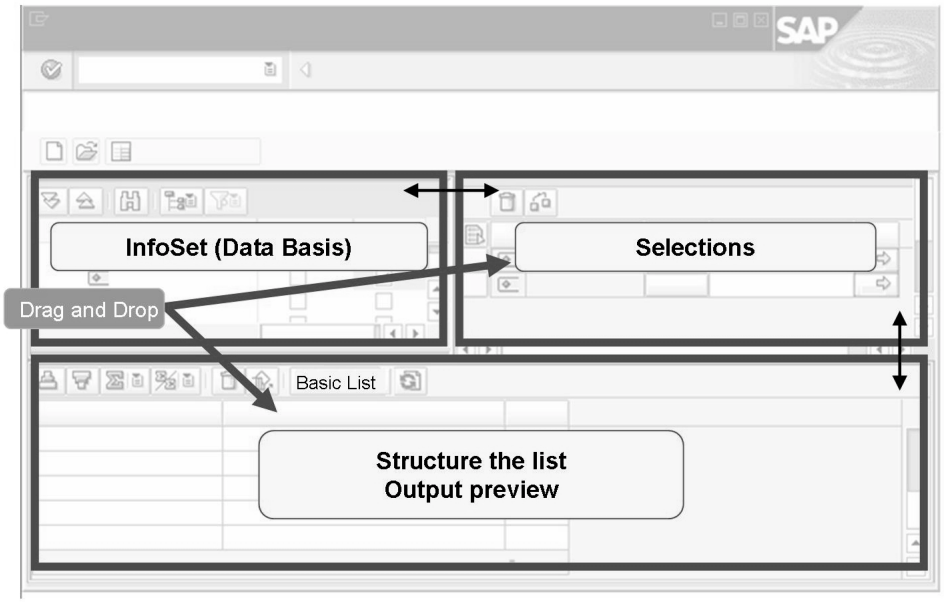


Figure 51: InfoSet Query Areas

In the InfoSet Query, the screen is divided into three areas.

- The fields available from the source InfoSet (the data source) are displayed to the left in tree form. You determine here, using a checkbox, which of these fields you want to display in the selection screen or on the list.
- The selection screen is on the right-hand side of the screen.
- In the lower screen area, the results list is displayed using the SAP List Viewer (list preview). This is where, for example, you determine the column sequence.

You can drag and drop (holding the left mouse button down) fields from the data pool into the list or the selection screen.

You can change the width and height of the three areas by clicking on the relevant separating wall and, keeping the mouse button pressed down, dragging the wall to the required position.

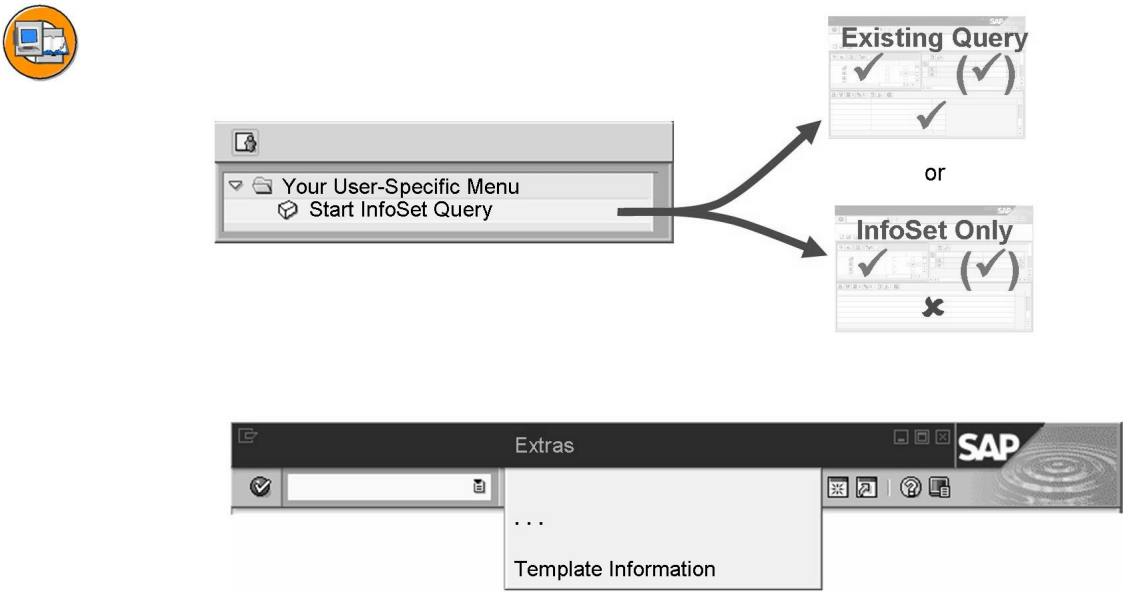


Figure 52: InfoSet Query Templates

What you are offered when you call the InfoSet Query

- The system displays either an existing **query as a template**. (You learn in the *User Administration unit*, which query is used as a template.) The field groups of the source InfoSet are shown along with this query. You can further process the query and save it under a new name, if necessary.
- Or the system simply offers an InfoSet (In the *User Administration* unit, you learn how to define an **InfoSet as a default setting**.) You can create a new InfoSet Query based on this InfoSet.

In the upper left area, the fields groups and fields are always displayed. To find out the name of the InfoSet (and query), choose *Extras* → *Template Information*.

If a standard selection screen is defined in the source InfoSet, then the system displays these selections.

If you have an existing query as a template, example data is displayed in the lower screen area.

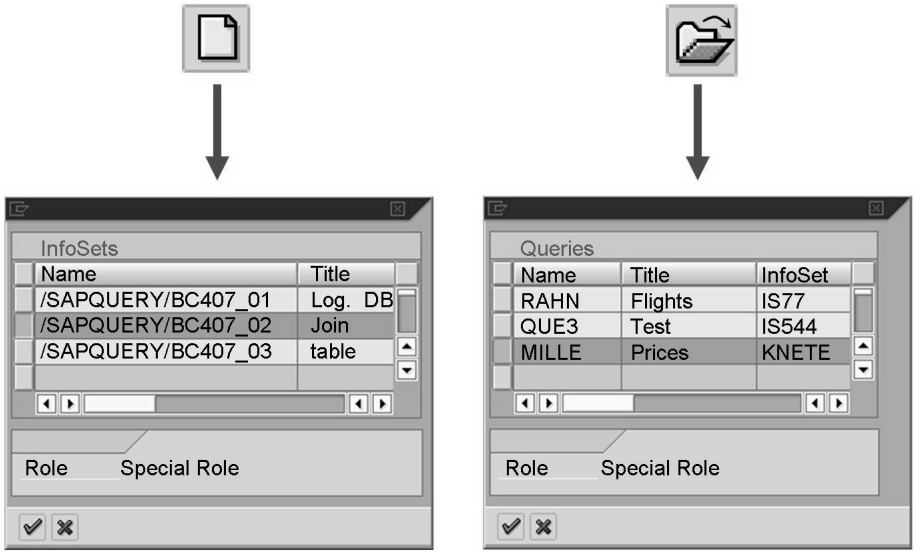


Figure 53: Creating/Opening an InfoSet Query

No matter which template you are offered when you start the InfoSet Query (whether it is an existing query or just an InfoSet), you can **create a new query**. To do this, choose *Create* and select the required InfoSet from your role or user group in the dialog box that appears.

Alternatively, you can **open an existing query**. To do this, choose *Open Query*.

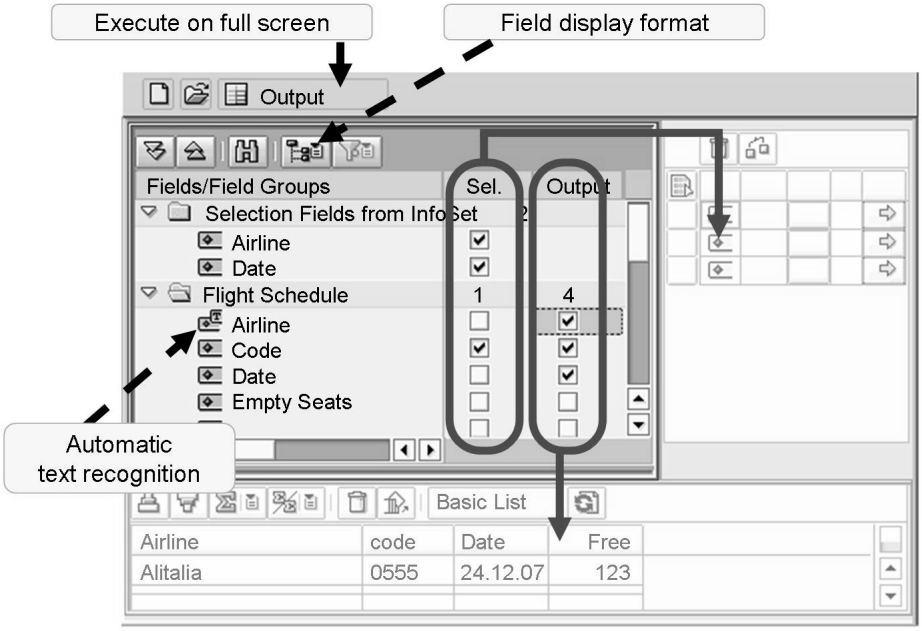


Figure 54: Using the InfoSet

The current **InfoSet** is displayed in the upper left screen area. There are three possible display formats:

- An overview tree of the InfoSet with the field groups belonging to it (standard display). If a selection screen has already been defined in the InfoSet, its fields are displayed at the top.
- A field catalog (ungrouped field list).
- If the InfoSet is based on a logical database, you can also display the structure of the logical database.

By marking the checkboxes, you can determine which fields to include for input in the list or selection screen. Moreover, you can drag & drop the field to the list or selection screen.

Some fields have an automatic **text** recognition, for example, the name of an airline as its identifier. These fields have a "T" in their icon. Use the **context menu** (right mouse button) to specify whether you want only the value of the field, the text or both to be transferred to the list or selection screen. If you make your selection using the checkbox, the default setting is used.

You can display the **technical information** for a field using the context menu.

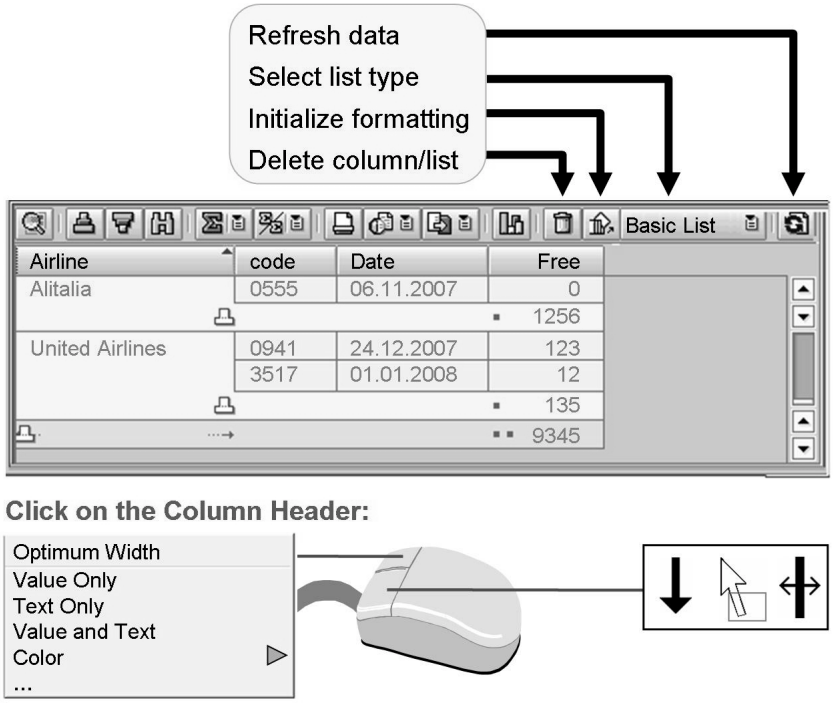


Figure 55: List Structure and List Preview

The lower area of the InfoSet Query serves a dual purpose:

- You can determine the **layout of the list** using the SAP List Viewer functions.
  - Determine the column sequence using *Drag&Drop*.
  - Use the context menu to determine whether fields with automatic text recognition are to have text, value or both displayed.

    Changes made to the layout in the SAP List Viewer such as sorting, column width or column sequence are retained in other output types, such as classical lists.
- The lower area also provides a **preview**: By choosing *Refresh Data*, the data that corresponds to your entries in the selection screen is read from the database. Before you refresh the data for the first time, example data is shown in the preview that does not correspond to actual data records in the database.

You can use the SAP List Viewer in the normal way, by choosing the pushbutton for it. You can, for example, calculate totals, sort the list and download it to your frontend computer in different formats. Along with *Refresh Data*, the following are of particular importance:

- *Delete*: When you highlight a column and choose delete, the column is removed from the list. (Alternatively, you can deactivate the relevant checkbox in the InfoSet display area.) If you do not mark a column, the entire list is deleted.
- *Initialize Formatting*: Sorting or totaling steps are reversed and all columns are displayed in their default width.

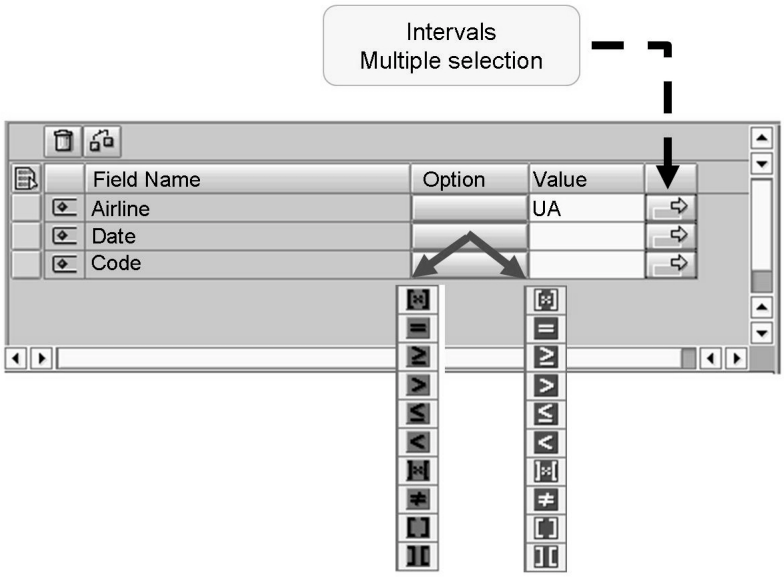


Figure 56: The Selection Screen

In the selection screen in the upper right corner, you can see the fields that you chose in the InfoSet area either by marking the checkboxes or by using drag & drop.

Unlike in the standard selection screen, there is only an input field and no interval for selections (*SELECT-OPTIONS*). If you want to enter and interval, choose *Multiple Selection.* A further input screen appears.

All fields on the selection screen are usually selection options, especially those which have not already been defined in the InfoSet. When you create an InfoSet you can define parameters for it (for entering single values). In this case, the *Option* pushbutton is not active.

If you have not already made selections in the selection screen, when you execute the InfoSet Query, you are asked to restrict the number of data records for performance reasons.

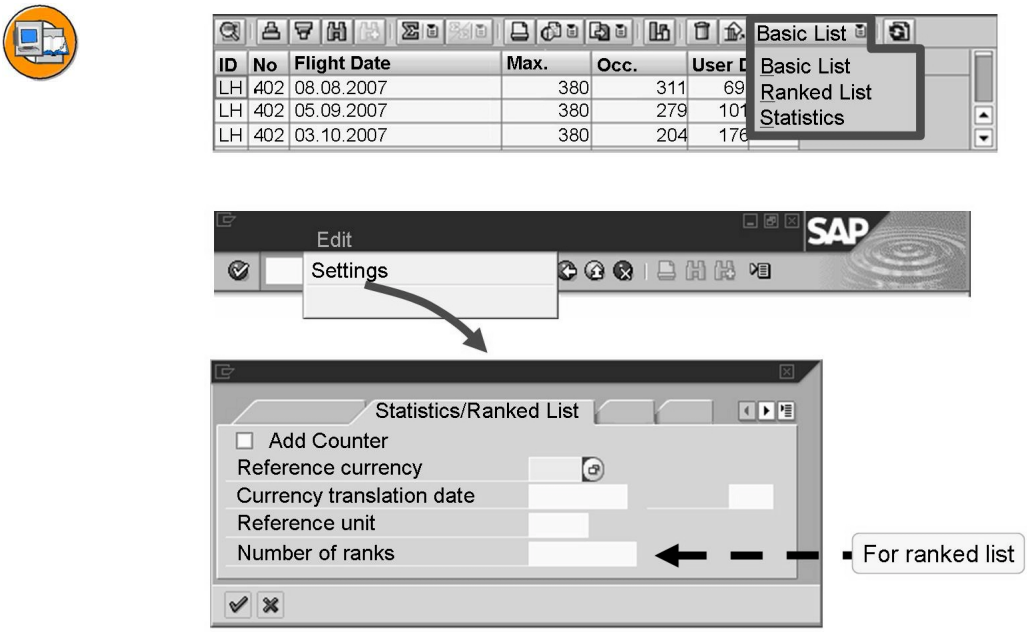


Figure 57: Basic Lists/Ranked Lists/Statistics

You can make settings for the current query using the *Basic List* pushbutton (or *Statistics* or *Ranked list*) in the preview area or by choosing menu option *Edit*. You use the *Output* tab to determine the type of output list. You can choose between a **basic list**, **ranked list** or **statistics**. You can only choose one of the three types for each query you define (combinations are not allowed). If you choose a basic list, every data record found is displayed in its own line. In ranked lists and statistics, the data records are compressed. The chosen list type is used the next time you output the query or when you refresh the preview or execute in the full screen mode.

You have to specify a reference currency on the *Statistics/Ranked List* settings tab. All currency amounts are converted into this currency so that they are comparable.

The reference unit behaves in the same way: You need a reference unit into which amounts are converted for amount fields in statistics and ranked lists. If you choose the *Standard List* output format, you see which conversions took place at the end of the statistic or ranked list.

You can add a counter to your statistic or ranked list. (The checkbox does not appear on this tab if the InfoSet used is based on a logical database.)

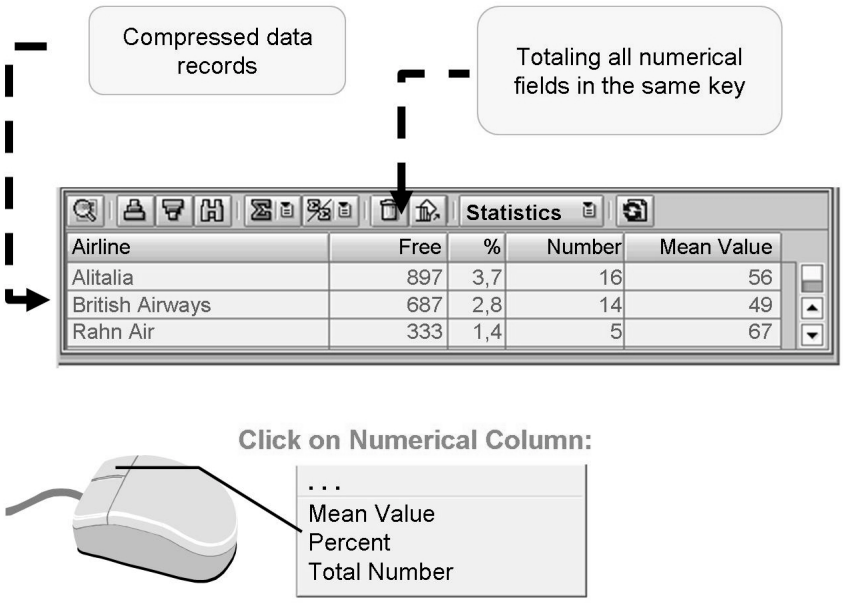


Figure 58: Statistics

The principle of statistics is identical for both SAP Query and InfoSet Query: They return data records in a compressed form. They group several data records together in the same key and create **one** information line for the key. All non-numerical fields automatically belong to the key. If the output format is the *Standard List*, the key fields are highlighted in color.

Column totals are calculated for all numerical fields. You can enhance statistics with mean values, percentage share and number of processed data records. You reach these functions in the SAP List Viewer by choosing one of the following from the context menu of the relevant column: *Output Mean Value, Output Proportion in %* or *Output Total Number*.

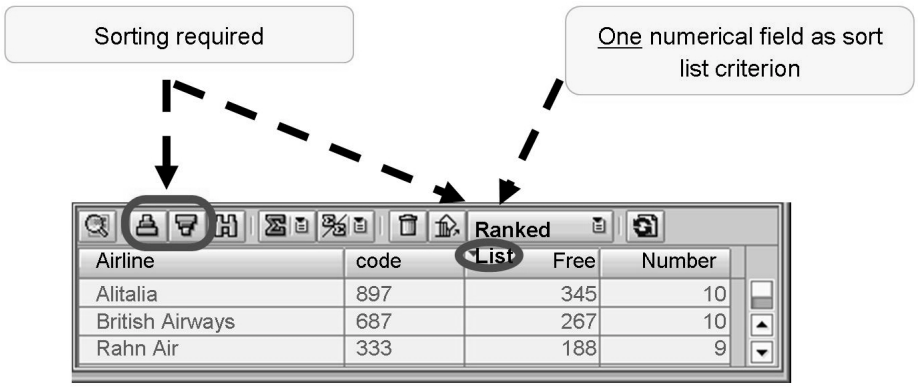


Figure 59: Ranked List

Ranked lists function in the same way in the SAP Query and InfoSet Query: They are special kinds of statistics. Data records that belong to a key are grouped together in one line in the list and the numerical values belonging are totaled. Sorting by **one** numerical value is mandatory. This value is referred to as the **ranked list criterion**. The system outputs only a certain number of places. In the example above, the three connections with the most available seats are displayed.

To create a ranked list, choose *Edit → Settings* to go to the *Output* tab page and choose the *Ranked List* entry. On the *Statistics/Ranked List* tab, you specify how many entries your ranked list is to have. In addition, you have to choose at least one numerical field from the InfoSet for the output.

In the SAP List Viewer, select the column by which you want to sort – by doing so you define the ranked list criterion. Then click on one of the two sorting buttons. If you choose *Sort in Ascending Order*, a “flop list” is created when you execute the InfoSet query which means that the entries with the lowest values in the ranked list criterion are shown at the top of the query. If you choose *Sorting in Descending Order* a “top list” is created with the highest values at the start.

