OBJECTIVE

- Definition
- Tables
- Views
- Domains
- Search Helps
- Locks

The ABAP Dictionary centrally describes and manages all the data definitions used in the system. The ABAP Dictionary is completely integrated in the ABAP Development Workbench. All the other components of the Workbench can actively access the definitions stored in the ABAP Dictionary.

The most important object types in the ABAP Dictionary are :-

1> Tables

2> Views

3> Types (data elements, structures, table types)

4> Domains

5> Search helps

6> Lock objects

Tables are defined in the ABAP Dictionary independently of the database. A table having the same structure is then created from this table definition in the underlying database.

Views are logical views on more than one table. The structure of the view is defined in the ABAP Dictionary. A view on the database can then be created from this structure.

Types are used in ABAP program. The structure of a type can be defined globally in ABAP programs. Changes to a type automatically take effect in all the programs using the type.

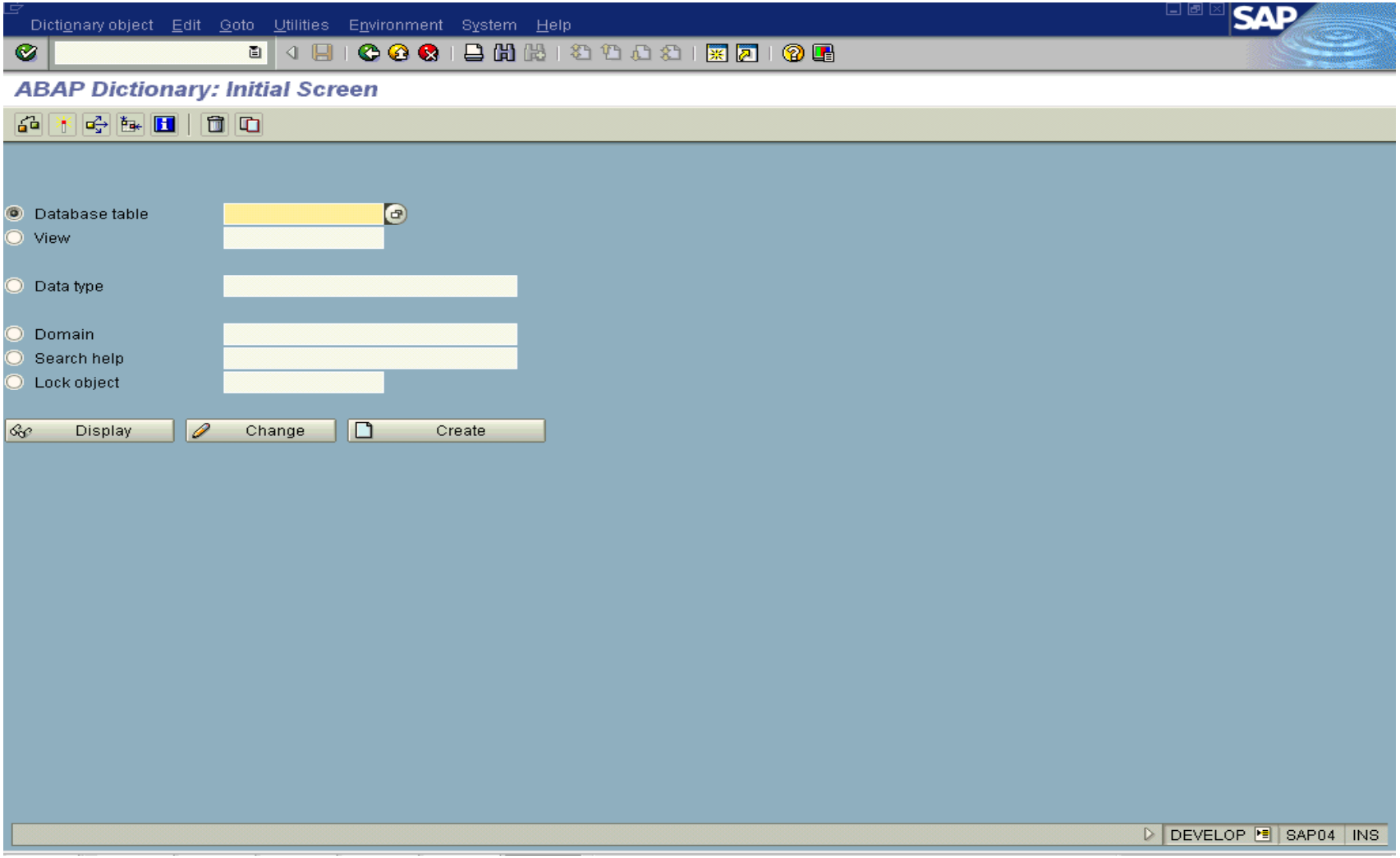
Lock Objects are used to synchronize access to the same data by more than one user. Function modules that can be used in application programs are generated from the definition of a lock object in the ABAP Dictionary.

Different fields having the same technical type can be combined in domains. A domain defines the value range of all table fields and structure components that refer to this domain.

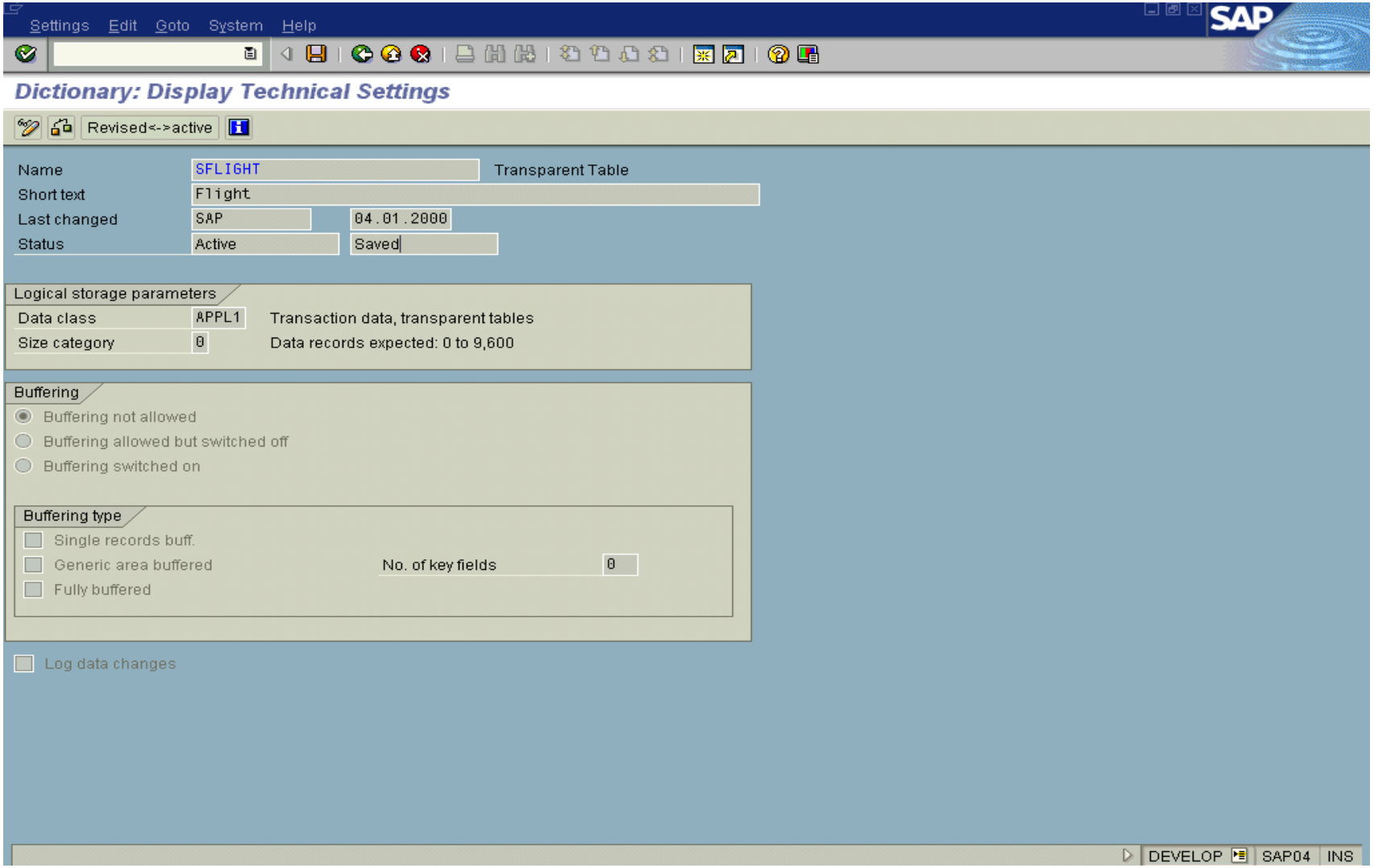
The ABAP Dictionary also contains the information displayed with the F1 and F4 help for a field in an input template.

The documentation about the field is created for a data element that describes the meaning of the contents of a table field.

The list of possible input values that appears for the input help is created by a foreign key or a search help.



ABAP Dictionary (Transaction se11): Initial Screen



Technical Settings

Data Class

If you choose the data class correctly, your table is automatically assigned to the correct area (tablespace or DBspace) of the database when it is created. Each data class corresponds to a physical area in which all the tables assigned to this data class are stored.

There are the following data classes:

APPL0 (master data): Data which is seldomly changed. An example of master data is the data contained in an address file, such as the name, address and telephone number.

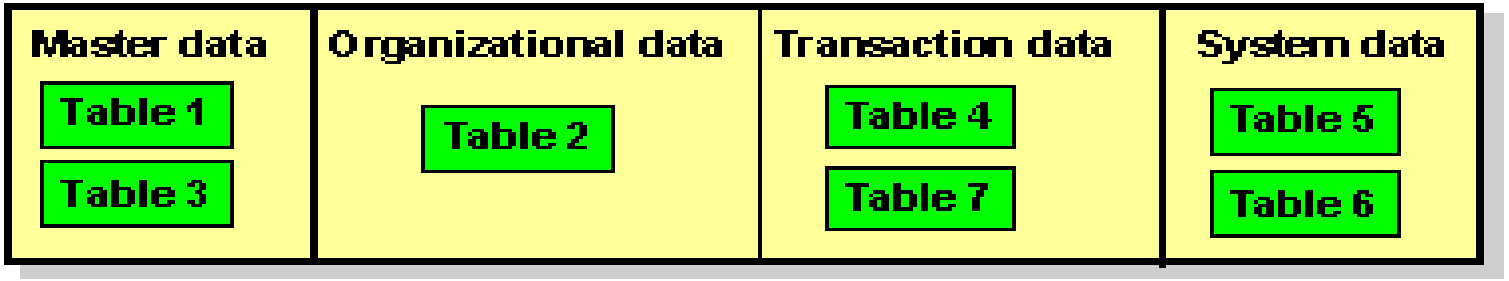
APPL1 (transaction data): Data that is frequently changed.

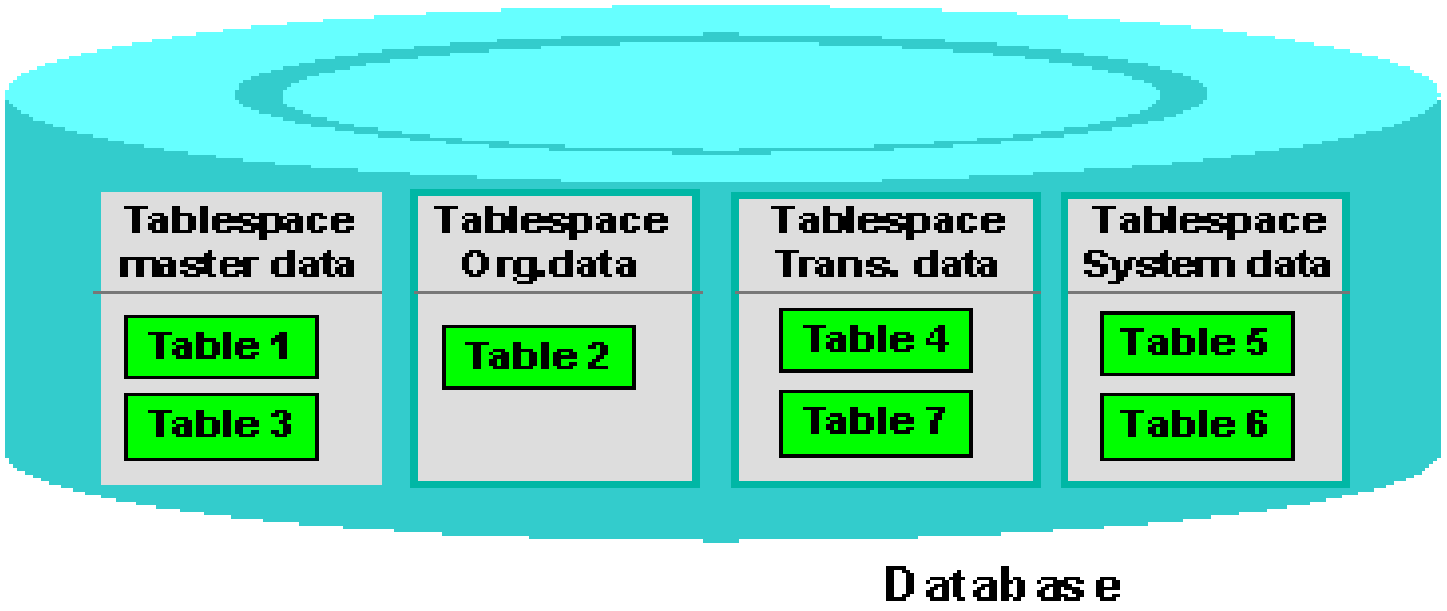
An example of transaction data is the goods in a warehouse, which change after each purchase order.

APPL2 (organizational data): Customizing data that is defined when the system is installed and seldomly changed.
An example is the table with country codes.

Two further data classes, USR and USR1, are provided for the customer. These are for user developments. The tables assigned to these data classes are stored in a tablespace for user developments.