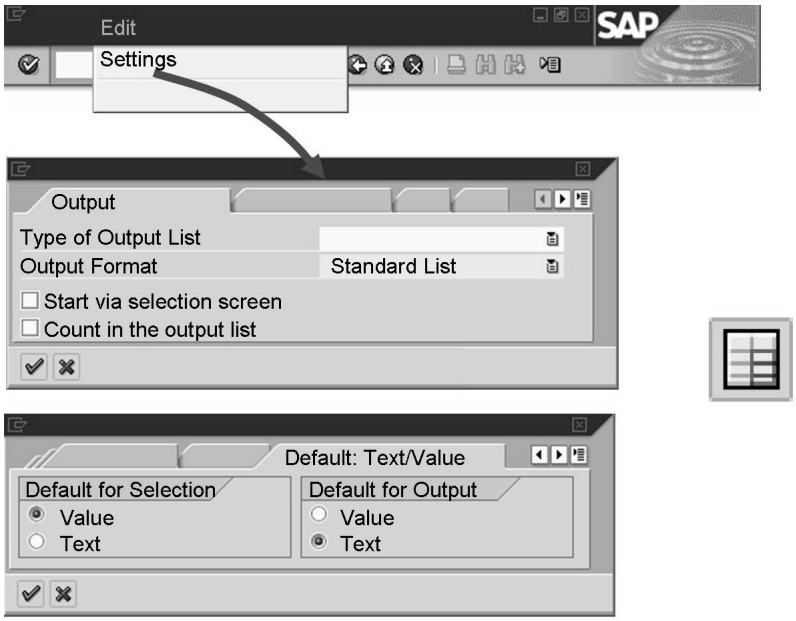


Figure 60: Other Settings

As well as the type of list, you can also determine the **type of output** in the settings. The SAP List Viewer is the default setting. Other types of output such as *Standard List* (ABAP List) correspond to the formats in the Quick Viewer or SAP Query. This settings that you make here only take effect when you execute the InfoSet Query on the full screen.

You can decide whether you want the system to display or skip over the **selection screen** when executing the query in the full screen. This means that you have the option of determining the type of output when you execute the InfoSet Query.

On the *Default: Value/Text* tab, you determine which field contents are to be displayed in the list and on the selection screen. This setting is valid for all fields that have the icon *Field with Text*, which means those fields which have automatic text recognition. You can override this setting for each column when you select the fields from the InfoSet or in the preview (SAP List Viewer) using the context menu. This is the only place where you can specify that you want the value **and** text field to be displayed in the list together.

Exercise 7: Creating Basic Lists

Exercise Objectives

After completing this exercise, you will be able to:

- Create basic lists using the InfoSet Query

Business Example

Your task:

Create an InfoSet Query that shows flight connections and occupied seats for your flights.

Model Solution:

BC407_S_03 in the global query area, user group BC407_LEARN

Task 1:

Create the query

1. From your user menu start InfoSet Query.
2. Create a new query based on the InfoSet /SAPQUERY/BC407_01.

Task 2:

Configure the InfoSet Query.

1. Create a basic list. Choose the SAP List Viewer as the default output. You do not want to display a selection screen when you execute the query.
2. As default values for the selection screen and the list, choose to display values but not text.

Task 3:

Structure the list

1. Include the following fields in the list:

   Airline code, flight code, flight date, departure city, arrival city, flight time, maximum capacity, and *number of occupied seats*. Display the short name and the full name of the airline.

Task 4:

Save the query

1. Save the query under the name **BC407##_03** in package **ZBC407_##**. ## stands for your group number.

Solution 7: Creating Basic Lists

Task 1:

Create the query

1. From your user menu start InfoSet Query.

   a) Navigate back to the initial screen of your system (for example, by entering /N in the OK code field). From your user menu start InfoSet Query.

2. Create a new query based on the InfoSet /SAPQUERY/BC407_01.

   a) Choose *Create new query*, then InfoSet /SAPQUERY/BC407_01 from the selection.

Task 2:

Configure the InfoSet Query.

1. Create a basic list. Choose the SAP List Viewer as the default output. You do not want to display a selection screen when you execute the query.

   a) Choose *Edit* → *Settings*. Choose the *Output* tab. Under *Type of Output List* choose the entry *Basic List* and under *Type of Output* choose *SAP List Viewer*. Deactivate the *Start via Selection Screen* checkbox.

2. As default values for the selection screen and the list, choose to display values but not text.

   a) Now go to the *Default: Value/Text* tab. Select the *Value* radio button both for the selection screen and the list. Confirm your entries by pressing the enter key.

Task 3:

Structure the list

1. Include the following fields in the list:

Continued on next page

Airline code, flight code, flight date, departure city, arrival city, flight time, maximum capacity, and *number of occupied seats*. Display the short name and the full name of the airline.

a) *Drag&Drop* the required fields from the upper left screen area to the preview. Alternatively, check the boxes in the list to include the fields. In this case, ensure that you have selected *Output* and not *Selection*.

If you place the fields in the list in the wrong order by mistake, move the columns by clicking on their headers in the preview and holding down the left mouse button to reposition them.

To display the *Airline code field* with its short and full names, from the context menu (right mouse button) of the Airline code column, choose *Value and Text*.

Task 4:

Save the query

1. Save the query under the name **BC407##_03** in package **ZBC407_##**. ## stands for your group number.

a) Click on the Save pushbutton (shortcut: Ctrl-S) to save the query. Enter the name **BC407##_03**. ## stands for your group number. Enter a title. Confirm your entries.

In the input template that follows, enter **ZBC407_##** as the package. Confirm your entries.

In the next screen, choose *Own Requests* and select your request by double clicking it.

Exercise 8: Ranked List with the InfoSet Query

Exercise Objectives

After completing this exercise, you will be able to:

- Create ranked lists using the InfoSet Query

Business Example

Create an InfoSet Query that displays the 10 fight connections with the highest number of free seats.

Task 1:

Create the query

1. From your user menu start InfoSet Query.
2. Create a new query based on the InfoSet **/SAPQUERY/BC407_02**.

Task 2:

Configure the InfoSet Query.

1. Create a ranked list with 10 entries. Choose the SAP List Viewer as the standard output.
2. As default values for the selection screen and the list, choose to display values but not text.

Task 3:

Determine the selection screen

1. Display the *Departure City* on the selection screen along with the two fields *Airline Code* and *Flight Date* (which were defined in the InfoSet template as selection fields).

Task 4:

Structure the list

1. Include the following fields in the list:

   Airline code, flight code, departure city, arrival city, free seats (numerical) and maximum capacity.

Continued on next page

Task 5:

Define the ranked list criterion

1. Determine the top 10 for free seats.

Task 6:

Save the query

1. Save the query under the name **BC407##_04** in development class **ZBC407_##**. ## stands for your group number.

Solution 8: Ranked List with the InfoSet Query

Task 1:

Create the query

1. From your user menu start InfoSet Query.

   a) Navigate back to the initial screen of your system (for example, by entering /N in the OK code field). From your user menu start InfoSet Query.

2. Create a new query based on the InfoSet **/SAPQUERY/BC407_02**.

   a) Choose *Create new query*, then InfoSet **/SAPQUERY/BC407_02** from the selection.

Task 2:

Configure the InfoSet Query.

1. Create a ranked list with 10 entries. Choose the SAP List Viewer as the standard output.

   a) Choose *Edit* → *Settings*. Choose the *Output* tab. Under *Type of Output List* choose the entry *Ranked List* and under *Type of Output* choose *SAP List Viewer*. Deactivate the *Start via Selection Screen* checkbox.

   On the *Statistics/Ranked List* tab, enter a reference currency and a reference unit (unimportant for this statistic since seats do not have a unit and therefore do not have to be converted). For the Number of Ranks, enter 10

2. As default values for the selection screen and the list, choose to display values but not text.

   a) Now go to the *Default: Value/Text* tab. Select the *Value* radio button both for the selection screen and the list. Confirm your entries by pressing the enter key.

Continued on next page

Task 3:

Determine the selection screen

1. Display the *Departure City* on the selection screen along with the two fields *Airline Code* and *Flight Date* (which were defined in the InfoSet template as selection fields).

   a) *Drag&Drop* the *Departure City* field from the upper left area to the selection screen in the upper right area.

Task 4:

Structure the list

1. Include the following fields in the list:

   Airline code, flight code, departure city, arrival city, free seats (numerical) and maximum capacity.

   a) *Drag&Drop* the required fields from the upper left screen area to the preview.

Task 5:

Define the ranked list criterion

1. Determine the top 10 for free seats.

   a) Select the *Free Seats* column in the preview then sort in descending order.

Task 6:

Save the query

1. Save the query under the name **BC407##_04** in development class **ZBC407_##**. ## stands for your group number.

   a) See the previous task.

Exercise 9: Statistics with the InfoSet Query (optional)

Exercise Objectives

After completing this exercise, you will be able to:

- Create statistics using the InfoSet Query

Business Example

Use an InfoSet Query to create a statistic for luggage

Task 1:

Create the query

1. From your user menu start InfoSet Query.
2. Create a new query based on the InfoSet /SAPQUERY/BC407_01.

Task 2:

Configure the InfoSet Query.

1. Create a statistic. The reference unit is KG (kilograms). Choose the SAP List Viewer as the standard output.
2. As default values for the selection screen and the list, choose to display the values but not the texts.

Task 3:

Structure the list.

1. Include the following fields in the list:

   Airline Code, Flight Connection Number, Flight Date and Weight of Luggage. You also want to see the mean value for the weight.

Task 4:

Save the query.

1. Save the query under the name BC407##_04_OPT in package ZBC407_##. ## stands for your group number.

Solution 9: Statistics with the InfoSet Query (optional)

Task 1:

Create the query

1. From your user menu start InfoSet Query.
   a) See the previous exercise.
2. Create a new query based on the InfoSet /SAPQUERY/BC407_01.
   a) See the previous exercise.

Task 2:

Configure the InfoSet Query.

1. Create a statistic. The reference unit is KG (kilograms). Choose the SAP List Viewer as the standard output.
   a) Choose *Edit → Settings*. Choose the *Output* tab. Under *Type of Output List* choose the entry *Statistics* and under Type of Output choose *SAP List Viewer*.

      On the Statistics/Ranked List tab, enter a reference currency. Choose KG as the reference unit.
2. As default values for the selection screen and the list, choose to display the values but not the texts.
   a) See the previous exercise.

Task 3:

Structure the list.

1. Include the following fields in the list:

   Airline Code, Flight Connection Number, Flight Date and Weight of Luggage. You also want to see the mean value for the weight.
   a) *Drag&Drop* the required fields from the upper left screen area to the preview.

      To issue the mean value, you must select the column, right-click and choose the *Output mean value* entry.

Continued on next page

Task 4:

Save the query.

1. Save the query under the name BC407##_04_OPT in package ZBC407_##. ## stands for your group number.

   a) See the previous exercise.

Internal Use SAP Partner Only

Lesson Summary

You should now be able to:

- Use InfoSet Query to create basic lists, statistics and ranked lists

Unit Summary

You should now be able to:

- Use InfoSet Query to create basic lists, statistics and ranked lists

Internal Use SAP Partner Only

Unit 5

Comparing the Tools

Unit Overview

In this unit we will compare the three tools QuickViewer, SAP Query and InfoSet Query from different perspectives.

Unit Objectives

After completing this unit, you will be able to:

- Describe the differences between the three reporting tools with regard to use, types of list, data sources and functions

Unit Contents

Lesson: Comparing the Tools

Lesson Overview

In this lesson we will compare the three tools QuickViewer, SAP Query and InfoSet Query from different perspectives.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe the differences between the three reporting tools with regard to use, types of list, data sources and functions

Business Example

You have been asked to compare the advantages of individual reporting tools with each other.

Operation

	QuickViewer	SAP Query	InfoSet Query
Start	SQVI	SQ01	Usually from a role
List Structure	• Classical Interface (Basic Mode) • Graphical Interface (Layout Mode)	• Classical interface for all 3 types of list • Graphical interface (Query Painter) for basic lists	Graphical interface
Mandatory Save to Execute	No	No	No

Organization

	QuickViewer	SAP Query	InfoSet Query
User groups	No QuickViews for each user	Yes (direct assignment) Queries by user group	Yes (direct assignment or using roles) Queries by user group
Query-specific authorization checks	No	*S_QUERY*	*S_QUERY*
Client-specific	Yes	Global area: No Standard area: Yes	See SAP Query
Transport	No	Yes • Global area: With automatic transport connection • Standard area: Manual	See SAP Query

There is no authorization object for the **QuickViewer**. When you execute the query, it is checked against authorization object *S_TABU_DIS* when a table, a table join or a database view is being used as the data source. If you are using an InfoSet as the data source, it inherits the authorization checks, especially the authorization groups (query with authorization object *S_PROGRAM*).

SAP Queries and **InfoSet Queries** have the authorization object *S_QUERY*, which you can use to control whether a user is allowed to change, create, manage or translate queries.

When you execute queries, the system makes checks against *S_TABU_DIS* if the InfoSet is based on a table, a table join or a database view. If a logical database is the data basis, the checks for the database are run. All other checks programmed into the InfoSet are executed.

Data Sources, Lists

	QuickViewer	SAP Query	InfoSet Query
Data basis	Table, DB view, LDB, InfoSet: Can create a join	InfoSet (has to be available)	See SAP Query
Local Fields	No	Yes	No
Automatic Text Recognition	No	Yes	Yes
Lists	1 basic list	max. 1 basic list + max. 9 statistics + max. 9 ranked lists	Basic list or statistic or ranked list
Hierarchical Lists	Yes (if output as ABAP list and definition in layout mode)	Yes (if using ABAP list as output format)	No

Functions, Other

	QuickViewer	SAP Query	InfoSet Query
Interactive Functions	Yes	Yes	Yes
Report/Report Interface (RRI)	No	Yes	Yes (Assignment only; call using SQ01)
Save List in Menu/Roles	No	Yes	No
Other	Can be converted into query	Can be processed further as an InfoSet Query (but not: sublist combination, hierarchical lists, local fields)	Can be processed further as a SAP Query

Lesson Summary

You should now be able to:

- Describe the differences between the three reporting tools with regard to use, types of list, data sources and functions

Unit Summary

You should now be able to:

- Describe the differences between the three reporting tools with regard to use, types of list, data sources and functions

Unit 6

ABAP Statements in InfoSet Creation

Unit Overview

In this unit, you will learn about the important ABAP statements that you require to take advantage of all the options of the InfoSet.

Unit Objectives

After completing this unit, you will be able to:

- Define data in an ABAP program
- Create simple selection screens
- Check Data
- Explain the event concept
- Create simple SELECT statements

Unit Contents

Lesson: ABAP Statements in InfoSet Creation

Lesson Overview

In this lesson, you will learn about the important ABAP statements that you require to take advantage of all the options of the InfoSet.

Lesson Objectives

After completing this lesson, you will be able to:

- Define data in an ABAP program
- Create simple selection screens
- Check Data
- Explain the event concept
- Create simple SELECT statements

Business Example

You want to create an InfoSet. To fill additional fields with a value at runtime, for example, you must get to know the important ABAP statements.

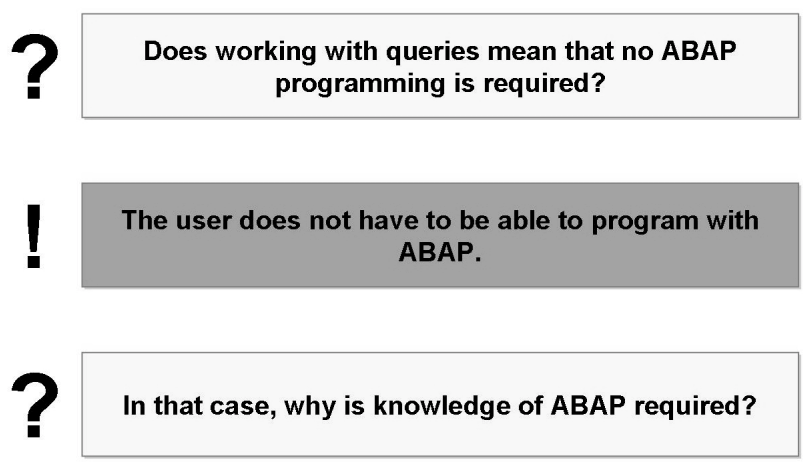


Figure 61: Motivation for this unit

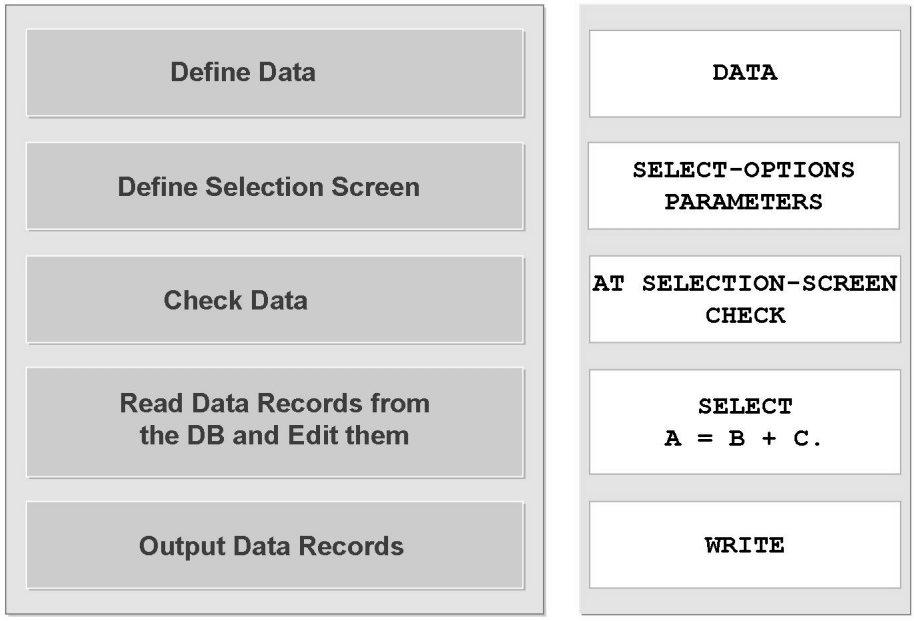


Figure 62: Program Overview

Data Objects

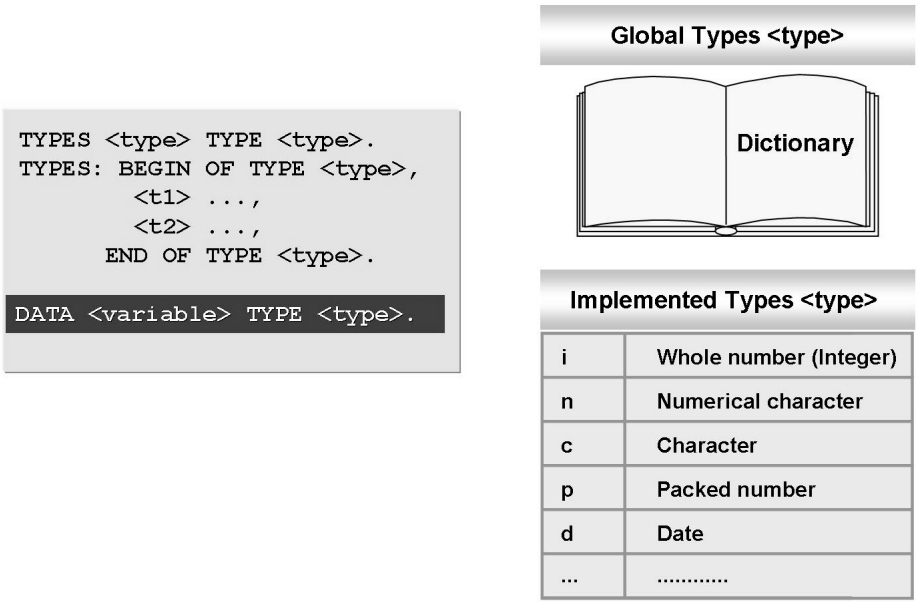


Figure 63: Data Objects

You create data objects using the ABAP statement *DATA*. You always have to type data objects. The type determines the technical properties of a data object.

You can define types globally in the ABAP Dictionary, which makes them accessible to all programs. You can also define your own types in the program. You use the ABAP statement *TYPES* to do this. Types that you create in your program using *TYPES* are know as local in program.

In addition, the following **predefined ABAP types** are available: *d* (date), *t* (time), *i* (integer), *f* (floating point number), *p* (packed number), *n* (numerical text), *c* (character), *x* (byte), *string* (string) and *xstring* (byte sequence).

You assign the types with the *TYPE* statement.

Memory is allocated to the data object according to the technical property (type) specified. This memory is filled with the content of the data object during runtime.

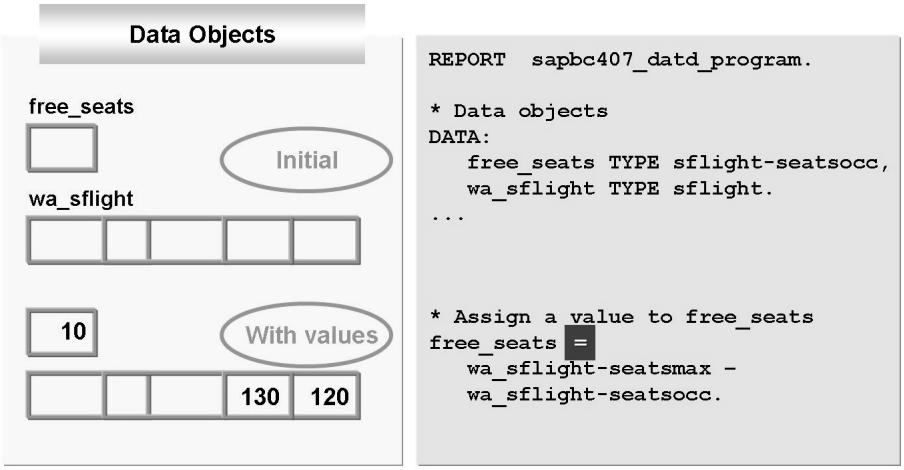


Figure 64: Filling Data Objects

Assign a value to the data object at runtime. You can overwrite this value with a new value assignment at any time (variable).