Tables in the ABAP Dictionary



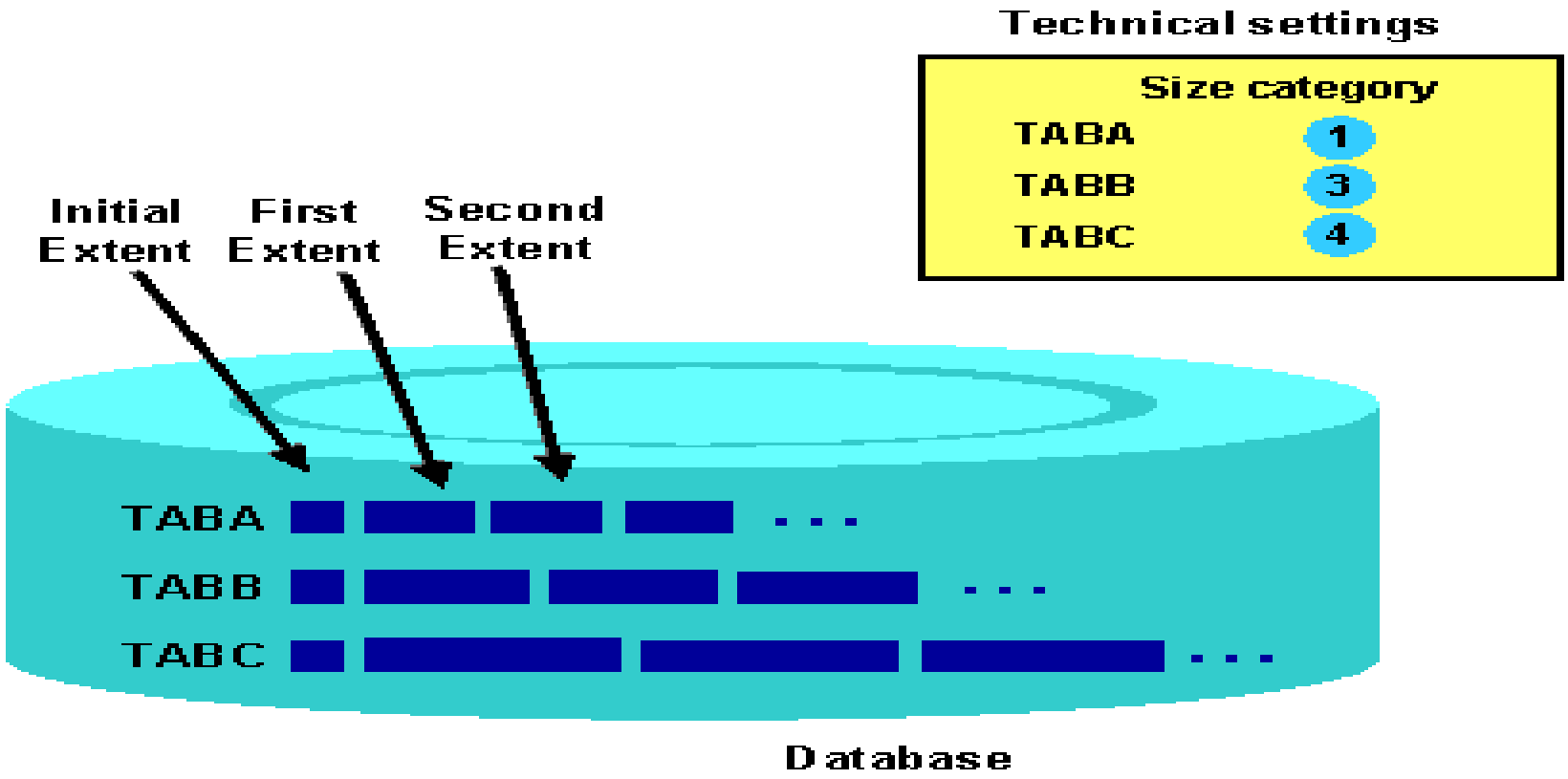


Size Category

The size category defines the expected space required for the table in the database. You can choose a size category from 0 to 4 for your table.

Each category is assigned a certain fixed memory size in the database, which depends on the database system used.

When a table is created, initial space (an Initial Extent) is reserved in the database. If more space is required at a later time due to data entries, additional memory will be added depending on the selected size category.



Selecting the correct size category prevents a large number of very small extents from being created for a table. It also prevents space from being wasted if extents which are too large are created.

Buffering

You must define whether and how a table is buffered in the technical settings for the table. There are three possibilities here:

1> Buffering not permitted: Table buffering is not permitted, for example because application programs always need the most recent data from the table or the table is changed too frequently.

2> Buffering permitted but not activated: Buffering is permitted from the business and technical points of view. Applications which access the table execute correctly with and without table buffering.

Whether or not table buffering will result in a gain in performance depends on the table size and access profile of the table (frequency of the different types of table access).

Table buffering is deactivated because it is not possible to know what these values will be in the customer system. If table buffering would be advantageous for the table size and access profile of the table, you can activate it in the customer system at any time.

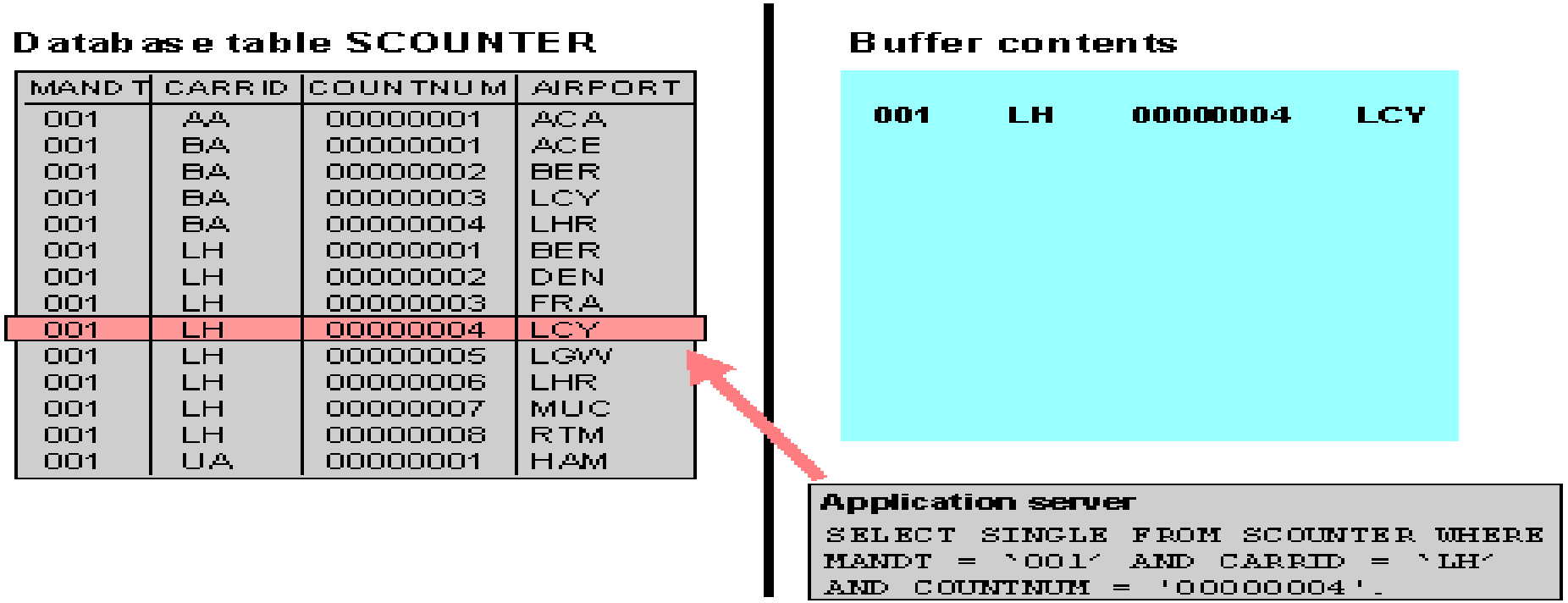
3> Buffering activated: The table should be buffered. In this case you must specify a buffering type.

Buffering types:

1> Single-Record buffering

With single-record buffering, only the records that are actually read are loaded into the buffer. Single-record buffering therefore requires less storage space in the buffer than generic and full buffering. The administrative costs in the buffer, however, are greater than for generic or full buffering. Considerably more database accesses are necessary to load the records than for the other buffering types.

In this example, the record highlighted in red is read by a program from table SCOUNTER. If single-record buffering is selected for the table, only the record that was read is loaded into the buffer.



When Should you Use Single-Record Buffering?

Single-record buffering should be used particularly for large tables where only a few records are accessed with SELECT SINGLE.

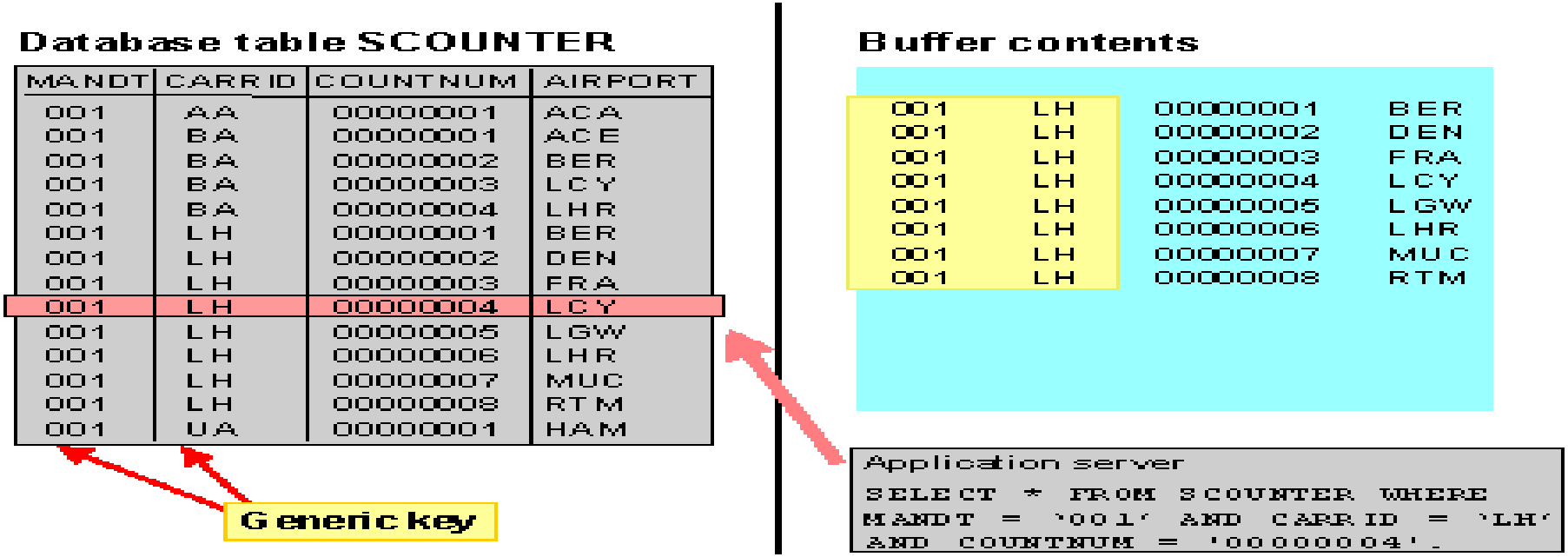
The size of the records being accessed should be between 100 and 200 KB.

Full buffering is usually more suitable for smaller tables that are accessed frequently. This is because only one database access is necessary to load such a table with full buffering, whereas several database accesses are necessary for single-record buffering.

2> Generic buffering

With generic buffering, all the records in the buffer whose generic key fields match this record are loaded when one record of the table is accessed. The generic key is a part of the primary key of the table that is left-justified.

In this example, the record highlighted in red is read by a program from table SCOUNTER. If the table is generically buffered, all the records read whose generic key fields (MANDT and CARRID) agree are loaded into the buffer.



When Should you Use Full Buffering?

A table should be buffered generically if only certain generic areas of the table are normally needed for processing.

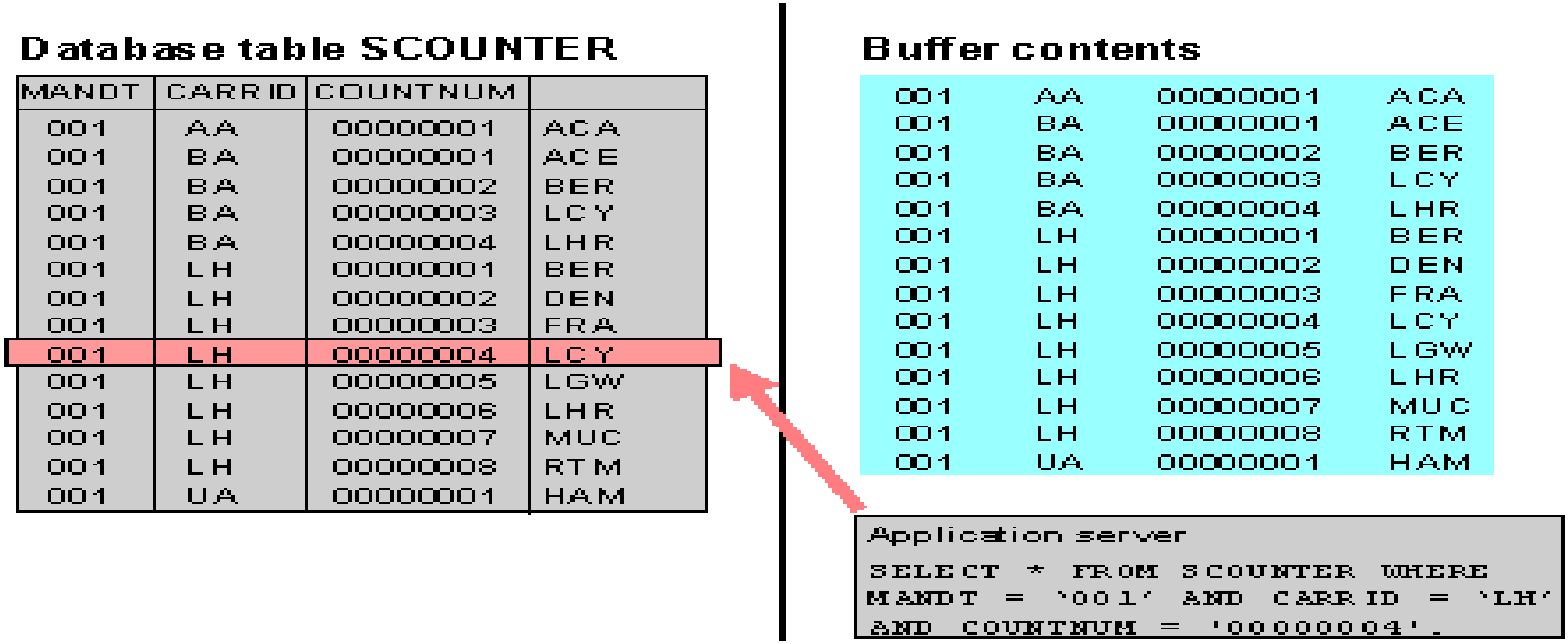
Client-specific, fully-buffered tables are automatically generically buffered since normally it is not possible to work in all clients at the same time on an application server. The client field is the generic key.