If you have not defined a default value for the data object in the *DATA* statement using the *VALUE* addition, this contains the type-related initial value:

Predefined Data Types with an Initial Value of an Appropriate Type

Data Type	Initial (Blank) Value
C	Space
N	“00...0”
D	YYYYMMDD “00000000”
T	HHMMSS “000000”
I	0 (Length 4)
P	0 (Length 8)

The Selection Screen

```
REPORT  sapbc407_datd_sel_screen.

* Data definition
DATA:  wa_sflight TYPE sflight.

* Selection screen
SELECT-OPTIONS:
   so_car FOR wa_sflight-carrid,
   so_fld FOR wa_sflight-fldate.
```

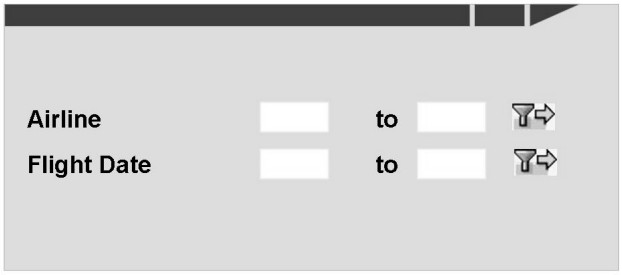


Figure 65: The Selection Screen

Selection screens serve as an interface between the program and the user. The user enters data into the fields that are ready-for-input. This data is evaluated while the program is running. This data is typically used for restricting the amount of data that is read from the database.

You create the selection screen using the ABAP statement *SELECT-OPTIONS*. After the addition *FOR*, you always have to list the name of a data object. An interval that is ready-for-input appears on the selection screen. The name that you give for the *SELECT-OPTIONS* can be a maximum of 8 places long. This name is displayed on the selection screen.

Since the internal program name of the *SELECT-OPTIONS* statement is usually of little use to the end user, you can overwrite this with a more descriptive text. You maintain the selection texts in the ABAP Editor under *Goto → Text Elements → Selection Texts*.

If you need only a simple ready-for-input field on the selection screen, use the *PARAMETERS* statement. This statement does not play a very important role in conjunction with the query.

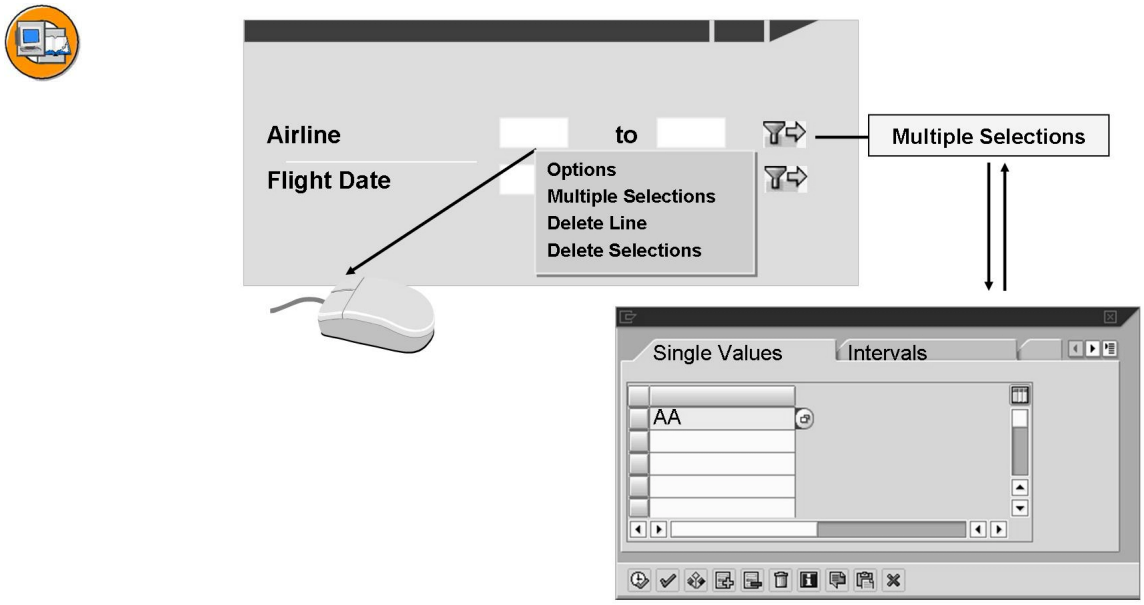


Figure 66: Selection Options and Multiple Selections

When you make entries on a selection screen, the system populates the selection table.

To show special relations, (for example, “larger” or “smaller”), choose *Selection options* (by double-clicking the entry field or by pressing the right mouse button). The relevant selection is provided with all the alternatives (when you enter an individual value, there are other options than when you enter an interval). If there is a green traffic light when you select, this means that the value is included in the selection. If there is a red traffic light, this means that the value is excluded.

To delete a table entry, choose the corresponding entry from the dropdown box *(Delete Line)*. To delete the entire selection table, choose *Delete Selections*.

Every selection criterion can be used to make multiple selections unless otherwise defined . If multiple selections are made, the color of the arrow changes from white to green.

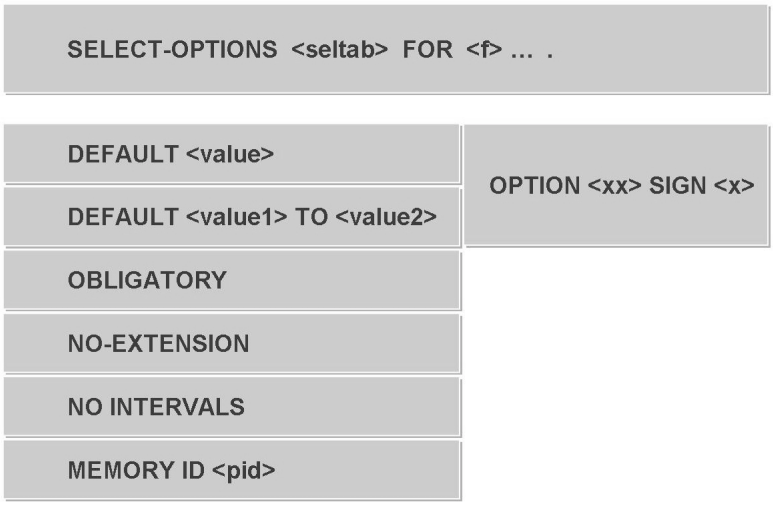


Figure 67: SELECT-OPTIOMS: Possible Additions

Additions to the *SELECT-OPTIONS* statement:

- *DEFAULT* enables you to set default values for an individual value or an interval. You can use *OPTION* and *SIGN* to set default values for the relation.
- *OBLIGATORY* generates a mandatory field. The user has to enter a value in the input field on the selection screen.
- *NO-EXTENSION* suppresses the option for multiple selections.
- *NO INTERVALS* suppresses the upper interval limit entry on the selection screen. You can use the additional screen *Multiple selection* to enter intervals.
- *MEMORY ID <pid>* allocates a set/get parameter. The value stored in SAP Memory with the ID <pid> is placed in the lower interval limit when you call the selection screen.

A logical database gives the program hierarchically structured table lines that can come from different database tables and that are transferred to the program one after the other.

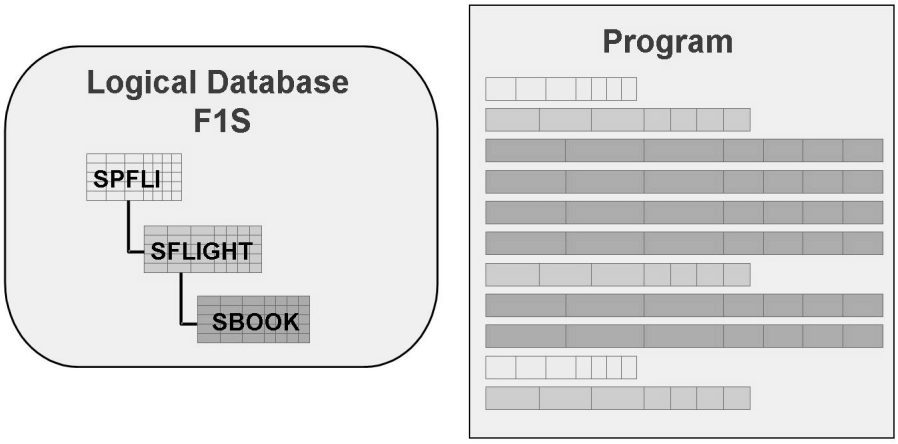


Figure 68: Using a Logical Database from a Program

A logical database is an ABAP program that reads **data from the database** and makes it available to other programs. A hierarchical structure determines the order in which the data is delivered to the programs. The programmer or user of the reporting tool decides which of the tables that are contained in the logical database he or she actually wants to use.

A logical database also provides the program with a **selection screen** that checks user entries and conducts error dialogs. If, for example, you use a logical database in a QuickView only the part of the selection screen that matches the tables that are read is sent.

You maintain logical databases in transaction SE36. You can also find further information there: *Help → Application Help.*

F1S is the logical database used in BC4XX courses. It includes the tables *SPFLI* (connections), *SFLIGHT* (flights) and *SBOOK* (reservations).

A logical database provides the program with a selection screen that corresponds to the database accessed.

Figure 69: Logical Database Selection Screens

In executable programs, you can enter a logical database in the *Program Attributes*. The logical database then supplies the program with data.

As well as reading data from the database, the logical data base offers a selection screen. The selection screen is always adapted to suit current program processing.

The Event Concept

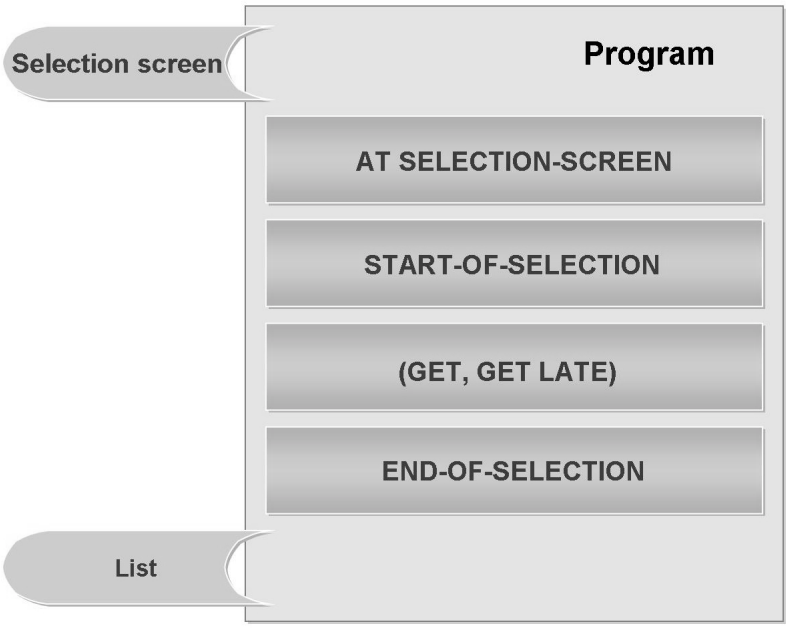


Figure 70: Process Flow of an Executable Program

The runtime system determines the process flow of an executable program (type 1) at predefined times (called events). The system specifies the sequence in which events are processed. Statements are processed sequentially within an event block.

The system cannot access all events of an executable program in connection with the query. The above list shows the events that are relevant for queries.

The system initiates an event block with an event keyword and ends it with either a new event block, the end of the program or the definition of subroutines.

The standard event is *START-OF-SELECTION*. The system assigns all ABAP statements that are not assigned to an event block to this event automatically.

During the *GET* events you need a logical database in the program.

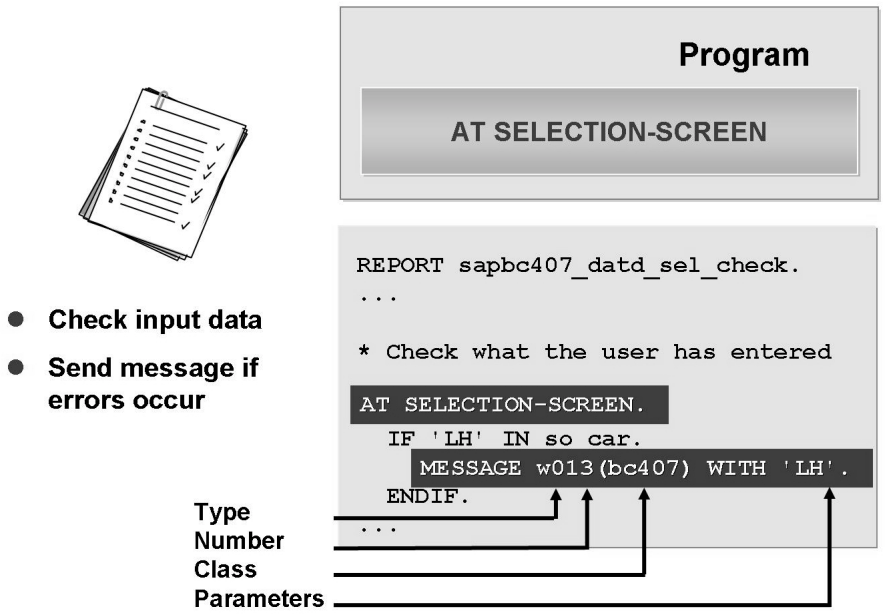


Figure 71: AT SELECTION-SCREEN Event

The system processes the event *AT SELECTION-SCREEN* after the selection screen has been sent and the user chooses *Execute* (for example, using *F8*).

You use the event for checking entries. The system checks the values that the user entered on the selection screen, for example, from logical points of view in the application.

It is important that the program reacts by sending a message if there is an error. To do this, you use the *MESSAGE* statement in ABAP. You assign messages to message classes. You have to enter the message class in the *MESSAGE* statement. You also have to determine the type of message. For error handling on the selection screen, you can choose either type E (error) or type W (warning). You

can give a message up to four parameters (variables) after the *WITH* addition. The system inserts these in the placeholders provided (&1 ... &4) in the message text at runtime.

If a user triggers an error, he or she sees a renewed selection screen. According to the message type, the user can now react in different ways.

If the program sends a *warning* message, the user can continue processing by pressing the enter key.

The user changes the entries and chooses *Execute* once again. This is required for message type *E*.

- **Output initiating information**
- **Read data from database**
- **Implement application logic**

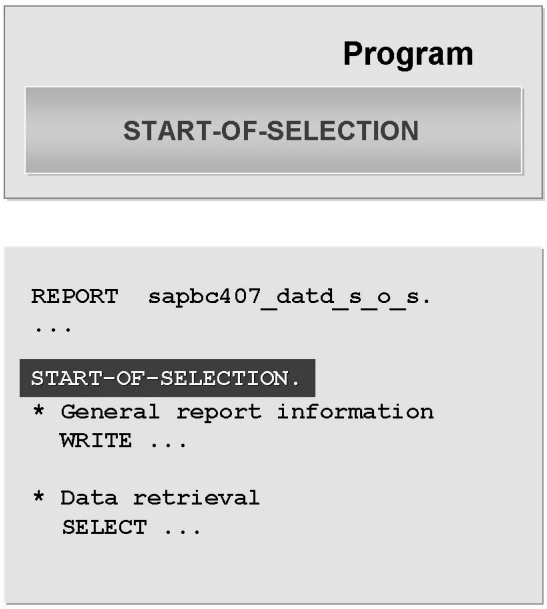


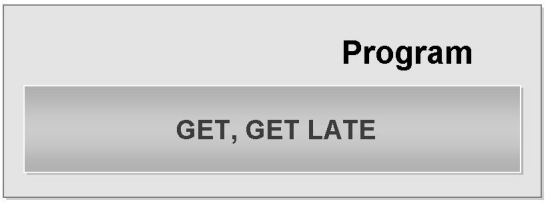
Figure 72: START-OF-SELECTION Event

START-OF-SELECTION is the standard event to which the system assigns all ABAP statements that are not assigned to another event block.

You use the event to:

- Output general texts at the start of the list
- Read data from the database
- Write data to the list

- **Deliver data records from the LDB**
- **Implement application logic**
- **LATE events: Output concluding evaluations**




```
REPORT sapbc407_datd_ldb.

* Get records of table SPFLI
GET spfli.
WRITE ...

* Get records of table SFLIGHT
GET sflight.
sum = sum + free_seats.

* LATE event of SPFLI
GET spfli LATE.
WRITE: sum.
```

Figure 73: GET and GET LATE Events

You can use the events *GET* and *GET LATE* only in connection with logical databases. In the program *Attributes*, you have to have specified the name of a logical database.

With the *GET <node>*, the logical database delivers the logical database to your program. The structure of the logical database determines which database tables can be read by that logical database. Logical database structures always form a hierarchical view of database tables (tree structure). The system delivers the data records into the program according to this hierarchy. This means that a *GET* event is usually executed more than once during the program run.

There is a *GET LATE* event for each node of a logical database. The system processes this *LATE* event after it has processed a hierarchy level. This means that these times are particularly suited to, for example, outputting totals or mean values of a hierarchy level. You calculate the values within the current *GET* event.

To maintain the logical databases, you use the *Logical Database Builder* (transaction SE36). Here you can, for example, read the structure of a logical database.

- Output concluding information
- Initialize variables

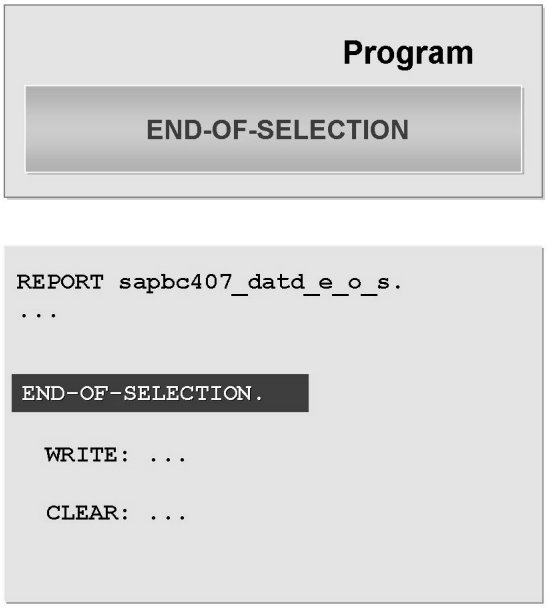


Figure 74: END-OF-SELECTION Event

You use the event *END-OF-SELECTION*, for example, to output concluding texts in the list or to initialize variables (for example, in connection with interactive reporting).

All *GET* events of the logical database have been processed at this time. The system displays the list after *END-OF-SELECTION*.

Reading from the Database

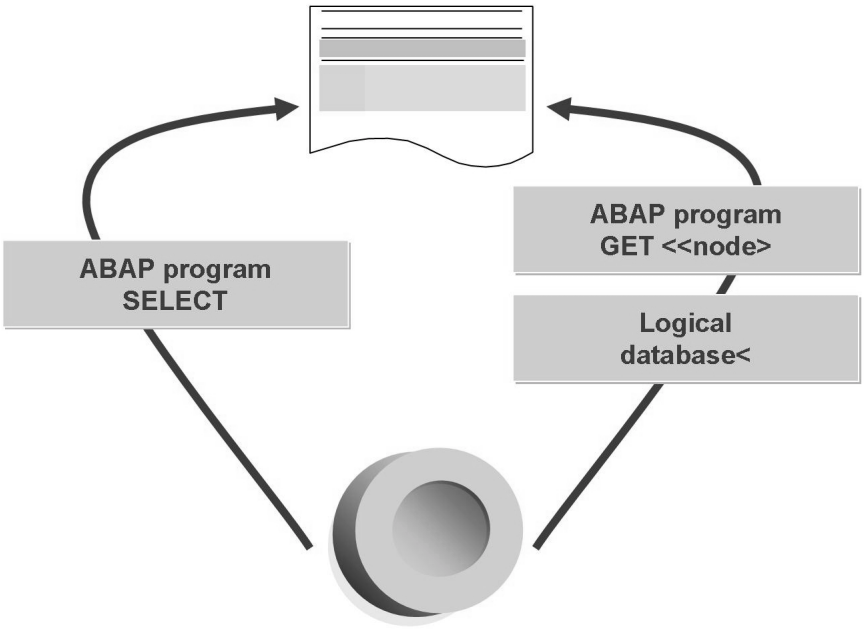


Figure 75: Read Options

In general, the system reads data that is to appear in a list from the database.

You can use OPEN SQL statements to read data from the database. You program the *SELECT* statement for this.

The use of a logical database program provides you with an excellent alternative to programming the data retrieval yourself. Logical databases retrieve data records and make them available to ABAP programs for the *GET* events.

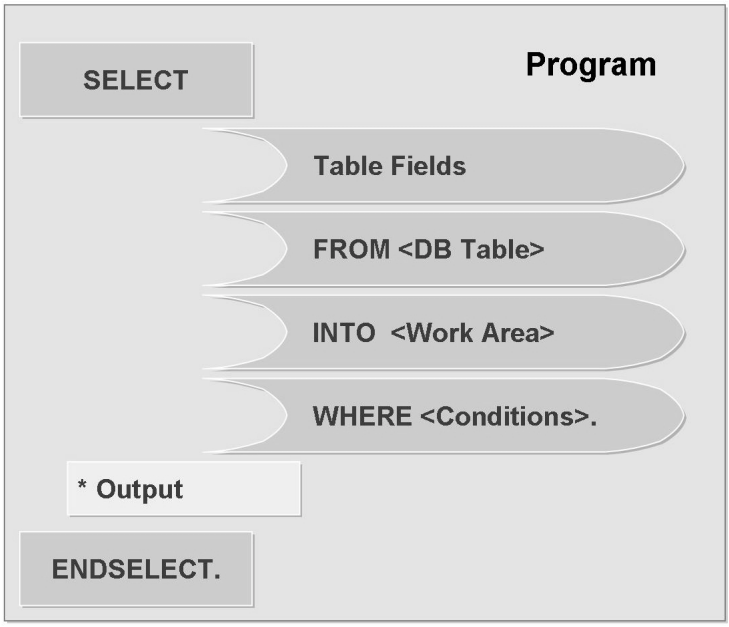


Figure 76: Structure of the SELECT Statement

The *SELECT* statement is structured in the following way:

In the *SELECT* clause, you name the fields of the database table that you want to read. If you program *SELECT* * this means that for each data record, the system delivers all columns of the database table to your program.

You specify the name of the database table to be read in the *FROM* clause. You can also list the name of a database view defined in the dictionary here.

In the *INTO* clause, you specify the target area in your ABAP program. The data records that are read are copied into here. If the target area does not correspond to the entry in the *SELECT* clause, runtime errors occur.

In the *WHERE* clause, you specify the conditions that are valid for the data records which are to be read. From the total of all saved data records, the system filters out those that correspond to these conditions. In this way, you restrict the number of data records returned. This is very important with regards to program performance.

You close single record processing using the *ENDSELECT* statement.

```
REPORT  sapbc407_datd_select.

* Data objects
DATA:  wa_sflight TYPE sflight.

* Data retrieval
SELECT *                    All table fields
  FROM sflight              DB table sflight
  INTO wa_sflight           Target area in ABAP
  WHERE ...                 Read conditions
 ...

ENDSELECT.
```

Figure 77: Reading Data Records - Example

In this example, the system is reading data from the SFLIGHT database table. All columns of the table are copied to the specified area wa_sflight, record by record.