Language-specific tables are another example where generic buffering is recommended. In general, only records of one language will be needed on an application server. In this case, the generic key includes all the key fields up to and including the language field.

3> Full buffering

With full buffering, either the entire table is in the buffer or the table is not in the buffer at all. All the records of the table are loaded into the buffer when one record of the table is read.

In this example, a program reads the record highlighted in red from table SCOUNTER. If the table is fully buffered, all the records of the table are loaded into the buffer.



When Should you Use Full Buffering?

When deciding whether a table should be fully buffered, you should take into account the size of the table, the number of read accesses, and the number of write accesses. Tables best suited to full buffering are small, read frequently, and rarely written.

Full buffering is recommended in the following cases:

Tables up to 30 KB in size. If a table is accessed frequently, but all accesses are read accesses, this value can be exceeded. However, you should always pay attention to the buffer utilization.

Larger tables where large numbers of records are frequently

accessed. If these mass accesses can be formulated with a very selective WHERE condition using a database index, it could be better to dispense with buffering.

Tables for which accesses to non-existent records are frequently submitted. Since all the table records reside in the buffer, the system can determine directly in the buffer whether or not a record exists.

Logging

The logging flag defines whether changes to the data records of a table should be logged. If logging is activated, every change (with UPDATE, DELETE) to an existing data record by a user or an application program is recorded in a log table in the database.

Note: Activating logging slows down accesses that change the table. First of all, a record must be written in the log table for each change. Secondly, many users access this log table in parallel. This could cause lock situations even though the users are working with different application tables.

Dependencies
Logging only takes place if parameter rec/client in the system profile is set correctly. Setting the flag on its own does not cause the table changes to be logged.

The existing logs can be displayed with Transaction Table history (SCU3).

Delivery class

The delivery class controls the transport of table data for installation ,upgrade, client copy and when transporting between customer systems. The delivery class is also used in the extended table maintenance.

There are the following development classes:

A: Application table (master and transaction data).

C: Customer table, data is only maintained by the customer.

L: Table for storing temporary data.

G: Customer table, SAP may insert new data records but may not overwrite or delete existing ones.

E: System table with its own namespace for customer entries.

The customer namespace must be defined in table TRESC.

S: System table, data changes have the status of program changes.

W: System table (e.g. table of the development environment) whose data is transported with its own transport objects

(e.g. R3TR PROG, R3TR TABL, etc.).

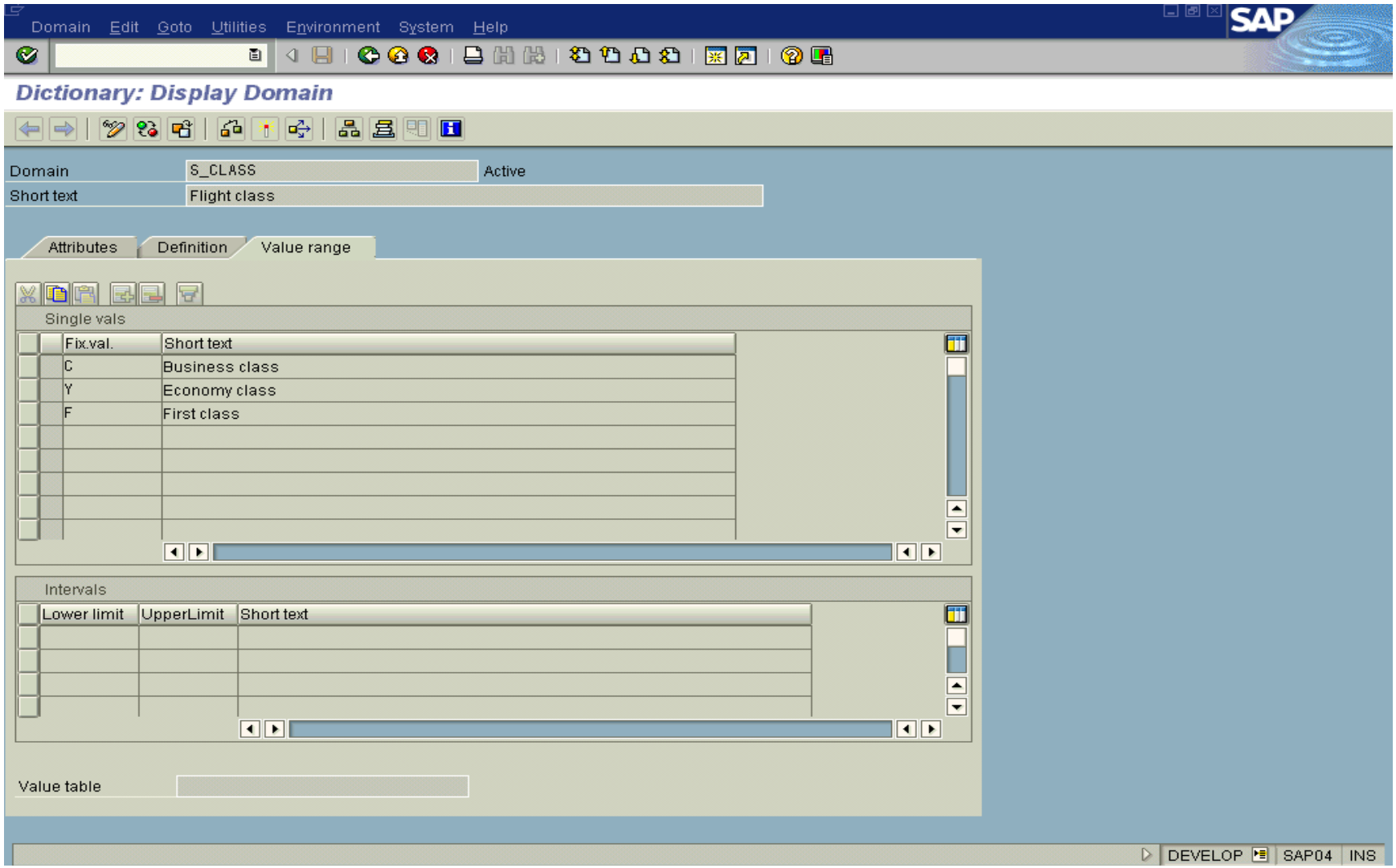
Fixed values for a domain

The value range of a domain can be further restricted by defining fixed values. If fixed values are defined for a domain, these are used in the input check in screen templates. If no other means of help is defined for a field (search help,foreign key), the fixed values are also offered in the input (F4) help.

Domain S_CLASS (data type CHAR, length 1) in the flight model describes the possible classes of a flight booking. The value range of domain S_CLASS is defined by the fixed values C (business class), F (first class) and Y (economy class). Only the values C, F and Y may be entered in screen templates for all the fields that refer to this domain.

You can define fixed value intervals either by entering upper and lower limits or by specifying single values. Value ranges and single values can be combined as required. You can enter an explanatory text for every single value or interval; it is displayed in the input help.

It is only possible to define fixed values for domains of data types CHAR, NUMC, DEC, INT1, INT2 and INT4.
There is only an input check of the template for data types CHAR and NUMC.



Fixed values for a domain

Value Table for a domain

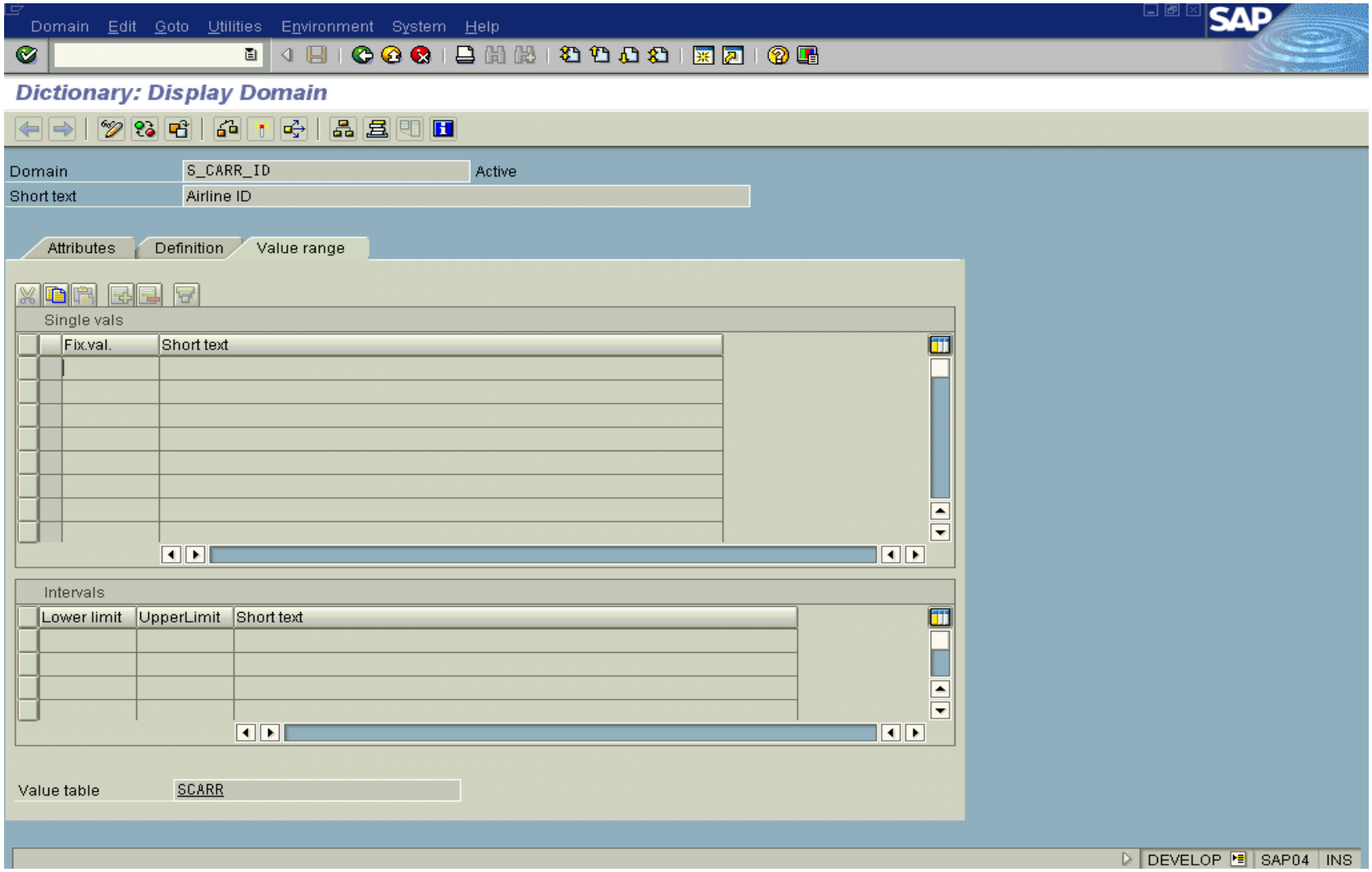
In some cases you can see when you define a domain that all the table fields or structure components referring to this domain should be checked against a certain table. This information can be stored in the domain by entering a value table.

The system proposes the value table as check table when you try to define a foreign key for the field or component. This proposal can be overridden.

Domain S_CARR_ID (data type CHAR, length 3) in the flight model describes the three-place code of the airlines. All the airlines are listed together with their codes in table SCARR.

It is generally advisable to check fields referring to domain S_CARR_ID against table SCARR. SCARR is therefore entered as value table for domain S_CARR_ID. If you want to define a foreign key for a field referring to S_CARR_ID, SCARR is proposed as the check table.

A check is not implemented by simply entering a value table! The check against the value table only takes effect when a foreign key has been defined.



Value table for a domain

Before creating a new domain, check whether a domain that defines the same value range already exists. In this case you should use the existing domain if possible.

Procedure

1> Select object type Domains in the initial screen of the ABAP Dictionary, enter the name of the domain and choose Create.

2> The maintenance screen for domains appears.

Enter an explanatory short text in the field Short text.

You can for example find the domain at a later time using this short text.

3> On the Data type tab page, choose the data type, number of places (valid positions without editing characters such as comma or period) and number of decimal places (only needed for data types DEC, FLTP, QUAN and CURR).

Note that some data types have a fixed length. For example, the data type CLNT (client) always has 3 places. If you enter an invalid number of places for such a data type, the system corrects this automatically after issuing a warning.

4> If only certain input values are valid for the domain, you can enter them in the Value range tab page as fixed values.

You can also define a value table as proposed value for foreign

key checks on this tab page.

5> Save the domain.

You are asked to assign the domain a development class.

6> Choose Activate icon.

Result

The domain is activated. You can find information about the activation flow in the activation log, which you can call with Utilities ® Activation log. If errors occurred when the domain was activated, the activation log is automatically displayed.

Procedure

1> In the field maintenance screen of the table, select the check field and choose If the domain of the check field has a value table, you can have the system create a proposal with the value table as check table. In this case a proposal will be made for the field assignment in the foreign key.

If the domain does not have a value table or if you reject the proposal, the screen for foreign key maintenance appears without proposals. In this case, enter the check table and save your entries. The check table must have a key field to which the domain of the check field is assigned.