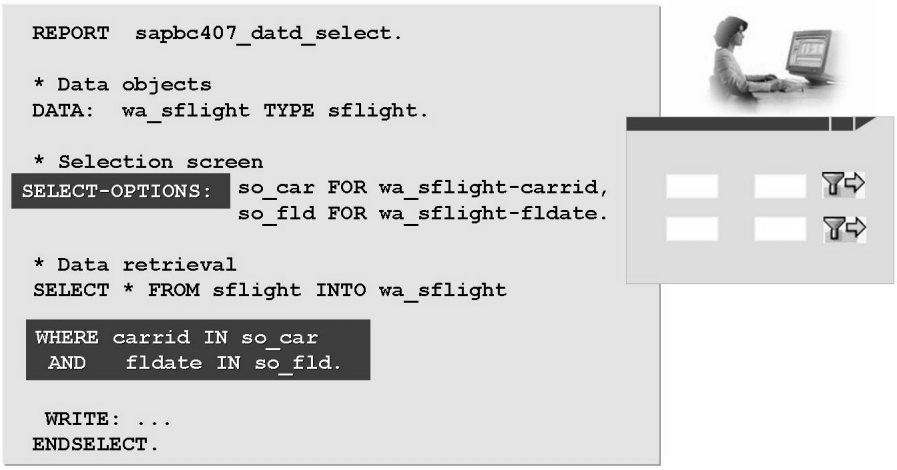


Figure 78: Connection with the Selection Screen

To restrict datasets, program the *WHERE* clause.

The *WHERE* clause is structured so that you list on the left the names of the table columns for which you want to formulate conditions. You then choose a relational operator. If you are making a comparison with a single value, use either *EQ* or =. If you are making a comparison with the contents of a selection table, you have to use the *IN* operator.

The user fills the selection table on the selection screen. The user indicates the records that he or she is interested in here.

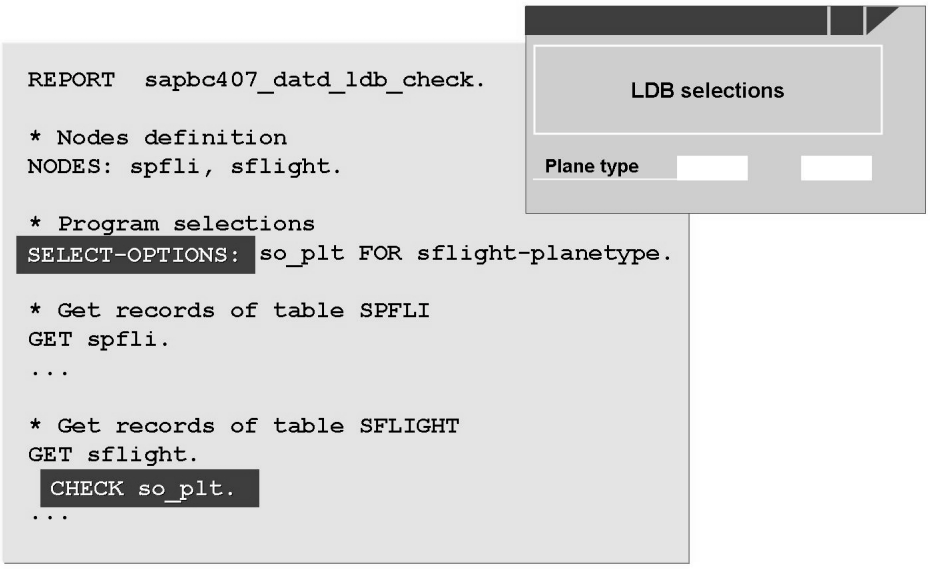


Figure 79: Distinctive Features in Connection with Logical Databases

If you are using a logical database to read data records in your program, you do not have to program the *SELECT* statement yourself. The system delivers the data records for the *GET* event into the program. The logical database offers a suitable selection screen.

If you include further *SELECT-OPTIONS* statements in your program, the logical database cannot usually convert these into the corresponding *WHERE* conditions. This is because the *SELECT* statements are already defined in the logical database program.

However, the user still sees the *SELECT-OPTIONS* that you included on their selection screen. The user fills in these selection tables and expects the dataset to be restricted accordingly. Since the logical database cannot do this, you have to check the validity of the data records delivered yourself. To do this use the ABAP command *CHECK <node>*.

Always use the *CHECK* statement in all *GET* events for tables in which you have provided additional selection tables. Always list this check as the first statement in the *GET* event.

Some logical databases provide *Free Selections* for certain tables. If you refer to such a table in the *SELECT-OPTIONS* statement (FOR), you do not have to include the check using *CHECK*.

Exercise 10: ABAP Statements in InfoSet Creation (optional)

Exercise Objectives

After completing this exercise, you will be able to:

- Program basic ABAP report commands

Business Example

Your task: Enhance a program template so that, in accordance with the user entries on the selection screen, data records of the flight table *FLIGHT* are read.

Template Program ZBC407_PROGRAM_##

Model Solution: Program SAPBC407_DATS_1_OPT

Task:

Start Editor

1. Start the ABAP Editor using transaction SE38. In the Program field, enter ZBC407_PROGRAM_## . ## stands for your group number.

Hint: The program is already created for you in package ZBC407_## and you can continue to edit it directly. You do not have to copy the template.

You choose *Change* to reach program maintenance.

The most important rules for ABAP commands:

- You end every ABAP command with a period.
- Upper-case and lower-case is irrelevant.

  Exception: Text literals in quotation marks.
- An * in the first column of a line indicates this as a comment.

The most important keyboard shortcuts are:

- **Ctrl-S**: Save
- **Ctrl-F2**: Syntax check
- **F8**:Test program
- **Ctrl-F8**: ABAP help

2. Declare data

Continued on next page

You need a work area for reading data from the SFLIGHT database table. Create a work area with the name WA_SFLIGHT and type it accordingly.

Hint: An English comment indicates the places where you have to include ABAP statements in the program.

3. Define selection screen

   The user has to be able to enter one or more airlines on the selection screen. Create a selection called SO_CAR for this and type it with reference to the field CARRID of the work area you created. Make sure that the user has to enter at least one airline.

4. Validate user input

   One of the airlines you specified is on strike. When a user selects this airline, a warning message should be displayed: “Airline XYZ is on strike.”

   Implement the check for event AT SELECTION-SCREEN using an IF statement. Send message 012 of message class BC407 if the airline which is on strike is implied in the selection table. Use the MESSAGE statement to give the name of the airline as a parameter.

5. Read data from the database. Program the data selection for the event START-OF-SELECTION. All data that matches the user entries on the selection screen is to be read from the table SFLIGHT to the work area WA_SLFIGHT. Program a SELECT command with a suitable FROM-, INTO- and WHERE clause.

6. Output data

   Within the SELECT ... ENDSELECT statement, output the following work area fields: CARRID, CONNID, FLDATE, PLANETYPE, SEATSMAX and SEATSOCC

Hint: Use the command WRITE: / wa_sflight...

7. Predefine the selection screen

   Provide an interval of your choice on the selection screen as a default.

8. Create Variant

   Create a variant for your program. Start the program with this variant.

Solution 10: ABAP Statements in InfoSet Creation (optional)

Task:

Start Editor

1. Start the ABAP Editor using transaction SE38. In the Program field, enter ZBC407_PROGRAM_## . ## stands for your group number.

Hint: The program is already created for you in package ZBC407_## and you can continue to edit it directly. You do not have to copy the template.

You choose *Change* to reach program maintenance.

The most important rules for ABAP commands:

- You end every ABAP command with a period.
- Upper-case and lower-case is irrelevant.

  Exception: Text literals in quotation marks.
- An * in the first column of a line indicates this as a comment.

The most important keyboard shortcuts are:

- **Ctrl-S**: Save
- **Ctrl-F2**: Syntax check
- **F8**:Test program
- **Ctrl-F8**: ABAP help

a) Start the ABAP Editor using transaction SE38. In the Program field, enter ZBC407_PROGRAM_## . ## stands for your group number.

The complete program is as follows:

```
REPORT sapbc407_dats_1_opt.

* Data declarations
* Datendeklarationen
DATA:
 wa_sflight TYPE sflight.

* Selection-screen
* Selektionsbild
SELECT-OPTIONS:
```

Continued on next page


```
 so_car FOR wa_sflight-carrid OBLIGATORY
 DEFAULT 'AA' TO 'DL'.

* Events
* Ereignisse
AT SELECTION-SCREEN.
 IF 'LH' IN so_car.
  MESSAGE w012(bc407) WITH 'LH'.
 ENDIF.

START-OF-SELECTION.
* Data retrieval
* Datenselektion
 SELECT *
  FROM sflight
  INTO wa_sflight
  WHERE carrid IN so_car.
  WRITE: / wa_sflight-carrid,
       wa_sflight-connid,
       wa_sflight-fldate,
       wa_sflight-planetype,
       wa_sflight-seatsmax,
       wa_sflight-seatsocc.
 ENDSELECT.
```

Hint: You can look at the model solution by entering, in the initial screen of the ABAP Editor, either SAPBC407_DATS_1 or SAPBC407_DATS_1_OPT and choosing Display.

You create a variant by starting your program, entering the required data in the selection screen and choosing Save. You reach the variant maintenance and have to enter the name of the variant and a description.

2. Declare data

You need a work area for reading data from the SFLIGHT database table. Create a work area with the name WA_SFLIGHT and type it accordingly.

Hint: An English comment indicates the places where you have to include ABAP statements in the program.

a) See the solution source code

Continued on next page

Internal Use SAP Partner Only

3. Define selection screen

   The user has to be able to enter one or more airlines on the selection screen. Create a selection called SO_CAR for this and type it with reference to the field CARRID of the work area you created. Make sure that the user has to enter at least one airline.

   a) See the solution source code

4. Validate user input

   One of the airlines you specified is on strike. When a user selects this airline, a warning message should be displayed: “Airline XYZ is on strike.”

   Implement the check for event AT SELECTION-SCREEN using an IF statement. Send message 012 of message class BC407 if the airline which is on strike is implied in the selection table. Use the MESSAGE statement to give the name of the airline as a parameter.

   a) See the solution source code

5. Read data from the database. Program the data selection for the event START-OF-SELECTION. All data that matches the user entries on the selection screen is to be read from the table SFLIGHT to the work area WA_SLFIGHT. Program a SELECT command with a suitable FROM-, INTO- and WHERE clause.

   a) See the solution source code

6. Output data

   Within the SELECT ... ENDSELECT statement, output the following work area fields: CARRID, CONNID, FLDATE, PLANETYPE, SEATSMAX and SEATSOCC

   Hint: Use the command WRITE: / wa_sflight...

   a) See the solution source code

7. Predefine the selection screen

   Provide an interval of your choice on the selection screen as a default.

   a) See the solution source code

8. Create Variant

   Create a variant for your program. Start the program with this variant.

   a) In the selection screen, choose the *Save As Variant* function. Maintain fields *Variant Name* and *Description*. Save your entries.

Lesson Summary

You should now be able to:

- Define data in an ABAP program
- Create simple selection screens
- Check Data
- Explain the event concept
- Create simple SELECT statements

Unit Summary

You should now be able to:

- Define data in an ABAP program
- Create simple selection screens
- Check Data
- Explain the event concept
- Create simple SELECT statements

Internal Use SAP Partner Only

Internal Use SAP Partner Only

Unit 7

Creating InfoSets

Internal Use SAP Partner Only

Unit Overview

In this unit, you will learn how to create InfoSets and make them available for query developers.

Unit Objectives

After completing this unit, you will be able to:

- Creating InfoSets
- Insert additional tables and fields
- Define the selection field
- Check Data
- Program ABAP in the InfoSet

Unit Contents

Lesson: Creating InfoSets

Lesson Overview

In this lesson, you will learn how to create InfoSets and make them available for query developers.

Lesson Objectives

After completing this lesson, you will be able to:

- Creating InfoSets
- Insert additional tables and fields
- Define the selection field
- Check Data
- Program ABAP in the InfoSet

Business Example

You have been asked to create an InfoSet that allows your colleagues to create F1S queries for the logical database data.

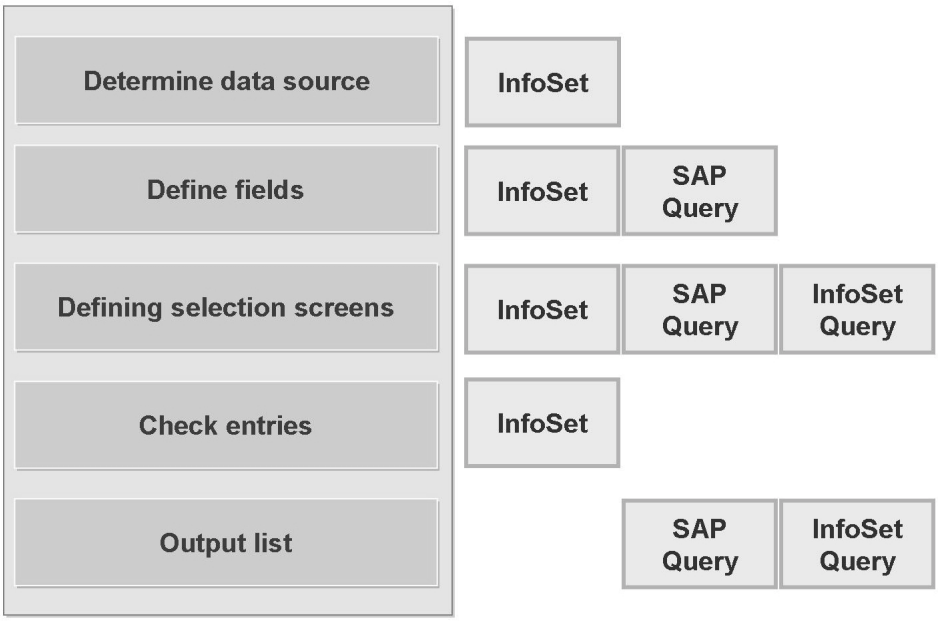


Figure 80: Overview: Distribution of Tasks

You define the data and fields in the InfoSet. This is the only option for the InfoSet Query; the SAP Query has local fields. Local fields are field definitions that are restricted to a SAP Query.

You define the selection screen in the InfoSet. The user can make further selections on the selection screen in the SAP Query and the InfoSet Query.

You program input checks in the InfoSet. To do this, you can use event *AT SELECTION-SCREEN*.

You determine the database tables that are to be read in the InfoSet. You program further processing options, such as converting or calculating values, in the InfoSet. However, you have to delimit it from data processing in the query: You can edit data in the query (SAP Query and InfoSet) using the functions available in the output. These include totaling, sorting and passing on data.

Only the user controls the output. The user determines in the query, which columns are to be displayed in the list.

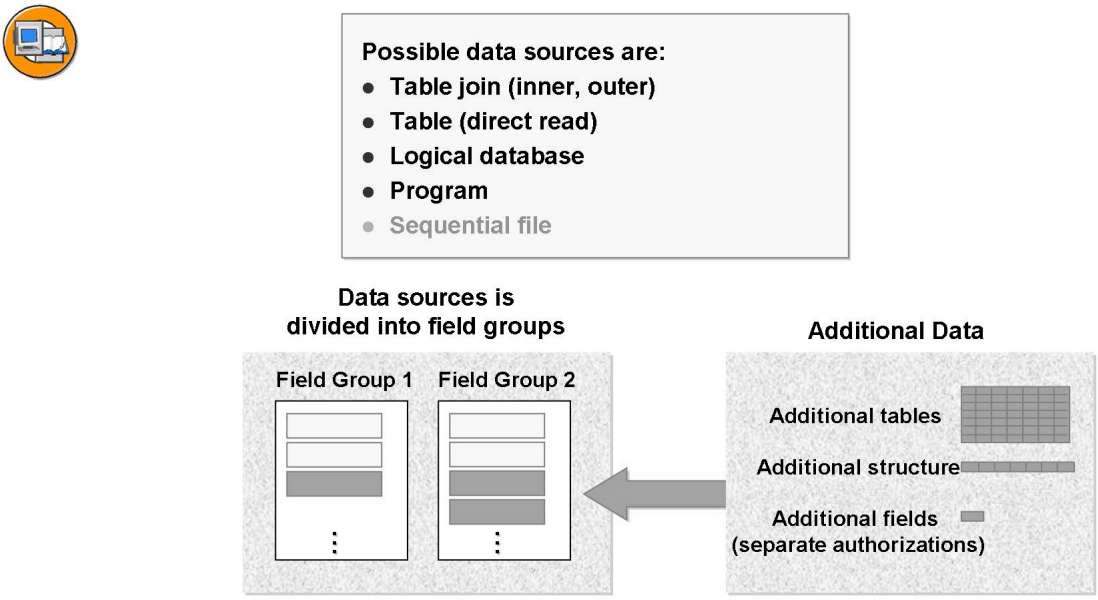


Figure 81: InfoSet DataSources

When defining an InfoSet, you must first choose a data source. There are five different types of data source:

Table join: Creating an InfoSet from a table join allows you to access data in tables that are linked with an INNER or OUTER JOIN (*SELECT*).

Direct Read: Evaluates data from a table or from a database view (*SELECT*).

Logical database: Database tables that are determined in the structure of the logical database (*GET <node>*).

User program: Reads data from a defined program.

Read sequential data: Evaluates data that is read with *READ DATASET* (no longer relevant).

You can integrate additional information into the data source (tables, additional fields).

Datasets are divided into logical units. These are called field groups. You have to assign a field to a field group so that the user can output the field in the list. Users cannot access fields that are not allocated to a field group. This means that you can filter database fields out of the InfoSet.

It is not important for the user to know whether a field comes directly from the data source or, for example, from an assigned table.

Figure 82: Creating an InfoSet: Example with Table Joins

You maintain InfoSets in transaction SQ02.

Firstly you must decide whether the InfoSet is to belong to the standard or global area. You can choose any name for the InfoSet. All SAP objects begin with the prefix */SAPQUERY/*.

When you create an InfoSet you have to give it a title and select a data source. If the data source is a table join, then you also have to define the table join. If you select a program as the data source, then the program must already exist.

The system manages every InfoSet in two versions: A generated version and a revised version. After you edit an InfoSet, you must generate it so that the system can update the changes. Queries work only with the generated versions of InfoSets. If you change an InfoSet that is already being used in queries, you have to refresh these queries.

You can delete an InfoSet only when it contains no queries.

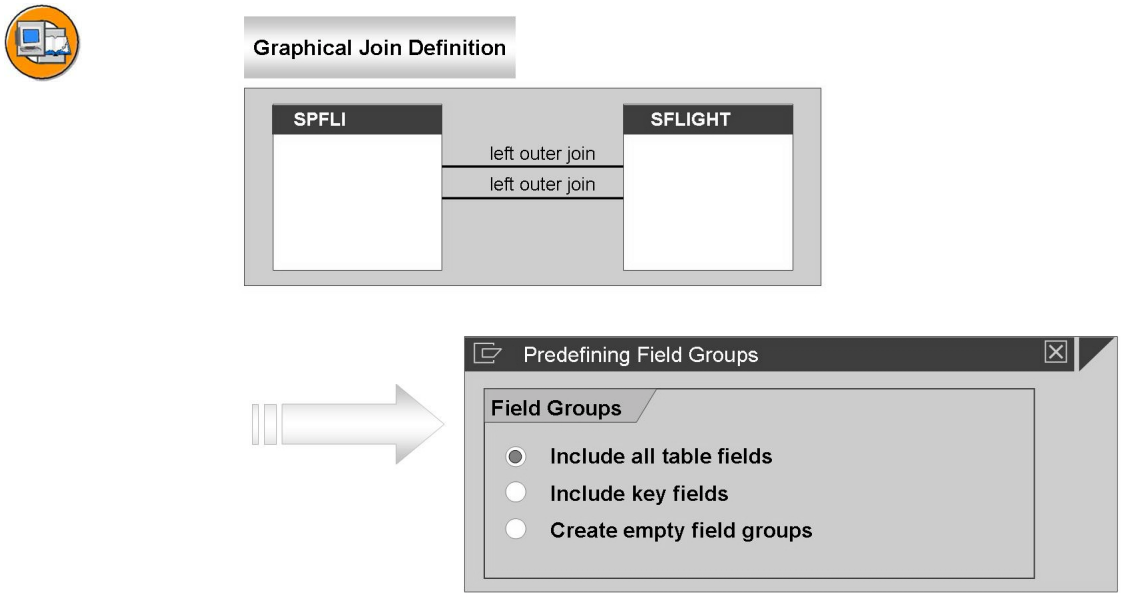


Figure 83: Predefining Join Definitions and Field Groups

If the data source consists of a table join, you must define the table join first. You can define an inner or left outer join. The system presents the user with join conditions, but these can be overwritten.

You then define the field groups. The system proposes the number and name of the field groups, and if you select the option in the dialog box, it fills the groups with fields automatically.

Field groups are classified to provide a logical division of the data. You are advised to reflect this data structure for the number of field groups, for example by creating two field groups for a join between two tables.

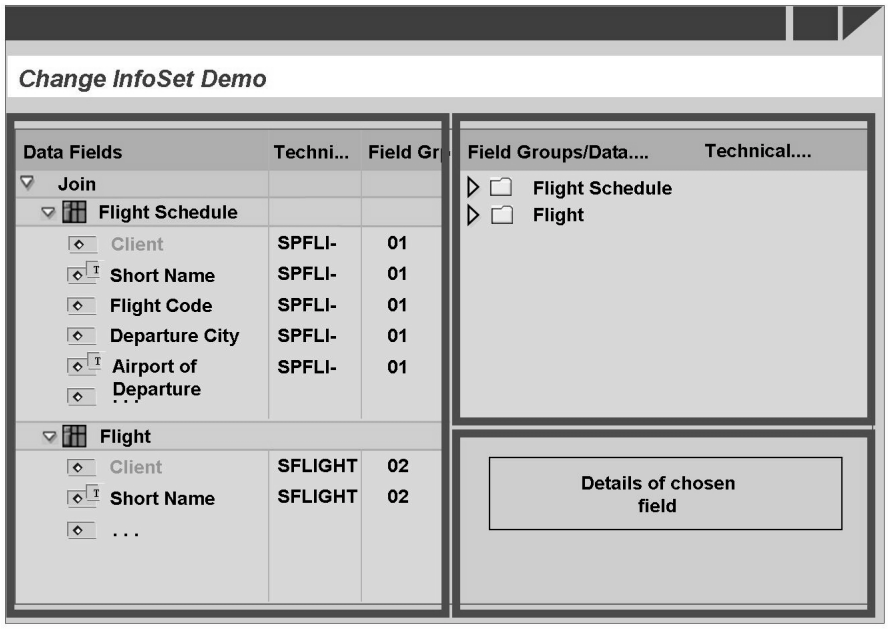


Figure 84: InfoSet: Overview

The InfoSet maintenance interface is divided into three areas. On the left-hand side of the screen the system manages the data fields in a tree. In the top-right corner of the screen, the system displays the field groups in a tree. In the bottom-right corner of the screen the system displays the details of a field that you have double-clicked.