2> Enter an explanatory short text in the field Short text.

The short text provides a technical documentation of the meaning of the foreign key.

3> Choose Copy. The foreign key is saved and you return to the maintenance screen for the table.

Inner Join and Outer Join

The data that can be selected with a view depends primarily on whether the view implements an inner join or an outer join.

With an inner join, you only get the records of the cross-product for which there is an entry in all tables used in the view. With an outer join, records are also selected for which there is no entry in some of the tables used in the view.

The set of hits determined by an inner join can therefore be a subset of the hits determined with an outer join.

Table TABA

Field 1	Field 2
A	Text 1
B	Text 2
C	**Text 5**

Table TABB

Field 3	Field 4
A	Text 3
B	Text 4

Join condition

What about the view is displayed?

Field 1	Field 2	Field 4
A	Text 1	Text 3
B	Text 2	Text 4

Inner join

?

Field 1	Field 2	Field 4
A	Text 1	Text 3
B	Text 2	Text 4
C	**Text 5**	

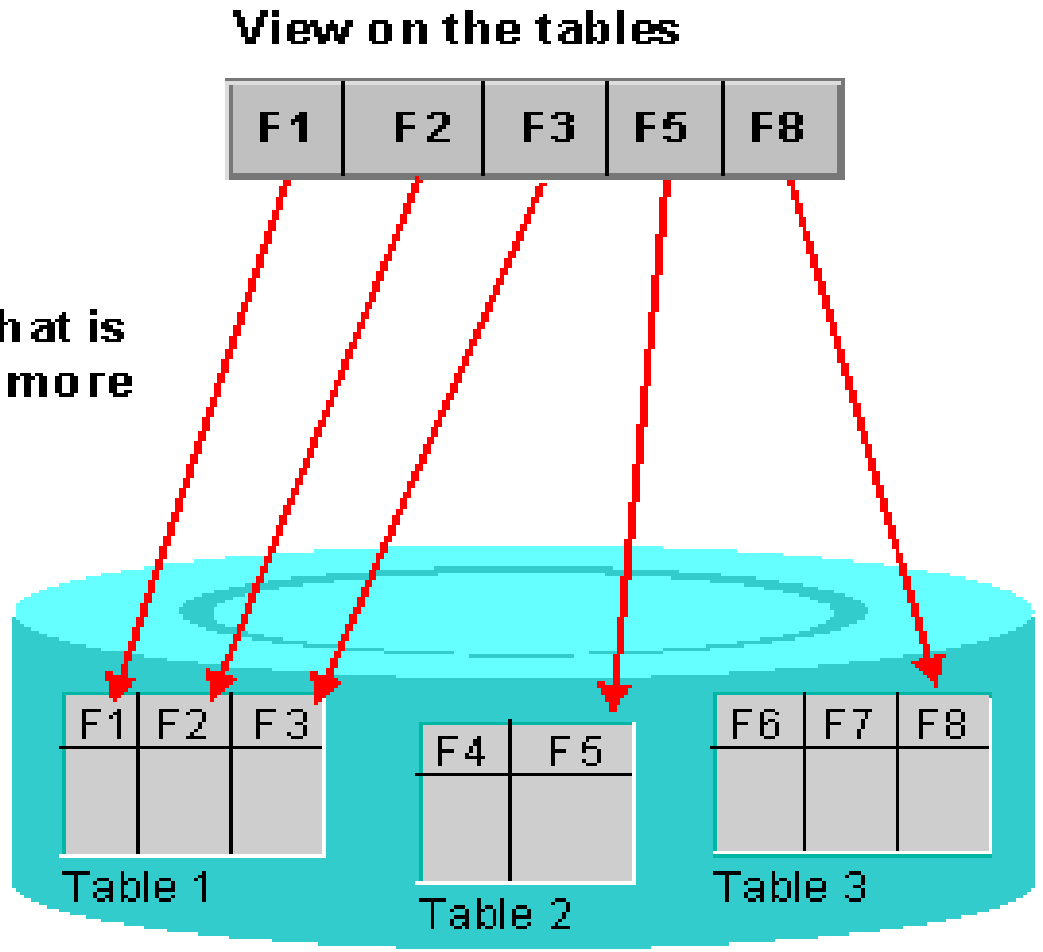
Outer join

Data about an application object is often distributed on several tables.

By defining a view, you can define an application-dependent view that combines this data. The structure of such a view is defined by specifying the tables and fields used in the view.

Fields that are not required can be hidden, thereby minimizing interfaces. A view can be used in ABAP programs for data selection.

View on data that is distributed on more than one table



Four different view types are supported. These differ in the way in which the view is implemented and in the methods permitted for accessing the view data.

Database views are implemented with an equivalent view on the database.

Projection views are used to hide fields of a table (only projection).

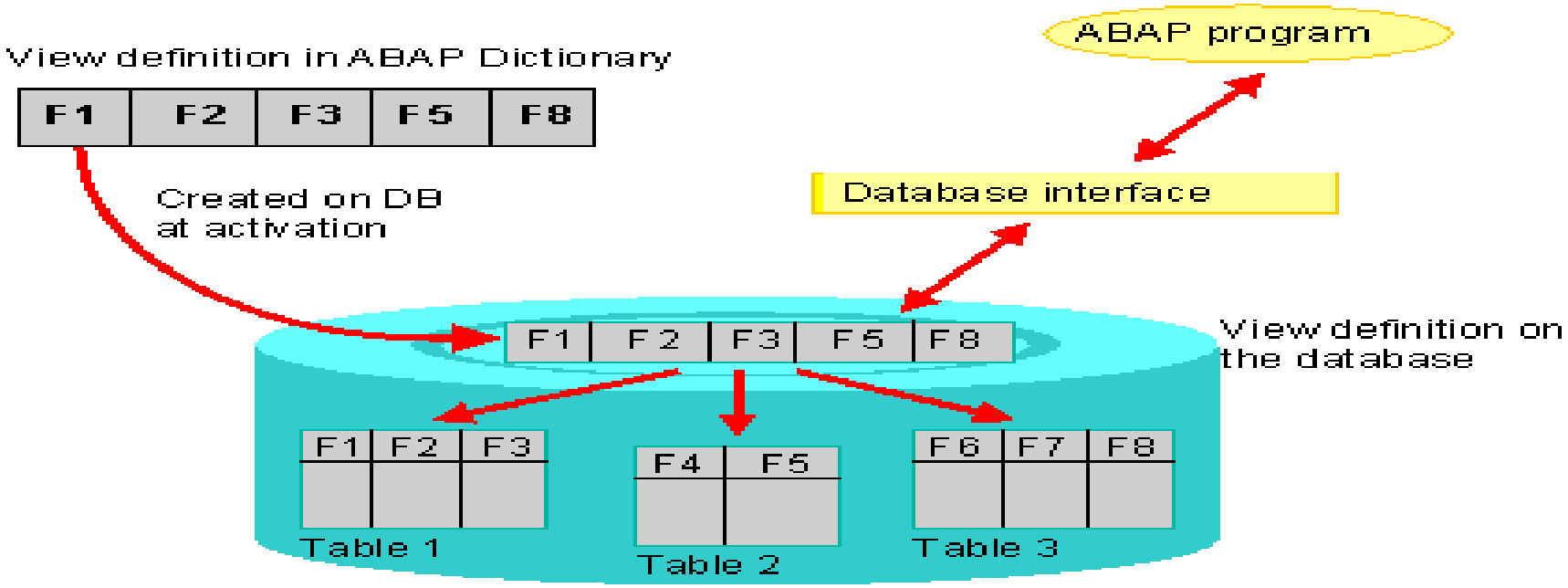
Help views can be used as selection method in search helps.