The data fields are displayed in two different methods: With or without text fields. With a text field means that the system generates a text for the field, for example from a different table. In the example, the system reads the text for the fields *SPFLI-AIRPFROM* and *SPFLI-AIRPTO* from the table *SAIRPORT*. The system identifies the connection using the foreign key in the dictionary. When editing the InfoSet query, you can define in the settings whether you want the system to display the value or text of the data field.

You can find the functions for the nodes in the tree in the context menu. This menu selection allows you to assign a data field to a field group by positioning the cursor on the required field group. In a query you can only select fields that have been assigned to a field group.

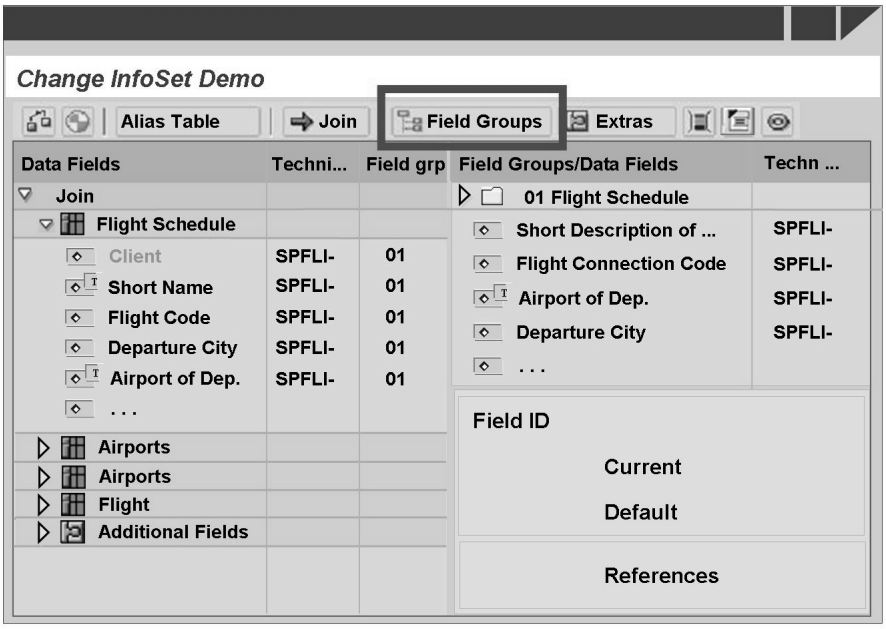


Figure 85: View: Field Groups

To add or remove fields in a field group, you can display the field groups in the upper right screen area. Using the right mouse button (context menu), you can delete a field from a field group.

You can add fields from the left tree structure to a field group using drag & drop.

If you double-click on a field, the details of this field appear in the lower right screen area. You can, for example, manipulate the texts of the fields or get the default texts from the Dictionary.

If a text field is offered for a field, you can see under *References* from which table or which table field the text was read.

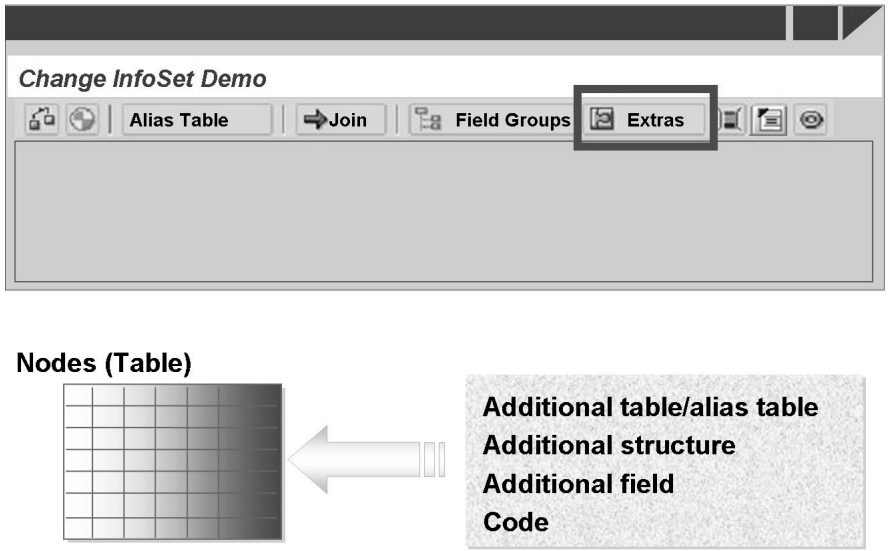


Figure 86: Additional information

You can assign additional tables, additional structures, and additional fields to every table. You can also define ABAP statements for specific, predefined events that are then integrated into the generated program.

To maintain additions for a particular table, you position the cursor on the table name in the left-hand tree and execute it with a double-click. You then choose *Extras*. Using the *Field Groups* function, you can switch back to the field group maintenance view or the detail view.

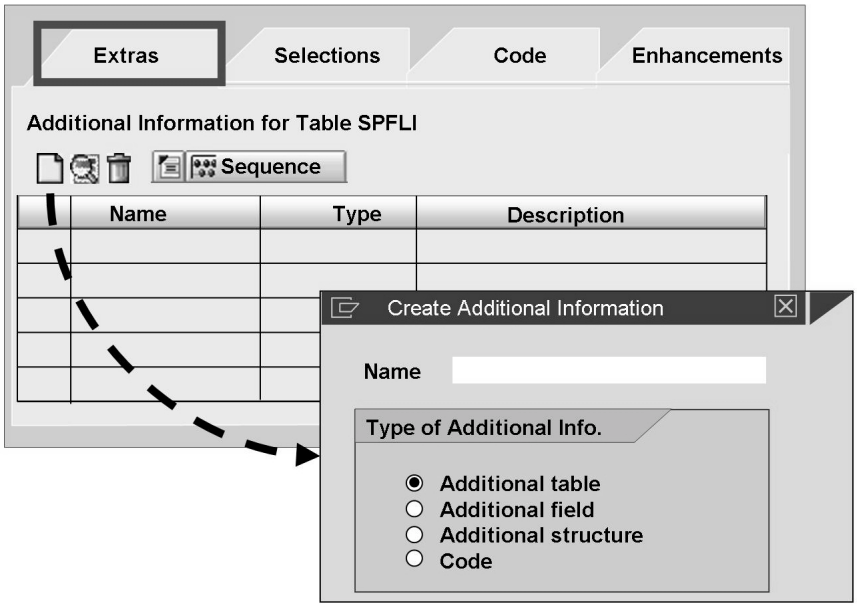


Figure 87: Creating Additions

When you choose *Extras*, a tab with the different additions appears in the right-hand section of the maintenance screen. To assign additions you must:

Position the cursor on the node name (table) in the left-hand tree (data field)

Call the *Extras* function on the screen.

You can choose the *Create* pushbutton and enter the name and type of the addition.

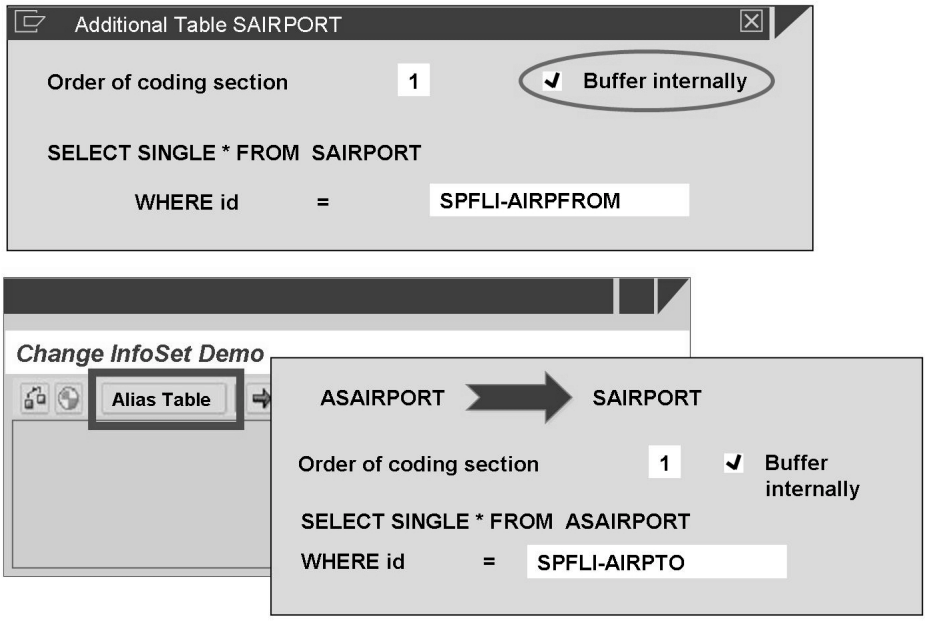


Figure 88: Connecting Additional Tables

An additional table can only be connected to a table using a *SELECT SINGLE* statement. This means that the system evaluates all key fields of the additional table in the *WHERE* condition and the system returns only one data record.

You can buffer data records in an internal table by selecting the corresponding check box. The system then creates an internal table with the single records that have been read. Whenever you request a data record, the system checks whether a data record already exists in the internal table using the *READ* statement. If a data record already exists, the system does not read it from the database another time. If the data record does not already exist, the system reads the data record from the database and adds it to the internal table.

You cannot use the same table more than once as an additional table. However, if you need to use the table again, you can define an alias table that refers to the required table, and connect the alias table instead.

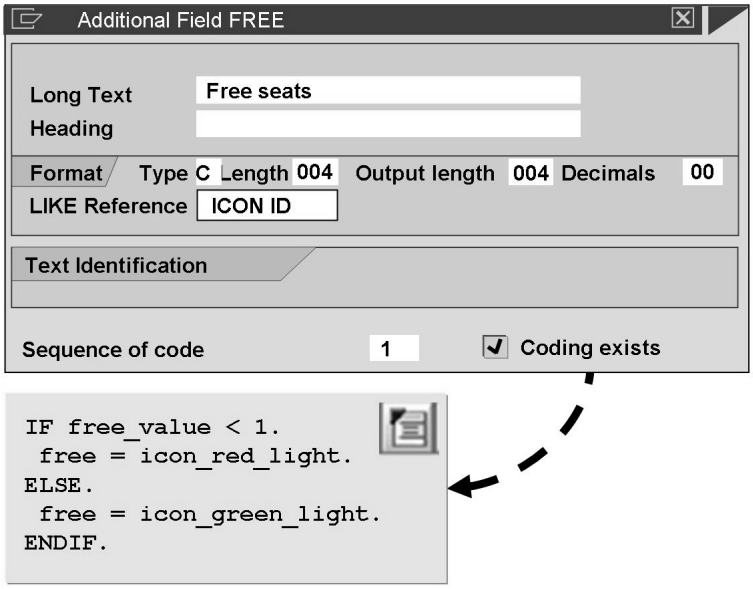


Figure 89: Creating Additional Fields

You can enter the format specifications directly using the data type or by referring to a Dictionary field. The number *Sequence of code* determines in what order the additions are integrated into the generated system. This means that you can portray dependencies.

Instead of determining the sequence manually, you can use the *calculate sequence* function to call the main screen. This function determines the dependencies between the additional fields and proposes a suitable sequence.

This screen does not contain ABAP statements for the additional field. To maintain such statements, you have to branch to the Editor using the *Code for Addition* function. This is where you store the value assignment. If you require further variables, for example the help variable *free_value* as shown above, you can define these using the DATA code.

Program Entry Points

Program Entry Points	Description
DATA	Defining Data
START-OF-SELECTION	Standard event (runs once after the selection screen)
GET	Logical Database Event
GET LATE	Logical Database Event
Record Processing	SELECT...ENDSELECT

Program Entry Points	Description
END-OF-SELECTION (Before List)	Event shortly before the end of the program but **before** the list is generated from the query (runs once)
END-OF-SELECTION (After List)	Event at the end of the program **after** the list is generated from the query (runs once)
TOP-OF-PAGE	Page Header Event
AT SELECTION-SCREEN	Entry Check Event
Free coding	Is added at the end of the query program, for example, for FORM routines

With InfoSets, you can define ABAP source text for specific program sections.

You can use events, such as *START-OF-SELECTION*, or specific subsections in the program, such as data definition and single record processing. The source text that you define has to be syntactically correct. You integrate this source text in the relevant places in the program generated.

You can only use the *GET* events if you have chosen a logical database as the data source for the InfoSet.

You cannot use the entry points to create source code for an additional field. You can only do this by navigating directly from the definition of the additional field.

In addition to the program entry points listed above, you can also use the ABAP events *INITIALIZATION* and *AT SELECTION-SCREEN OUTPUT* for complex programming for the selection screen.

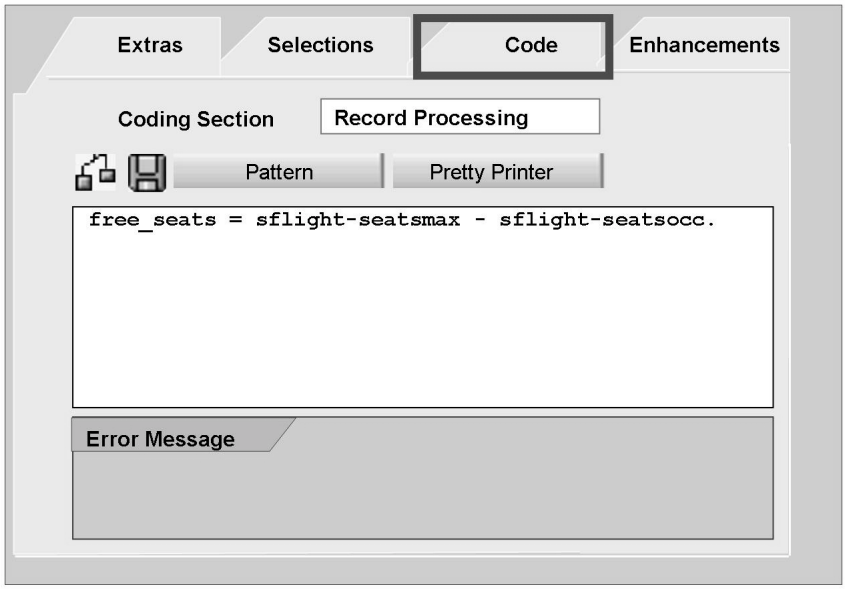


Figure 90: Record Processing

You can access the data records that were read by the database in record processing. You can change data records here if you, for example, want to convert them.

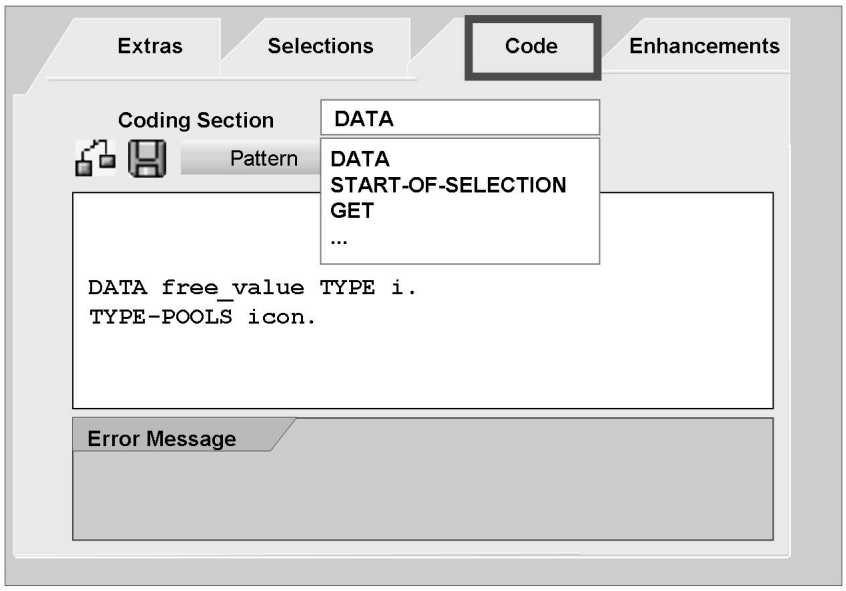


Figure 91: ABAP Statements

You can add your own statements for specific, predefined events. The statements are integrated into the generated program at the corresponding locations.

If you require further variables alongside the data source, additional tables, and additional fields, for example to calculate sums, then you can declare this in the *DATA* code. These variables cannot be assigned to a field group and therefore cannot be output on the list.

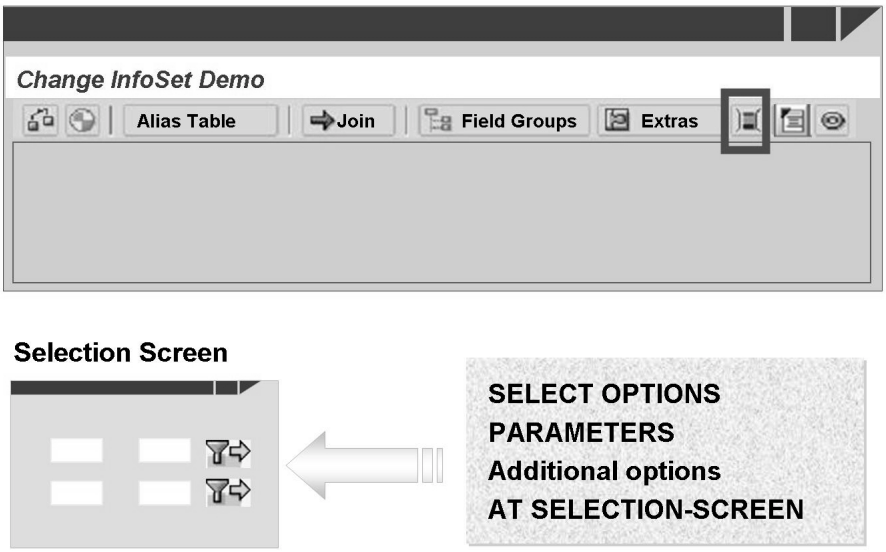


Figure 92: Define selection screen

The selection screen is defined using the *Selections* function. You can choose between selection tables and single values (parameters).

If the data source is not a logical database, you have to determine the selections manually. You can use *PARAMETERS* and *SELECT-OPTIONS*. The selections must relate to the data source or to a variable declared in the *DATA* code.

We do not recommend that you include parameters on the selection screen because they are not converted into the *WHERE* condition. If you are working with parameters, you always have to check the user's entries manually.

You can use parameters, for example, for a program control (for example, *output headers yes/no*).

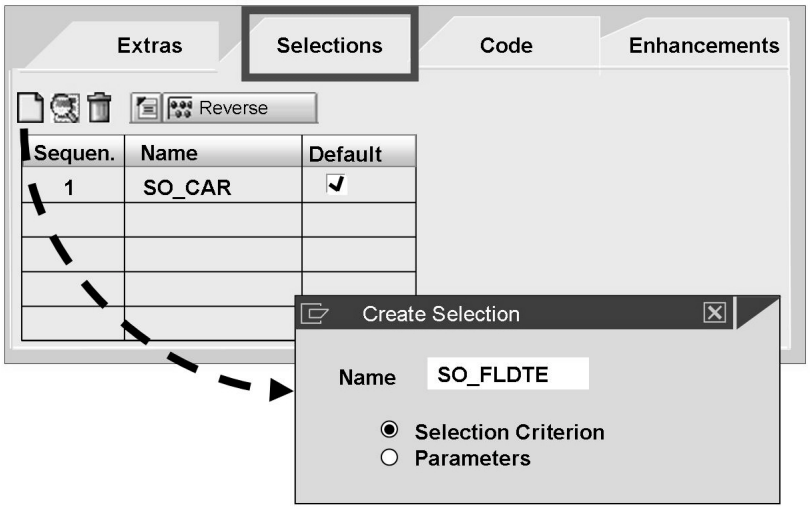


Figure 93: Selections

When declaring the sequence you determine the assignment of single selections on the selection screen. If you use a logical database as a data source, its selections appear first on the selection screen.

You can determine whether a selection in the InfoSet query reports (ad hoc reports) is offered automatically on the selection screen. To do this select the checkbox *Default on Selection Screen*.

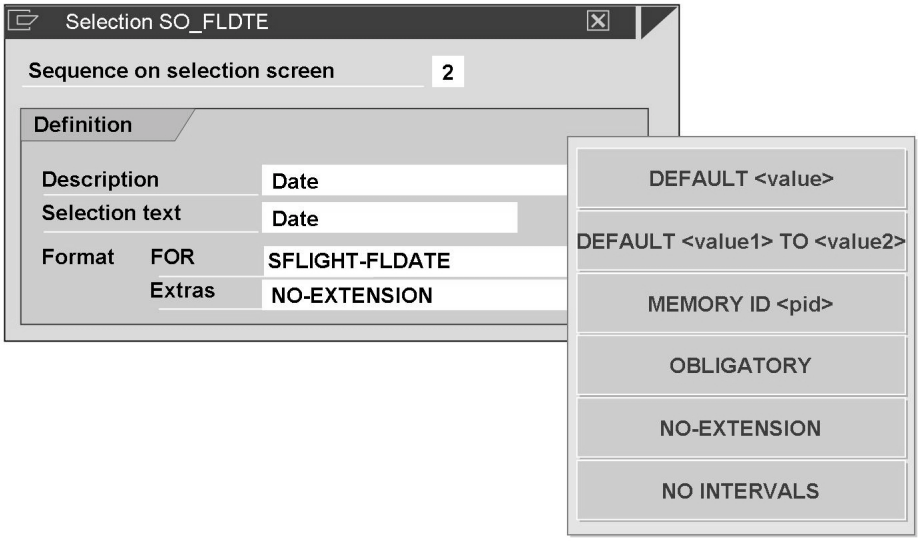


Figure 94: Selection Criteria: Determining Properties

The selections have to relate to a data object that you specify in *FOR*. Moreover, you can give all additions for the *SELECT-OPTIONS* statement.

You also determine the selection texts here. The user sees this text on the selection screen. (Comparison: When programming *SELECT-OPTIONS* statements the system overwrites technical names with selection texts.)

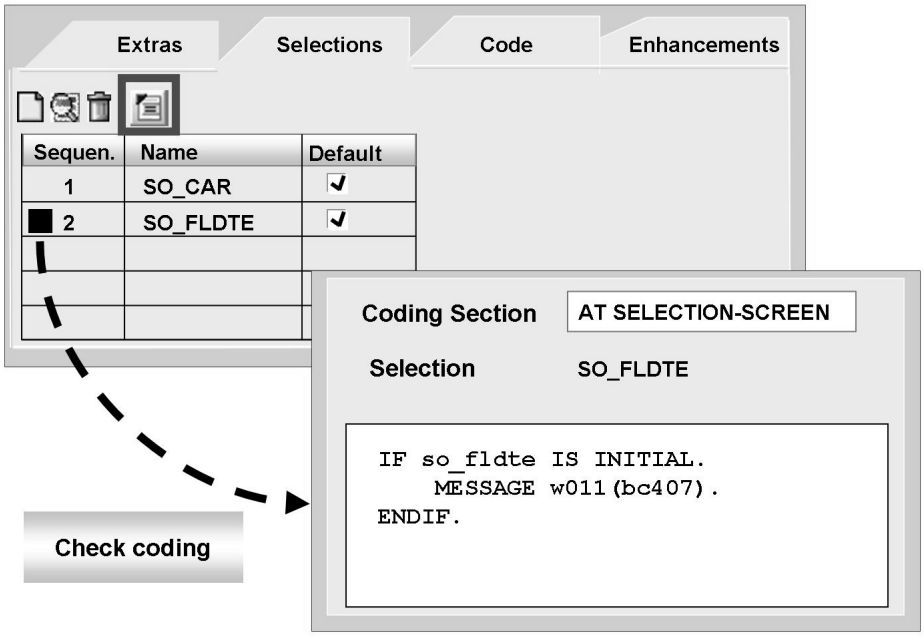


Figure 95: Defining Checks

If you want to add an input check at the time of *AT SELECTION-SCREEN*, you must program the corresponding check code. For example, you can run an authorization check or check the validity of input.

If an error occurs, make sure a message is sent. Choose either message type *E* or *W*. If an error occurs, the system displays the selection screen again.

Exercise 11: Creating an InfoSet Using a Table Join

Exercise Objectives

After completing this exercise, you will be able to:

- Create InfoSets built on a table join
- Define additional fields
- Insert additional coding
- Define the selection screen

Business Example

Create an InfoSet BC407##_01, with a table join in the global query area (where ## corresponds to your group number).

Model solution: /SAPQUERY/BC407_S1 in the global query area.

Task 1:

1. Create InfoSet BC407##_01 and maintain the description. Structure the InfoSet with a left outer join between the database tables SFLIGHT and SBOOK. Choose SFLIGHT as the left table. Adopt the join conditions suggested by the system.
2. The system asks whether you want to create field groups. Accept this suggestion. Select the setting which states that you want to transfer all fields in the field group.
3. Make sure that the fields that appear in both database tables are assigned only once to a field group. To do this, delete the redundant fields from one of the field groups. Which field group should you remove the fields from?
4. Create a variable (for example, *free_seats*) under the *DATA* coding. You need this variable to calculate the available seats (in economy class) on a flight. Type this variable in accordance with *SFLIGHT-SEATSOCC*.

Task 2:

Create an additional field (for example, *Status*). Is it important which of the two database tables you assign this additional field to?

1. Assign a long text (for example, *Display Reservation Status*) and a header text (for example, *Status*). Make sure that the technical properties correspond to those if *ICON-ID*.

Continued on next page

2. You have to assign a value to this additional field. Write the relevant ABAP coding in accordance with the following description:

   a) Compute the difference between SFLIGHT-SEATSMAX and SFLIGHT-SEATSOCC. Assign this to the variable (*free_seats*).

   b) Program the following logic using *IF – ELSE – ENDIF*:

   If a flight does not have any free seats, assign the *ICON_RED_LIGHT* icon or the *ICON_GREEN_LIGHT* icon to the additional field *Status*.

   c) To ensure that the names of the icons are interpreted correctly, you must include the statement `TYPE-POOLS icon.` in the DATA source code.

3. Assign the additional field to the field group which includes the field SFLIGHT-SEATSMAX and SFLIGHT-SEATSOCC.

Task 3:

1. Define the *selection screen*. Create three selections of the type Selection Criterion. Choose the following attributes:

Name	Reverse	Meaning/Header	Reference (FOR)
SO_CAR	1	Airline	SFLIGHT-CARRID
SO_CON	2	Connection	SFLIGHT-CONNID
SO_FLD	3	Date	SFLIGHT-FLDATE

2. All of the selection criteria have to appear on the selection screen of an InfoSet Query automatically.

Task 4:

1. Change the text *Airline Code* to *Airline*.
2. Check and generate the InfoSet. Assign the InfoSet to user group BC407_LEARN. Stay in the assignment screen only as long as is absolutely necessary. Only one user can assign InfoSets to a user group at any one time (the system sets a lock).
3. Create an SAP Query to test your new InfoSet. You can use query BC407_S_05 from user group BC407_LEARN as a reference. Enter fields from both tables.

Solution 11: Creating an InfoSet Using a Table Join

Task 1:

1. Create InfoSet BC407##_01 and maintain the description. Structure the InfoSet with a left outer join between the database tables SFLIGHT and SBOOK. Choose SFLIGHT as the left table. Adopt the join conditions suggested by the system.

   a) Start transaction SQ02 and create the InfoSet BC407##_01. Make sure that you are in the global query area. Enter a short text and in the Data Source section, select **Table Join using Basis Table**. Enter the table SFLIGHT and confirm your entries. Add the table SBOOK and change the join conditions. To do this, position the cursor on one of the series lines and click the right mouse button. Change to **left outer**. Leave the screen by choosing the green arrow.

2. The system asks whether you want to create field groups. Accept this suggestion. Select the setting which states that you want to transfer all fields in the field group.

   a) The system asks whether you want to create field groups. Accept this suggestion. Select the setting which states that you want to transfer all fields in the field group.

3. Make sure that the fields that appear in both database tables are assigned only once to a field group. To do this, delete the redundant fields from one of the field groups. Which field group should you remove the fields from?

   a) From the Flight Bookings field group, remove the fields *Airline, Flight Connection Code* and *Flight Date*. By doing this, the user sees the relevant key fields in the first field group. You can find the function *Remove Field from Field Group* in the context menu (right mouse button).

4. Create a variable (for example, *free_seats*) under the *DATA* coding. You need this variable to calculate the available seats (in economy class) on a flight. Type this variable in accordance with *SFLIGHT-SEATSOCC*.

   a) Change the right-hand side of the screen by choosing *Extras*. Choose the *Coding* tab. In the *Code Section* field, select *DATA*. Implement the following ABAP coding:

   ```
   DATA: free_seats TYPE sflight-seatsocc.
   ```

Continued on next page

Task 2:

Create an additional field (for example, *Status*). Is it important which of the two database tables you assign this additional field to?

1. Assign a long text (for example, *Display Reservation Status*) and a header text (for example, *Status*). Make sure that the technical properties correspond to those if *ICON-ID*.

   a) Choose the *Extras* tab. Create the addition field *Status*. It does not matter with which table you create this field in since the same (join!) data record is delivered to the program from either of the tables SFLIGHT and SBOOK.

   b) Assign the long text *Display Reservation Status* and the header text *Status*. For the LIKE reference, enter *ICON-ID*. Leave the screen.

2. You have to assign a value to this additional field. Write the relevant ABAP coding in accordance with the following description:

   a) Compute the difference between SFLIGHT-SEATSMAX and SFLIGHT-SEATSOCC. Assign this to the variable (*free_seats*).

   b) Program the following logic using *IF – ELSE – ENDIF*:

   If a flight does not have any free seats, assign the *ICON_RED_LIGHT* icon or the *ICON_GREEN_LIGHT* icon to the additional field *Status*.

   c) To ensure that the names of the icons are interpreted correctly, you must include the statement `TYPE-POOLS icon.` in the DATA source code.

   a) Place the cursor on the name of the additional field and choose *Coding for Addition* (icon).

   b) Implement the following ABAP coding:

```
free_seats = sflight-seatsmax - sflight-seatsocc.
 IF free_seats < 1.
   status = ICON_RED_LIGHT.
 ELSE.
  status = ICON_GREEN_LIGHT.
ENDIF.
```

   c) Choose the *Coding* tab. In the *Code Section* field, select *DATA*. Implement the following ABAP coding: `TYPE-POOLS icon.`

3. Assign the additional field to the field group which includes the field SFLIGHT-SEATSMAX and SFLIGHT-SEATSOCC.

   a) Assign the additional field to the *Flight* field group.

Continued on next page

Task 3:

1. Define the *selection screen*. Create three selections of the type Selection Criterion. Choose the following attributes:

Name	Reverse	Meaning/Header	Reference (FOR)
SO_CAR	1	Airline	SFLIGHT-CARRID
SO_CON	2	Connection	SFLIGHT-CONNID
SO_FLD	3	Date	SFLIGHT-FLDATE

   a) Choose *Selections* (icon). Create three selections of the type *Selection Criterion*.

2. All of the selection criteria have to appear on the selection screen of an InfoSet Query automatically.

   a) For all three selections, mark the checkbox *As Default on Selection Screen*.

Task 4:

1. Change the text *Airline Code* to *Airline*.

   a) Change the right-hand side of the screen by choosing *Field Groups*. When you double-click the *Airline Code* field, the detail view for the field appears in the lower right area. Change the text to *Airline* and choose *Copy*.

2. Check and generate the InfoSet. Assign the InfoSet to user group BC407_LEARN. Stay in the assignment screen only as long as is absolutely necessary. Only one user can assign InfoSets to a user group at any one time (the system sets a lock).

   a) Check and generate the InfoSet. Assign the InfoSet to the user group BC407_LEARN.

3. Create an SAP Query to test your new InfoSet. You can use query BC407_S_05 from user group BC407_LEARN as a reference. Enter fields from both tables.

   a) If you only enter fields from table *SFLIGHT*, you will receive records that appear identical.

Exercise 12: Creating an InfoSet Using a Logical Database

Exercise Objectives

After completing this exercise, you will be able to:

- Create InfoSets built on a logical database
- Define additional tables
- Insert additional coding
- Define the selection screen

Business Example

Your task: Create an InfoSet BC407##_02, with logical database F1S in the global query area where ## is your group number.

Model solution: /SAPQUERY/BC407_S2 in the global query area.

Task 1:

1. Create the InfoSet BC407##_02 and maintain the long text. Structure the InfoSet using the database F1S.

Task 2:

The system automatically creates empty field groups that correspond to the structure of the logical database.

1. Transfer all of the fields of table SPFLI to the first node of the field group. Transfer all of the fields of the table SFLIGHT, without the fields that are redundant for table SPFLI, to the second node of the field group. Transfer all of the fields of the table SBOOK, without the fields that are redundant for table SFLIGHT, to the third node of the field group.
2. Create a further (new) field group with the name *Flight Customer*.

Task 3:

1. Create the table SCUSTOM as an additional table for table SBOOK.
2. Create a connection between the two tables SBOOK-CUSTOMID and SCUSTOM-ID. Ensure optimum performance.
3. Add fields from the additional table to the *Flight Customer* field group.

Continued on next page

Task 4:

1. Define the selection screen. Create a selections of the type *Selection Table*. Hide the multiple selection (addition of statement *SELECT-OPTIONS*). Where do the selections that are already defined come from?

Name	Reverse	Meaning/Header	Reference
SO_PRICE	1	Airfare	SFLIGHT-PRICE

Task 5:

1. You have an InfoSet that is based on a logical database. You have also defined a selection criterion (price) that relates to the node SFLIGHT from the logical database. Consider whether the logical database can evaluate the price entries made by the user. What can you do?
2. Check and generate the InfoSet. Assign the InfoSet to the user group BC407_LEARN.
3. Create an InfoSet Query to test your new InfoSet. You can use query BC407_S_06 from user group BC407_LEARN as a reference.
4. **Optional**

   Consider whether it is worthwhile querying only the price on the selection screen, without offering the user a reference currency. If you want to use a reference currency, what do you have to do in ABAP?

Solution 12: Creating an InfoSet Using a Logical Database

Task 1:

1. Create the InfoSet BC407##_02 and maintain the long text. Structure the InfoSet using the database F1S.

   a) Call transaction SQ02 and create the InfoSet BC407##_02. Make sure that you are in the global query area. Enter a short text and in the *Data Source* section, select **Logical Database** and enter the logical database F1S.

Task 2:

The system automatically creates empty field groups that correspond to the structure of the logical database.

1. Transfer all of the fields of table SPFLI to the first node of the field group. Transfer all of the fields of the table SFLIGHT, without the fields that are redundant for table SPFLI, to the second node of the field group. Transfer all of the fields of the table SBOOK, without the fields that are redundant for table SFLIGHT, to the third node of the field group.

   a) Proceed as described in the previous task.

2. Create a further (new) field group with the name *Flight Customer*.

   a) Proceed as described in the previous task. Create the new field group by choosing *Create*.

Task 3:

1. Create the table SCUSTOM as an additional table for table SBOOK.

   a) In the left-hand tree, double-click on the table name SBOOK. Change the right-hand side of the screen by choosing *Extras*. Choose the *Extras* tab. Create additional table SCUSTOM for table SBOOK.

2. Create a connection between the two tables SBOOK-CUSTOMID and SCUSTOM-ID. Ensure optimum performance.

   a) In the screen that follows, the SELECT SINGLE statement is displayed. The WHERE condition is suggested correctly. Mark the Internal Buffer checkbox.

Continued on next page

3. Add fields from the additional table to the *Flight Customer* field group.

   a) Proceed as described in the previous task.

Task 4:

1. Define the selection screen. Create a selections of the type *Selection Table*. Hide the multiple selection (addition of statement *SELECT-OPTIONS*). Where do the selections that are already defined come from?

Name	Reverse	Meaning/Header	Reference
SO_PRICE	1	Airfare	SFLIGHT-PRICE

   a) Choose *Selections* (icon). Create a selection of the type Selection Criterion. Hide the multiple selections with the addition NO-EXTENSION. The selections that are already defined come from the logical database and are always displayed on the selection screen to start with.

Task 5:

1. You have an InfoSet that is based on a logical database. You have also defined a selection criterion (price) that relates to the node SFLIGHT from the logical database. Consider whether the logical database can evaluate the price entries made by the user. What can you do?

   a) Choose the *Coding* tab. In the *Code Section* field, select GET. Make sure that you assign the coding to table SFLIGHT (double-click in the left-hand tree on the table name beforehand). Implement the following ABAP coding:

      CHECK so_price.

2. Check and generate the InfoSet. Assign the InfoSet to the user group BC407_LEARN.

   a) Check and generate the InfoSet. Assign the InfoSet to the user group BC407_LEARN. Leave the assignment screen as soon as possible, since the system locks this function for one user exclusively.

3. Create an InfoSet Query to test your new InfoSet. You can use query BC407_S_06 from user group BC407_LEARN as a reference.

   a) Create an InfoSet Query to test your new InfoSet. You can use query BC407_S_06 from user group BC407_LEARN as a reference.

Continued on next page

Internal Use SAP Partner Only

4. **Optional**

Consider whether it is worthwhile querying only the price on the selection screen, without offering the user a reference currency. If you want to use a reference currency, what do you have to do in ABAP?

a) A worthwhile procedure would be to offer a reference currency on the selection screen and convert all delivered data records into this currency under the GET event and then run the validity check.

Lesson Summary

You should now be able to:

- Creating InfoSets
- Insert additional tables and fields
- Define the selection field
- Check Data
- Program ABAP in the InfoSet

Unit Summary

You should now be able to:

- Creating InfoSets
- Insert additional tables and fields
- Define the selection field
- Check Data
- Program ABAP in the InfoSet

Internal Use SAP Partner Only

Internal Use SAP Partner Only

Unit 8

User administration

Unit Overview

In this unit, you will learn how to administer query users. You will learn how to create user groups and how to call the InfoSet Query directly from a menu entry for a role.

Unit Objectives

After completing this unit, you will be able to:

- Manage users for SAP Queries and InfoSet Queries
- Set up the user menu so that the user can access the InfoSet Query

Unit Contents

Lesson: User Management

Lesson Overview

In this lesson, you will learn how to manage query users. You will learn how to create user groups and how to call the InfoSet Query directly from a menu entry for a role.

Lesson Objectives

After completing this lesson, you will be able to:

- Manage users for SAP Queries and InfoSet Queries
- Set up the user menu so that the user can access the InfoSet Query

Business Example

You have been asked to manage the users of SAP Query in your company, that is, in particular to ensure that each user can access the data they require for their job.

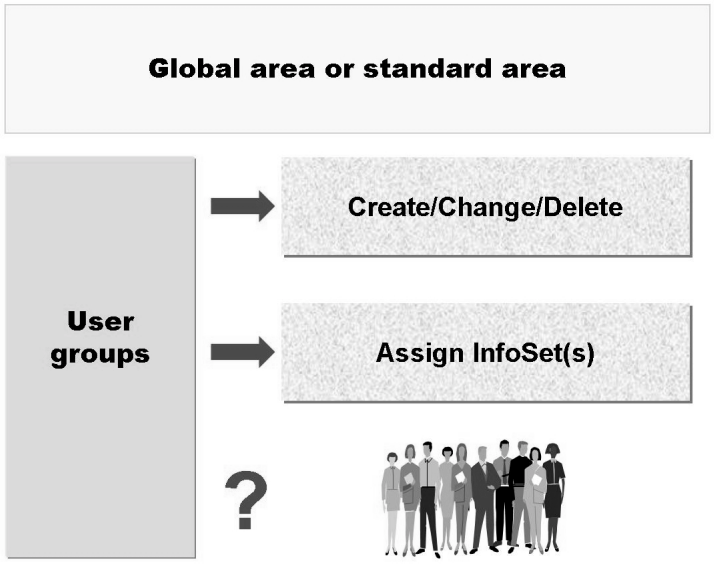
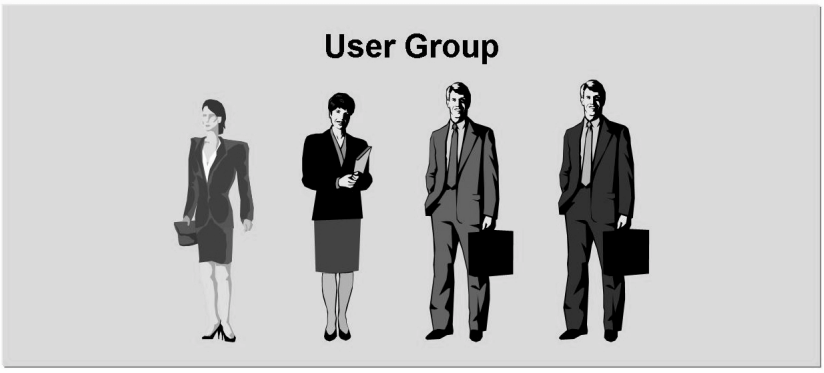


Figure 96: Creating User Groups

You create user groups in transaction SQ03. The SAP namespace begins with */SAPQUERY/*.

You assign a user group to the standard area or the global area. You have to assign the InfoSets, with which the user of the group can create queries, to the user group. You can allocate an InfoSet to several user groups.

You assign user groups to users differently in SAP Query and InfoSet Query.



All users in a user group

- **Access the same InfoSets**
- **See all queries for the user group**
- **Can have different authorization for creating and changing queries**

Figure 97: SAP Query: Assigning User Groups

You assign the users of SAP Query by entering the user in the user group (transaction SQ03).

The user sees all of the queries for the group he or she is assigned to and can execute these queries. As the user, you can create new queries using the InfoSets assigned to the user group or change existing queries, as long as you have the correct authorization.

You manage authorizations for changing or creating queries in the *S_QUERY* authorization object. This object means that a user has the general authorization to create and change queries. You have the option in the user group of taking away the change authorization for a **specific** user group from a user with change authorization in accordance with the authorization object *S_QUERY*. You do this by removing the mark from the relevant checkbox when you specify the user.

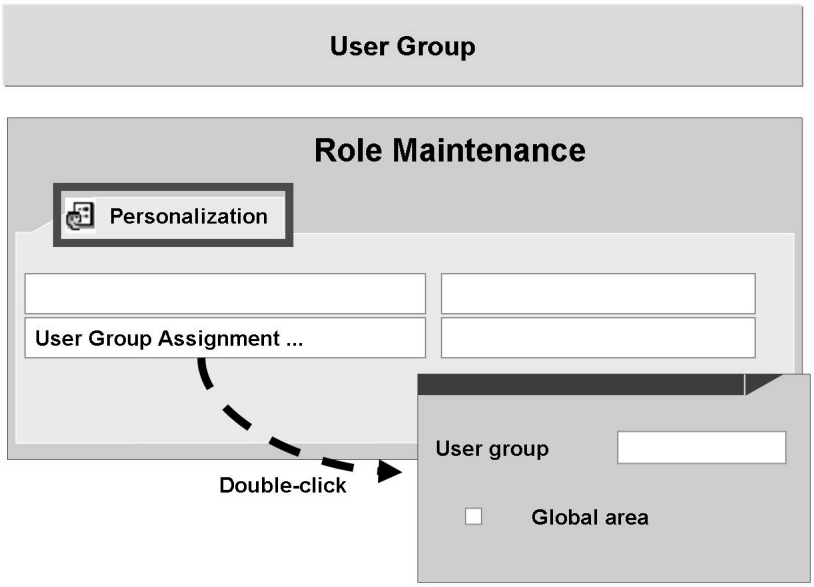


Figure 98: InfoSet Query: Assigning User Groups

InfoSet Query is integrated into the role concept. You use transaction PFCG to maintain roles. You assign the user group to a role you can only assign one user group to each role.

You assign a user group to a role on the *Personalization* tab. You see the entry *Assign User Group to SAP Query*. When you double-click on this line, a dialog box appears in which you enter the name and area of the user group. This links the role to a specific user group.

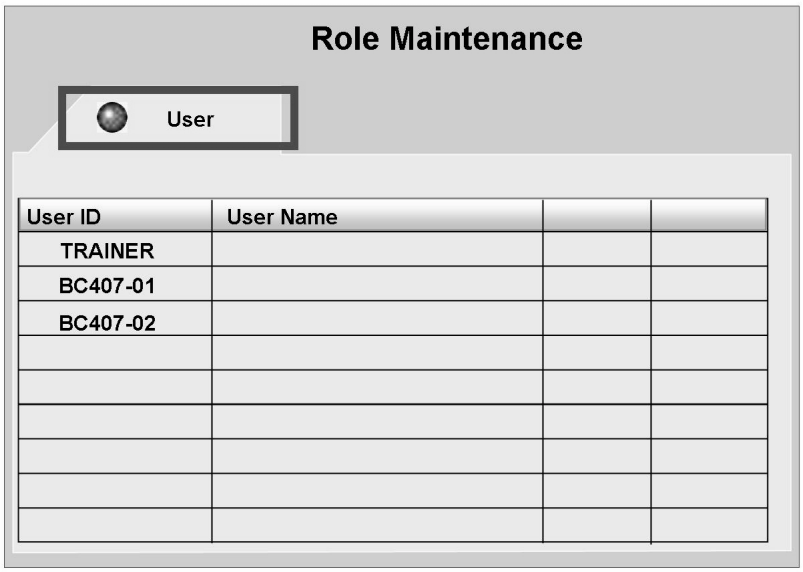


Figure 99: InfoSet Query: Assigning Users

In contrast to the SAP Query, you do **not** have to enter a user in the user group. You define the users that can access the user group, and therefore the InfoSets assigned to it, in the role maintenance

The *users* that you listed in the role maintenance are the ones that are assigned to the user group that you defined in *Personalization*.

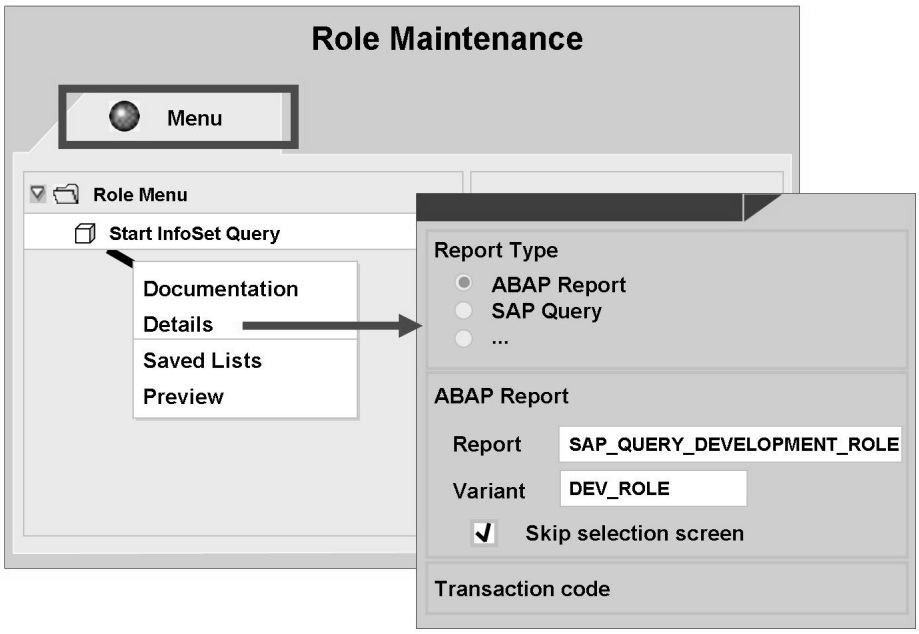


Figure 100: Entering the InfoSet Query in the Menu

There are four programs that you can use to call up the InfoSet query and to determine the access authorizations and type of reporting (ad-hoc or development). These programs display a selection screen that it is best to fill using a variant and then skip when you call up the query.

***SAP_QUERY_DEVELOPMENT_ROLE*:** Access using role, development (that is, global area)

- Selection screen parameters: Role (mandatory), InfoSet (optional), query (optional).
- You have to a user group from the global query area to the role. If you specify further parameters, make sure these are appropriate.

SAP_QUERY_AD_HOC_ROLE: Access from role, ad-hoc reporting (that is, standard area).

- Selection screen parameters: Role (mandatory), InfoSet (optional), query (optional). A user group that comes from the standard work area must be assigned to the role. If you specify further parameters, make sure these are appropriate.
- If you specify a start query, this is used as a template in the InfoSet Query.

SAP_QUERY_DEVELOPMENT: Access from user group, development.

SAP_QUERY_AD_HOC: Access from user group, ad-hoc reporting

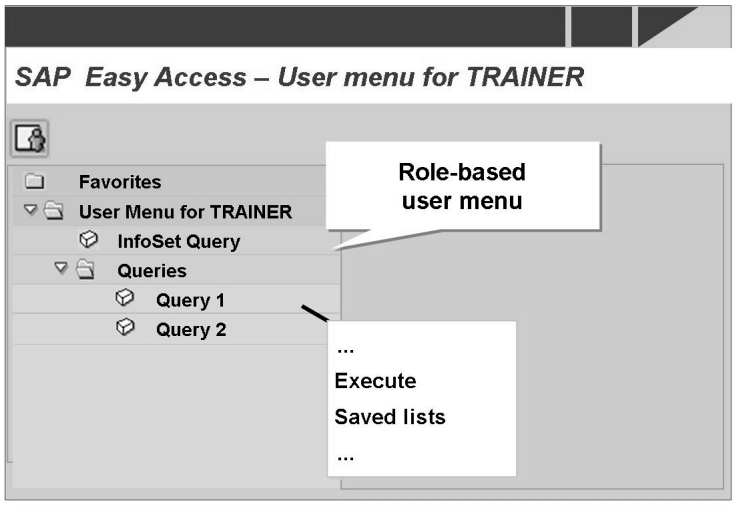


Figure 101: Starting Queries from the User Menu

When the user starts their user menu, he or she sees an InfoSet Query which corresponds to the menu entries that you made in the role maintenance.

The user can also read the saved lists for a query here.

Authorization Object S_QUERY

S_QUERY (ACTVT)	**Activate the users**
Change (02)	Changing and creating queries
Maintain (23)	Maintaining user groups and InfoSets
Change (02) Maintain (23)	Superuser
Translate (67)	Language comparison for query

You use the authorization object *S_QUERY* for the InfoSet and SAP Query. The system checks this object in the four standard programs *SAP_QUERY_DEVELOPMENT_ROLE, SAP_QUERY_AD_HOC_ROLE, SAP_QUERY_DEVELOPMENT* and *SAP_QUERY_AD_HOC*.

Using the field *ACTVT* you can determine the activities that the user can do in conjunction with the query. With the value *ACTVT = 02* a user is allowed to create queries and change existing queries. Authorization to maintain InfoSets is restricted, so that a user who wants to insert ABAP coding in an InfoSet can only do this if the user has authorization for the authorization object *S_DEVELOP* with the value *'PROG'* for the field *OBJTYPE* and the value *'AQ*'* for the field *OBJNAME*.

You require *ACTVT = 23* to maintain InfoSets or user groups, that is, for typical administration tasks. If these activities are combined, a user is allowed to do everything in principle, that is, the user is a superuser in principle.

A separate authorization (*ACTV = 67*) is required to translate the query objects.

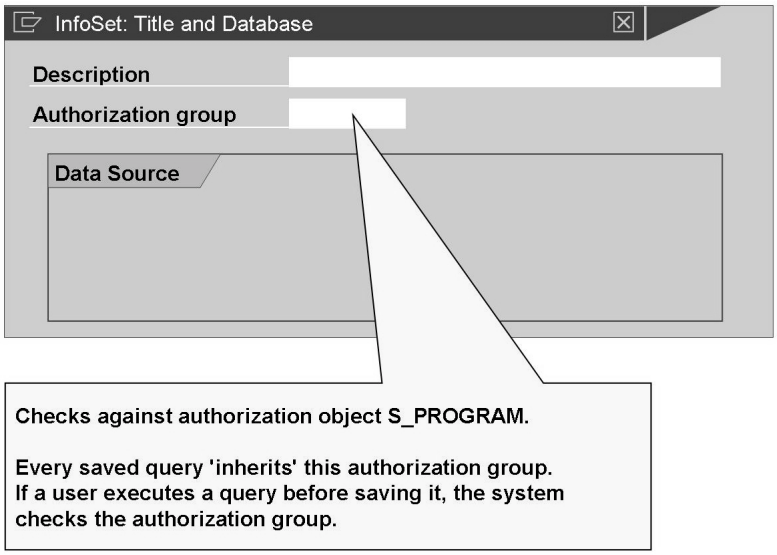


Figure 102: InfoSet: Assigning Authorization Groups

You can specify that an authorization group is assigned to every query in an InfoSet. The system carries out the check using authorization object *S_PROGRAM*.

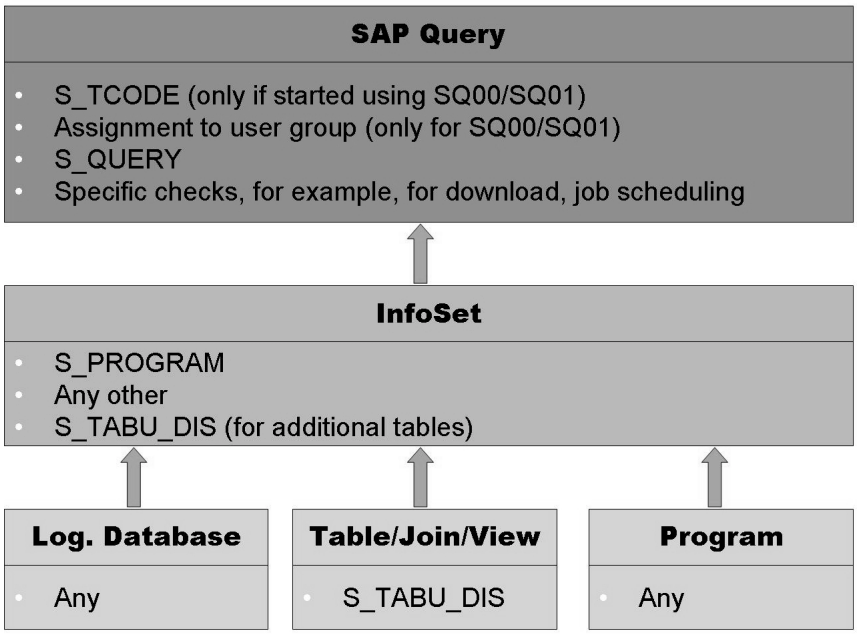


Figure 103: Authorization Checks when you Execute an SAP Query

A logical database (LDB) developer can implement any authorization checks in a LDB. These authorization checks are always executed when a query uses this LDB. This is the case if the InfoSet of the Query is based on this LDB.

If an InfoSet is based on an individual data retrieval program, all authorization checks that are implemented in this program run if a query is called that is based on this InfoSet.

In the additional source code of this InfoSet, you can implement any additional checks (for example, for program entry point *AT SELECTION-SCREEN*).

The user does not need authorization for authorization object *S_QUERY* to execute a query. *S_QUERY* controls whether a user is allowed to display or change a **query definition**.

You can execute a query in the following ways:

- You can start it using a report name *AQ...*, for example, in transaction SA38.
- You can start it using a specifically created report transaction. You can create this in the ABAP Workbench and it calls the report generated from the query definition. However, this is only possible for queries from the global area since transactions are always defined independently of the client.
- You can start it using an entry in the role menu: In the menu of a role (transaction PFCG), you can call a query directly (without specifying the name of the generated report).

If you use one of these options, you do not need to include the user in a query user group (transaction SQ03). However, the user requires an authorization for each report transaction that you create specifically (authorization object *S_TCODE*).

In addition, you can include a user in a user group without giving them authorization for authorization object *S_QUERY*. In this case, even though the user can call queries in their user group in transaction SQ01, they cannot display or change the definitions. Transaction SQ00 is also of interest here. In this transaction, the user is not even offered editing functions for a query, for example, *Change*, *Delete* or *Copy*, rather they see only a list of the queries in their user group. (If a user has authorization for S_QUERY, transaction SQ00 looks exactly the same as transaction SQ01.)

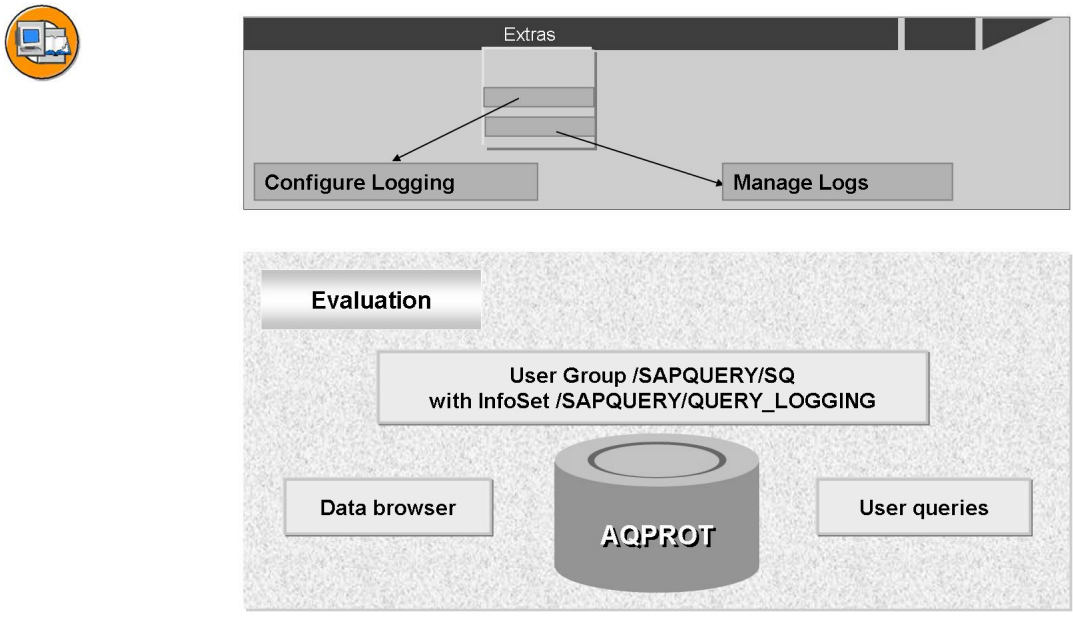


Figure 104: Logging

You can log work with the **InfoSet Query**. To do this you have to connect an InfoSet to the logging function. You can do this in maintenance transaction SQ02 from the *Extras* menu.

If necessary, you can write the following specifications to database table *AQPROT* each time a query is processed:

- Query area and InfoSet
- User, date and time
- All selection fields and the values for selection
- All output fields

You can find a similar function in Customizing. The function leads to a table control in which you can maintain entries for database table *AQPROTCUST*. The database table contains the fields Query Area (possible values are *SPACE* for the standard area and *G* for the global area) and InfoSet.

You can delete logs in the maintenance transaction SQ02 from the *Extras* menu by choosing Manage logs. You reach a selection screen in which you, for example, restrict the selection to a user, which means that you can delete all logs for this user.

Exercise 13: Creating User Groups and Recording Logs (optional)

Exercise Objectives

After completing this exercise, you will be able to:

- Create a user group
- Assign InfoSets to the user group
- Activate and read the logs

Business Example

Task 1:

1. Create a user group BC407_## in the global query area.
2. Assign the InfoSets you created in the exercise in the unit *Creating InfoSets* to the user group.

Task 2:

Open an InfoSet from the exercise in the unit *Creating InfoSets* in the log mechanism. This InfoSet should have (at least) one query.

1. Execute this query.
2. Use the Data Browser (transaction SE16) to verify that the log records have been recorded.

Solution 13: Creating User Groups and Recording Logs (optional)

Task 1:

1. Create a user group BC407_## in the global query area.
   a) Start transaction SQ03 and create the user group BC407_## in the global query area.
2. Assign the InfoSets you created in the exercise in the unit *Creating InfoSets* to the user group.
   a) Choose Assign Users and InfoSets. Assign the InfoSets you created in the exercise in the unit *Creating InfoSets*.

Task 2:

Open an InfoSet from the exercise in the unit *Creating InfoSets* in the log mechanism. This InfoSet should have (at least) one query.

1. Execute this query.
   a) Change to transaction SQ02. Choose *Extras* → *Set Logs*. Choose *New Entries*. Enter the query area G and the name of your InfoSet that has a query.
   b) Start the InfoSet Query and refresh the data.
2. Use the Data Browser (transaction SE16) to verify that the log records have been recorded.
   a) Read the log using transaction SE16.

Lesson Summary

You should now be able to:

- Manage users for SAP Queries and InfoSet Queries
- Set up the user menu so that the user can access the InfoSet Query

Unit Summary

You should now be able to:

- Manage users for SAP Queries and InfoSet Queries
- Set up the user menu so that the user can access the InfoSet Query

Internal Use SAP Partner Only

Unit 9

Transporting Query Components

Unit Overview

In this lesson, you will learn how you can transport query components from the development system to follow-up systems, how you can transport query components from the standard area to other clients, and how you can copy query components between the standard area and the global area.

Unit Objectives

After completing this unit, you will be able to:

- Transporting Query Components

Unit Contents

Lesson: Transporting Query Components

Lesson Overview

In this lesson, you will learn how you can transport query components from the development system to follow-up systems, how you can transport query components from the standard area to other clients, and how you can copy query components between the standard area and the global area.

Lesson Objectives

After completing this lesson, you will be able to:

- Transporting Query Components

Business Example

You have been asked to transport query components from the standard area of your development client to another client of the same system.

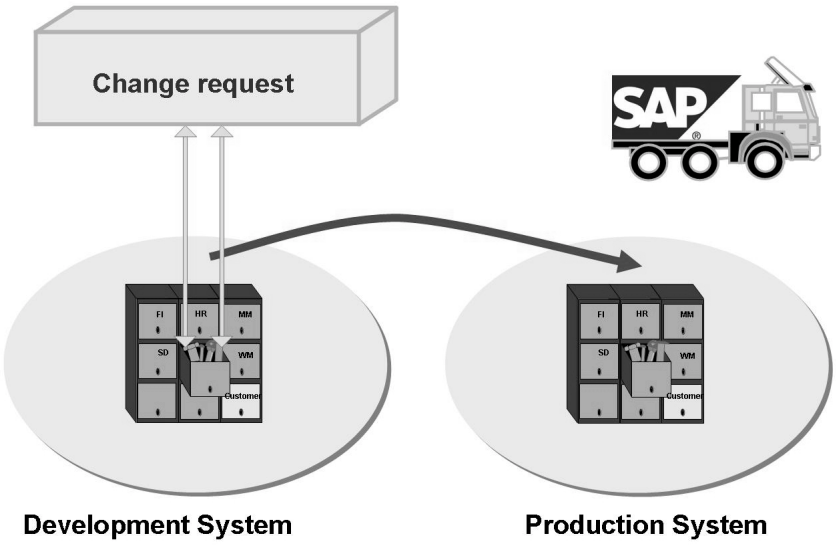


Figure 105: Transporting Query Components

You create program objects (technical term: repository objects) in a development system and then transport them to receiving systems.

The system assigns query components in the global query area to a transport system automatically by classifying them in a package along with their usual, logical classification.

A deciding factor for grouping together development tasks is which query components have to be transported together due to their dependencies.

You organize common transports of query components using a change request.

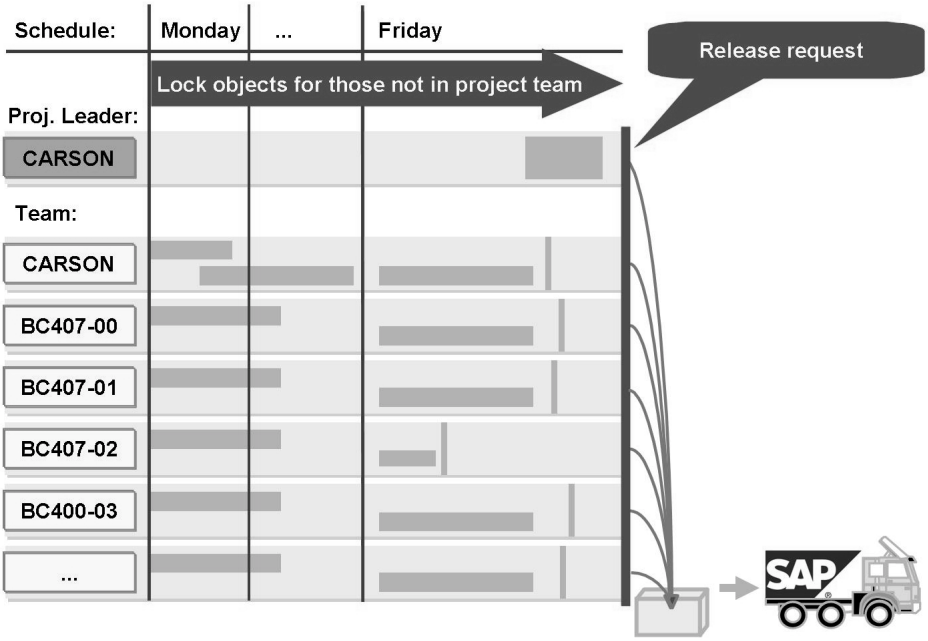


Figure 106: Development Project: Schedule

Scheduling a development project using a change request covers the following aspects:

- All team members being at a fixed time.

Each team member can trace and check his or her activities separately.

- All project team members can edit the affected programming objects. For those developers who do not belong to the team, the repository objects remain locked until the project is completed.
- All programming objects affected are triggered to be transported together.

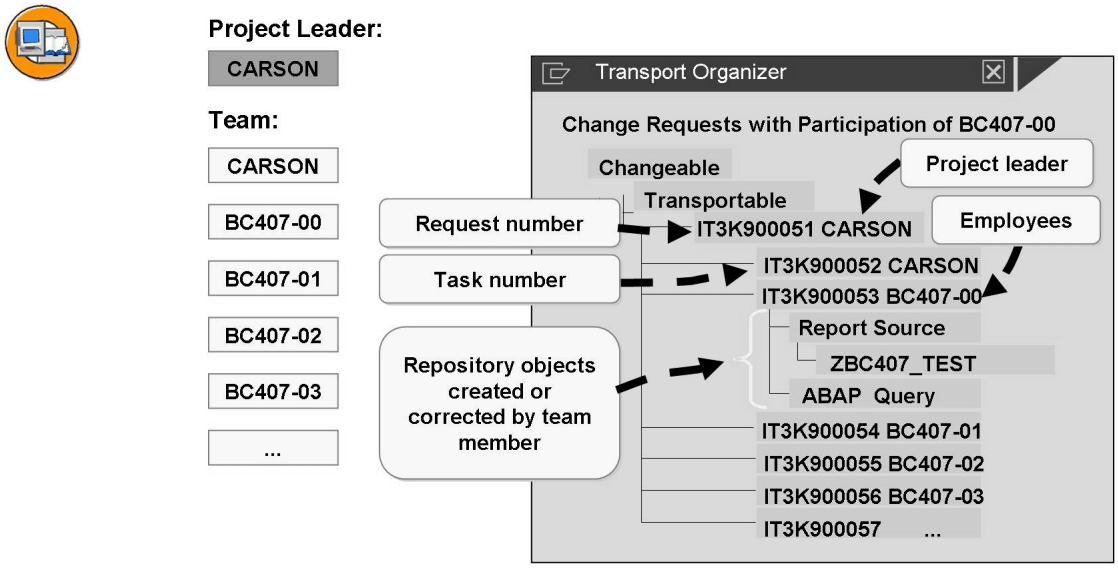


Figure 107: Development Projects in the Transport Organizer

When a project leader starts a project, he or she creates a change request in the Transport Organizer (transaction SE09). The project leader assigns the project team members to the change request and creates a task for each developer.

If a team member assigns a programming object to this change request (for example, the first time the team member saves the project), it is entered into that team member's tasks. In this way, all repository objects that an employee edits during the development project are collected in the task. The repository objects are therefore locked for all developers who do not belong to this particular project.

Unlike the logical selection of objects through development classes, the change log deals with a project-related (and therefore time-related) selection. While a program or a query always belongs to a development class, it can belong to different projects at different times.

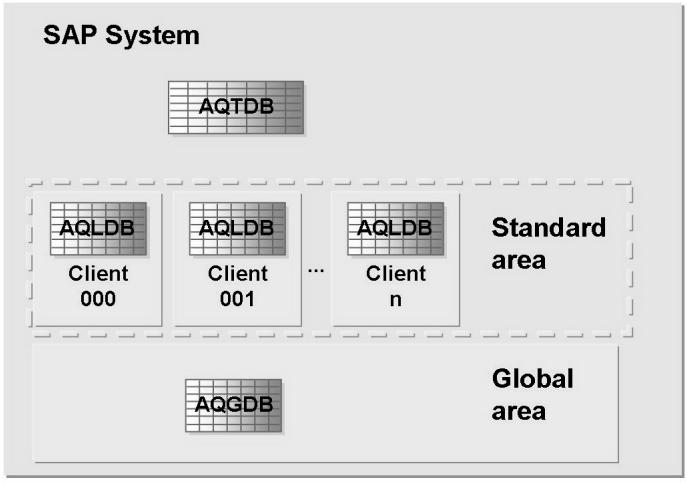


Figure 108: Database Tables for the Different Areas

To be able to understand each of the transport options, you have to know how the object is stored on the database.

The system stores the objects from the standard area in the client-dependent table ***AQLDB***. The objects from the global query area are in the table ***AQGDB***.

Objects in the global area are assigned to development classes. You transport these objects using the Transport Organizer.

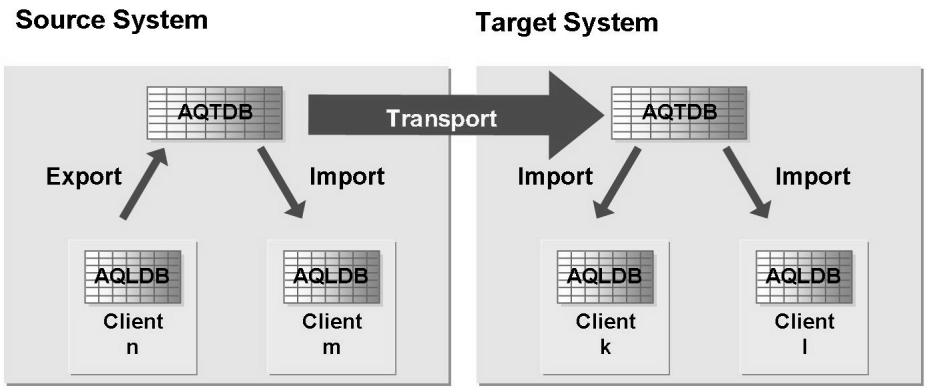


Figure 109: Transporting Using Export/Import

To transfer objects from the standard area into another standard area in the same or a different SAP system, create the relevant client-independent objects. You do this either by copying them into the global area or exporting them to the table *AQTDB*. This assigns the objects to a transport request. You can either copy the objects in the target system from the global area into the target client or import them from the table *AQTDB*.

In the initial screen of transactions SQxx, choose *Transports*. You can also start the program *RSAQR3TR* directly. You can carry out the following actions:

- Export: Create a transport dataset in the table *AQTDB*.
- Import: Read a transport dataset from table *AQTDB*.
- Display: Display the transport dataset in table *AQTDB*.
- Delete: Delete the transport dataset from table *AQTDB*.
- Copy *Standard Area* → *Global Area*: Copy objects from table *AQLDB* to table *AQGDB*.
- Copy *Global Area* → *Standard Area*: Copy objects from table *AQGDB* to table *AQLDB*.

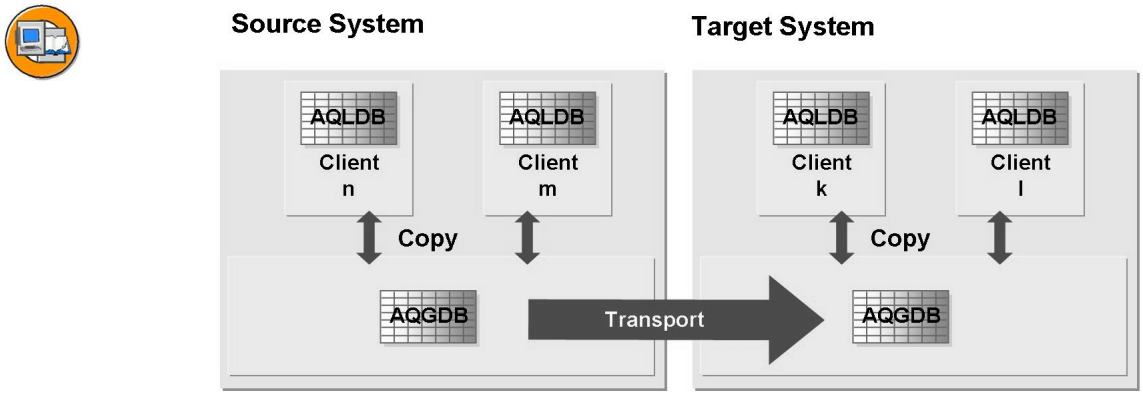


Figure 110: Transporting in the Global Area

The second option is copying the objects from the standard area into the global query area. Copying objects between different query areas is the same as transporting objects between two system clients. Therefore, copying is seen as a transport in the standard area.

The transport action Copy *Standard Area* → *Global Area* copies objects from the standard area to the global area. This is the same as exporting objects from the standard area and importing them into the global area. Note the following regarding packages for copied objects in the global area:

- If an object in the global area is overwritten, the copied object inherits the package of the overwritten object.
- If the copied object in the global area is a new object, a window usually appears, in which you have to assign a package to the object.

Exercise 14: Transporting Between the Global Area and the Standard Area (optional)

Exercise Objectives

After completing this exercise, you will be able to:

- Transport query components

Business Example

You want to transport query components (user groups, InfoSets and queries).

Task:

1. Start the transport from transaction SQ02 or start the program RSAQR3TR.
2. Copy the user group BC407_## from the exercise in *User Administration* from the global area to the standard area. First do a test run. If you get the return code 0, make the copy.
3. Copy the InfoSets BC407##_01 and BC407##_02 from the global area into the standard area. Also copy the queries created for these InfoSets. First do a test run. If you get the return code 0, make the copy.
4. Note the *Information* button that is offered in the transport program. You can read about the different transports modes (for example, REPLACE) in detail.

Solution 14: Transporting Between the Global Area and the Standard Area (optional)

Task:

1. Start the transport from transaction SQ02 or start the program RSAQR3TR.

   a) Start the transport from transaction SQ02 or start the program RSAQR3TR (transaction SE38). In Transport Actions, choose *Copy Global Area → Standard Area*. Mark the *Test Run* checkbox.

2. Copy the user group BC407_## from the exercise in *User Administration* from the global area to the standard area. First do a test run. If you get the return code 0, make the copy.

   a) Choose *Transport User Groups* and enter the name of your user group. Select the import option REPLACE. If the return code 0 is sent, remove the mark from the *Text Run* checkbox and execute the transport.

3. Copy the InfoSets BC407##_01 and BC407##_02 from the global area into the standard area. Also copy the queries created for these InfoSets. First do a test run. If you get the return code 0, make the copy.

   a) Turn *Test Run* on again. Execute the same actions for your InfoSets and queries.

4. Note the *Information* button that is offered in the transport program. You can read about the different transports modes (for example, REPLACE) in detail.

   a) Read the documentation provided by choosing *Information*.

Lesson Summary

You should now be able to:

- Transporting Query Components

Unit Summary

You should now be able to:

- Transporting Query Components

Course Summary

You should now be able to:

- Create and execute QuickViews
- Create and execute SAP Queries
- Create and execute InfoSet Queries
- Maintain InfoSets
- Administer users for SAP Queries and InfoSet Queries

Internal Use SAP Partner Only

Appendix 1

Appendix

Transaction Codes

Transaction code	Description
SQVI	QuickViewer
SQ01	Queries
SQ02	InfoSets
SQ03	User groups
PFCG	Role maintenance
SE09	Transport Organizer
SE36	Logical Database Builder
SM36	Define job
SE80	Object Navigator

Create packages

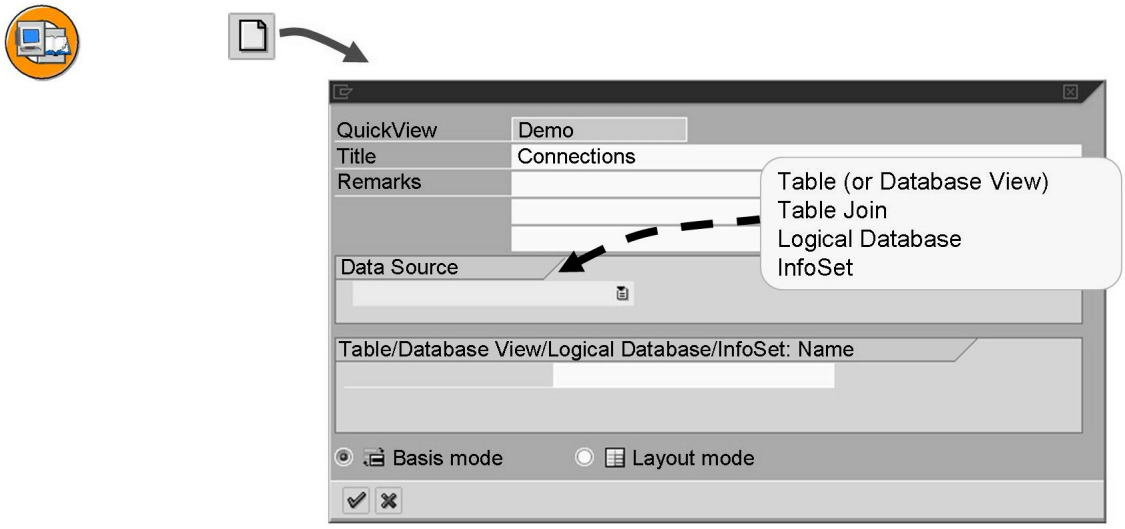


Figure 111: Create Package I

Packages are logical groups of programming objects that are to be transported. The customer namespace beings with Z or Y.

You create packages in the Object Navigator (transaction SE80). From the selection list on the left-hand side of the screen, choose *Package* and in the field below, enter a name for your package. Choose enter. If the package already exists, the system displays its properties. Otherwise, the system asks you if you want to create the package. Confirm the question. Another input template screen appears.

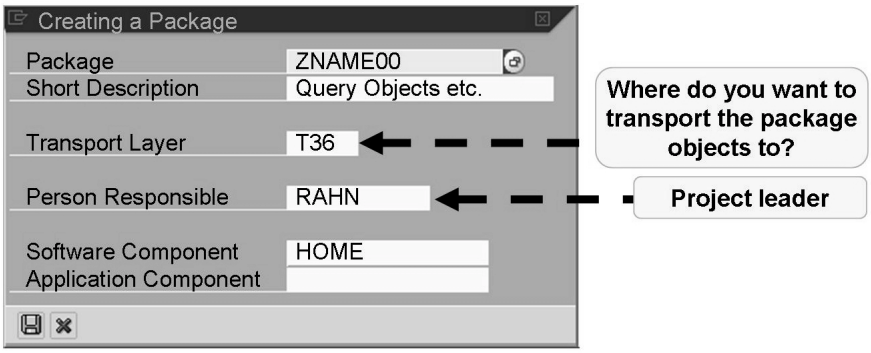


Figure 112: Create Package II

Check the default settings for the following attributes:

- **Transport Layer:** If you are performing customer development, you have to set up a transport layer for customer developments.
- **Person responsible:** This person has full responsibility for the objects in this package. Your user name is entered as a default value.
- **Software component:** You usually enter *HOME* for customer developments here. You can see more detailed information using the F1 Help.
- **Application component:** Determine a classification for the package in the application hierarchy and therefore its business context. You can find this later in transaction SE81.

When you save your entries, you have to determine a temporal assignment, that is a **change request**. Proceed here exactly as when you saved the query objects in the global query area.

Creating variants

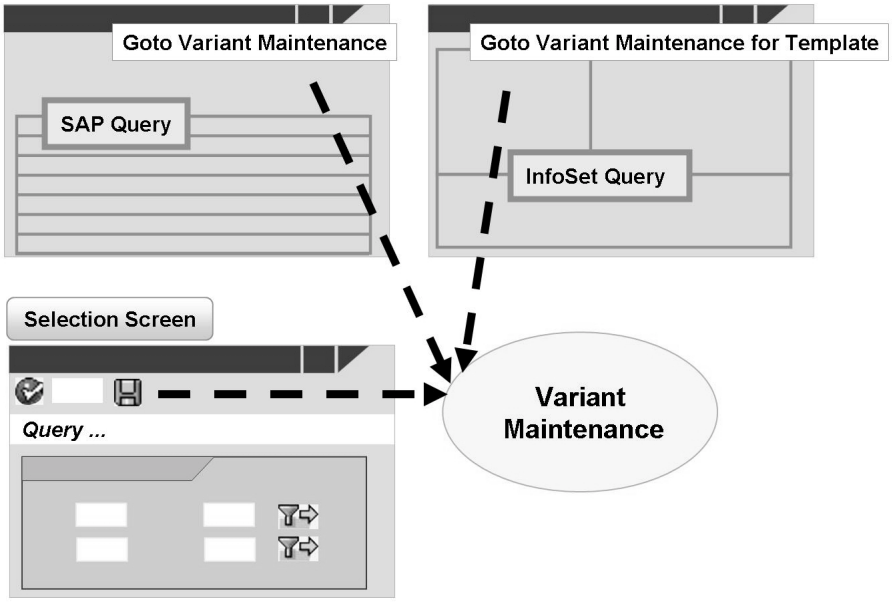


Figure 113: Selection Screen: Creating Variants

You use variants to store selection variants. This means that you can start a query from these variants and do not have to fill in the contents of the selection screen in the dialog box.

Variants are required for executing queries in the background.

You can set variants only for selection screens and not for the selection screen control in the InfoSet Query. If you want to access a variant from the InfoSet Query, you have to execute it using *Output* or in transaction SQ01.

If you are creating variants from the selection screen, first enter the values in the selections and save the selection screen as a variant. You have to maintain variant attributes or the individual selections.

If you got to the variant maintenance from the SAP or InfoSet Query, you have to specify the value **and** the attribute.

Background Processing

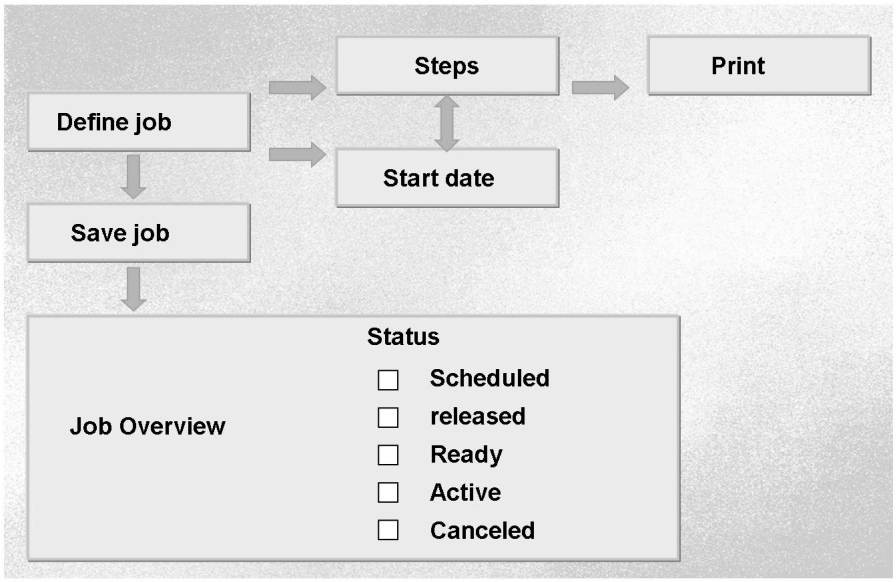


Figure 114: The Phases of Background Processing

When a program converts large datasets and requires a long runtime, it makes sense to start it in the background.

Background runs take place without user dialogs, and can take place in parallel to online operations. The dialog work processes are available for online processing. Background job runs are performed by special work processes (background processes), which enables distributed processing.

To start a program in the background, you must first add it to a job.

Use the job definition to determine which programs (steps) will run during this job. You can specify print parameters and set the start time for the job.

The job overview (transaction SM37) tells you the current status of the job.

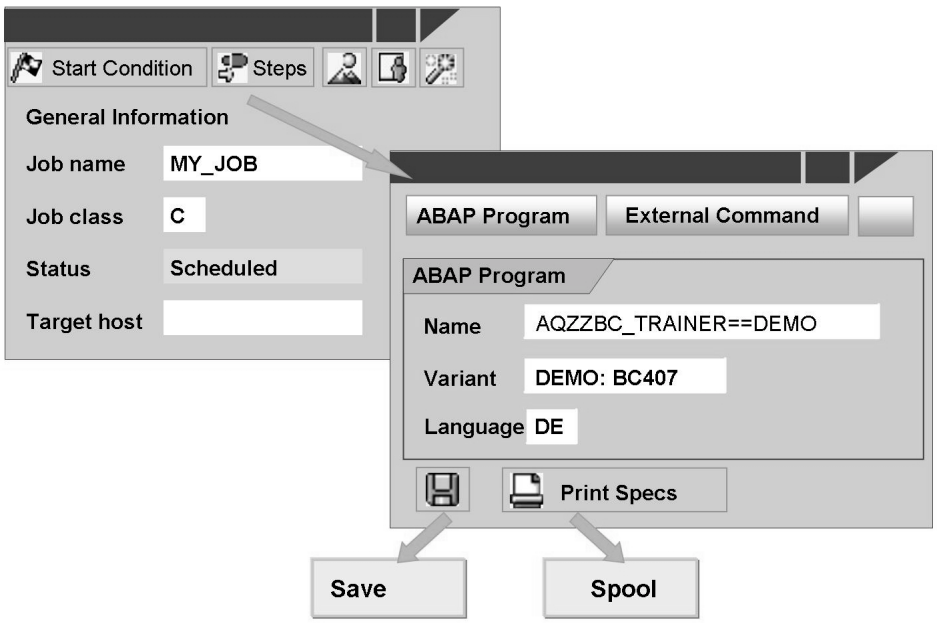


Figure 115: Defining Steps

You can find the *Define Job* by following the menu path *System → Services → Jobs → Define Jobs* (transaction SM36). For queries, it is easiest to schedule jobs in the initial screen of transaction SQ01, using the *Execute in Background* pushbutton. You can also schedule QuickViews from your initial screen (transaction SQVI).

First assign a name (of your choice) and determine the priority (job class) and target host (using input help).

Then determine the individual steps of the job. If you want the program to run with a selection screen, you also have to specify a variant. The list can be stored in the spool or printed immediately. This depends on the specified print parameters. When you have defined all the steps, save them and return to the initial screen of the job definition.

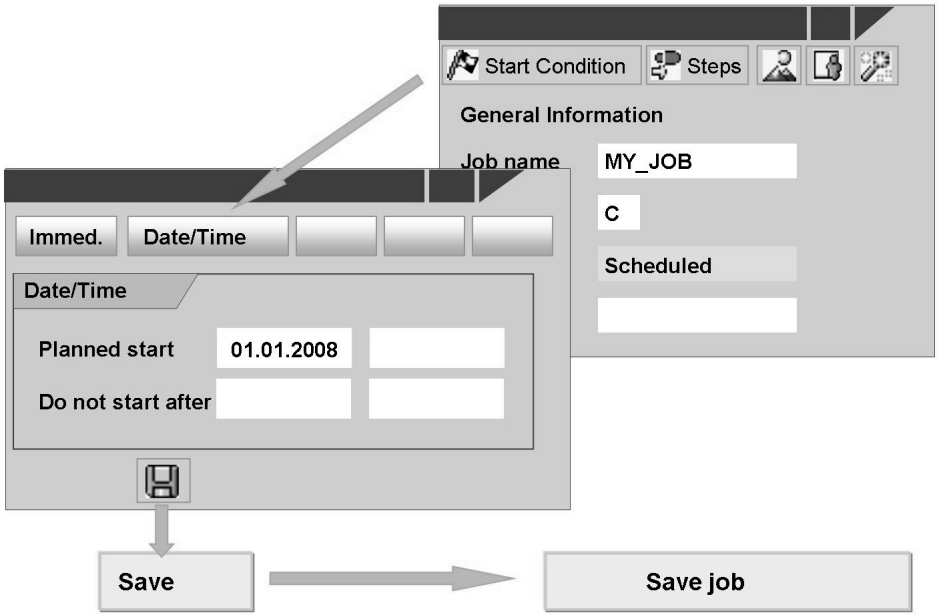


Figure 116: Determining the Start Date and Releasing the Job

Once you have defined the steps, determine the start date of the job. For example, you can start a job on a specific day at a specific time.

Once you have defined the start date, save your entries and return to the initial screen of the job definition. Now save the job, which releases it to run at the specified time.

You can also use the automated job scheduling with the function modules in function groups *BTCH* and *BTC2*.

Index

Internal Use SAP Partner Only

Feedback

SAP AG has made every effort in the preparation of this course to ensure the accuracy and completeness of the materials. If you have any corrections or suggestions for improvement, please record them in the appropriate place in the course evaluation.

Internal Use SAP Partner Only