Maintenance views permit you to maintain the data distributed on several tables for one application object at one time.

Data about an application object is often distributed on several database tables. A database view provides an application-specific view on such distributed data.

Database views are defined in the ABAP Dictionary. A database view is automatically created in the underlying database when it is activated.

Application programs can access the data of a database view using the database interface. You can access the data in ABAP programs with both OPEN SQL and NATIVE SQL. However, the data is actually selected in the database. Since the join operation is executed in the database in this case, you can minimize the number of database accesses in this way. Database views implement an inner join.



If the database view only contains a single table, the maintenance status can be used to determine if data records can also be inserted with the view. If the database view contains more than one table, you can only read the data.

Procedure

1> Enter an explanatory short text in the field Short text.

You can for example find the view at a later time using this short text.

2> Define the tables to be included in the view in the Tables field of the Tables/Join conditions tab page.

Keep in mind that you can only include transparent tables in a database view.

3> Link the tables with join conditions.

Place the cursor on a table name and choose Relationships.

All foreign keys to other tables defined for this table are displayed. Select the foreign keys and choose Copy.

The join condition is now derived from the definitions in the foreign key.

4> On the View fields tab page, select the fields that you want to copy to the view.

Choose Table fields. All the tables contained in the view are displayed in a dialog box. Select a table. All the fields contained in this table are displayed. You can copy fields by selecting them in the first column and choosing Copy.

5> On the Selection conditions tab page, you can (optionally) formulate restrictions for the data records to be displayed with the view .

The selection conditions define the data records that can be selected with the view.

6> With Goto - Technical settings , you can (optionally) maintain the technical settings of the database view.

You can define whether and how the database view should be buffered here. Proceed as for the technical settings of a table

7> On the Maintenance status tab page, select the maintenance status of the database view.

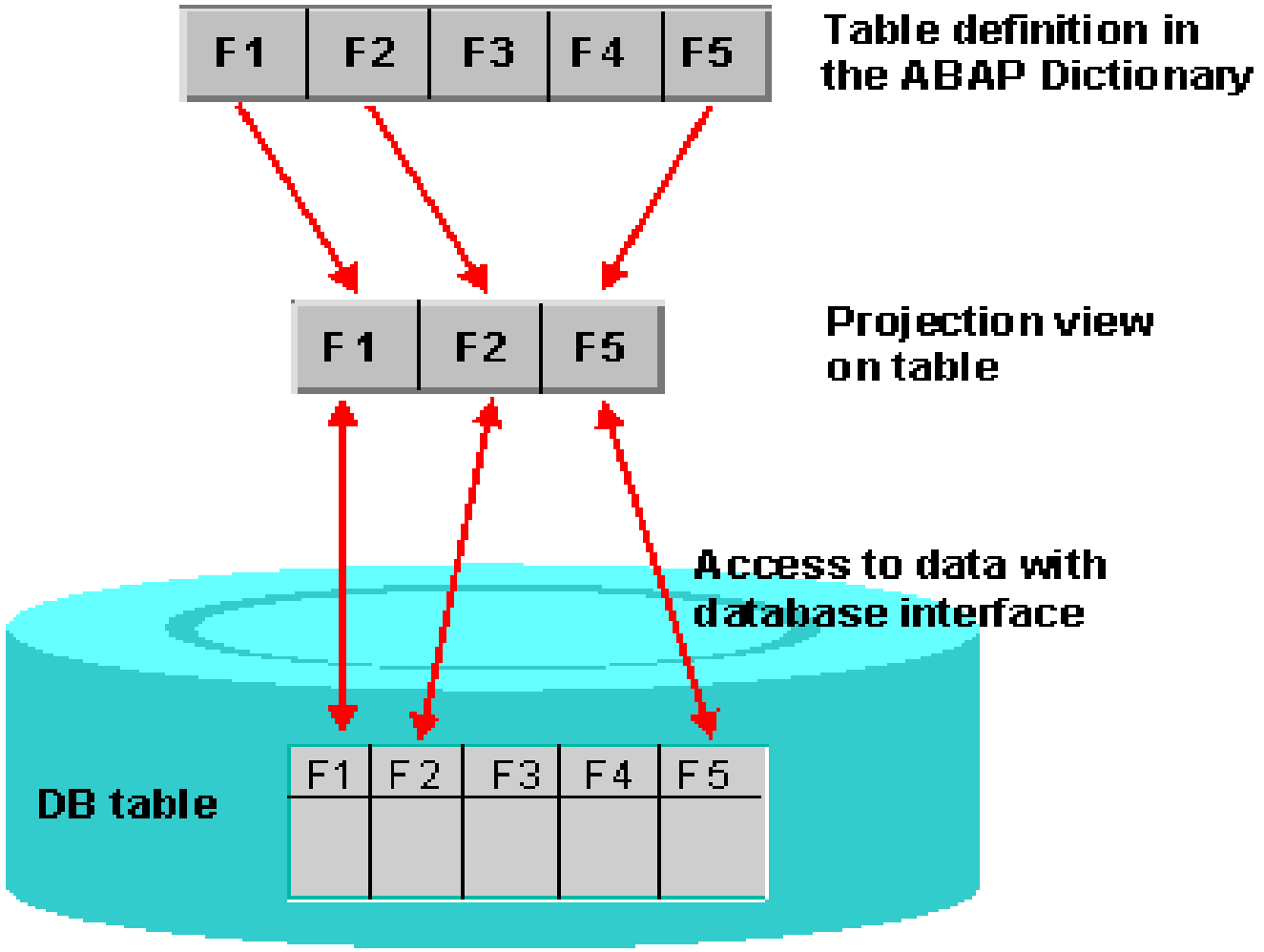
If the view contains more than one table, the maintenance status read only cannot be altered.

8> Save your entries. You are asked to assign the view a development class.

9> Choose Activate .

Projection views are used to hide fields of a table. This can minimize interfaces; for example when you access the database, you only read and write the field contents actually needed.

A projection view contains exactly one table. You cannot define selection conditions for projection views.
There is no corresponding object in the database for a projection view. The R/3 System maps the access to a projection view to the corresponding access to its base table. You can also access pooled tables and cluster tables with a projection view.



Projection View

Procedure

1> Enter an explanatory short text in the field Short text.

You can for example find the view at a later time using this short text.

2> Enter a table name in the field Base table.

A projection view always contains exactly one table.

3> Select the fields of the base table that you want to include in the view. Choose Table fields. The fields of the table are now displayed in a dialog box. You can copy fields by selecting them in the first column and choosing Copy.

4> Save your entries.You are asked to assign the view a development class.

5> Choose Activate.

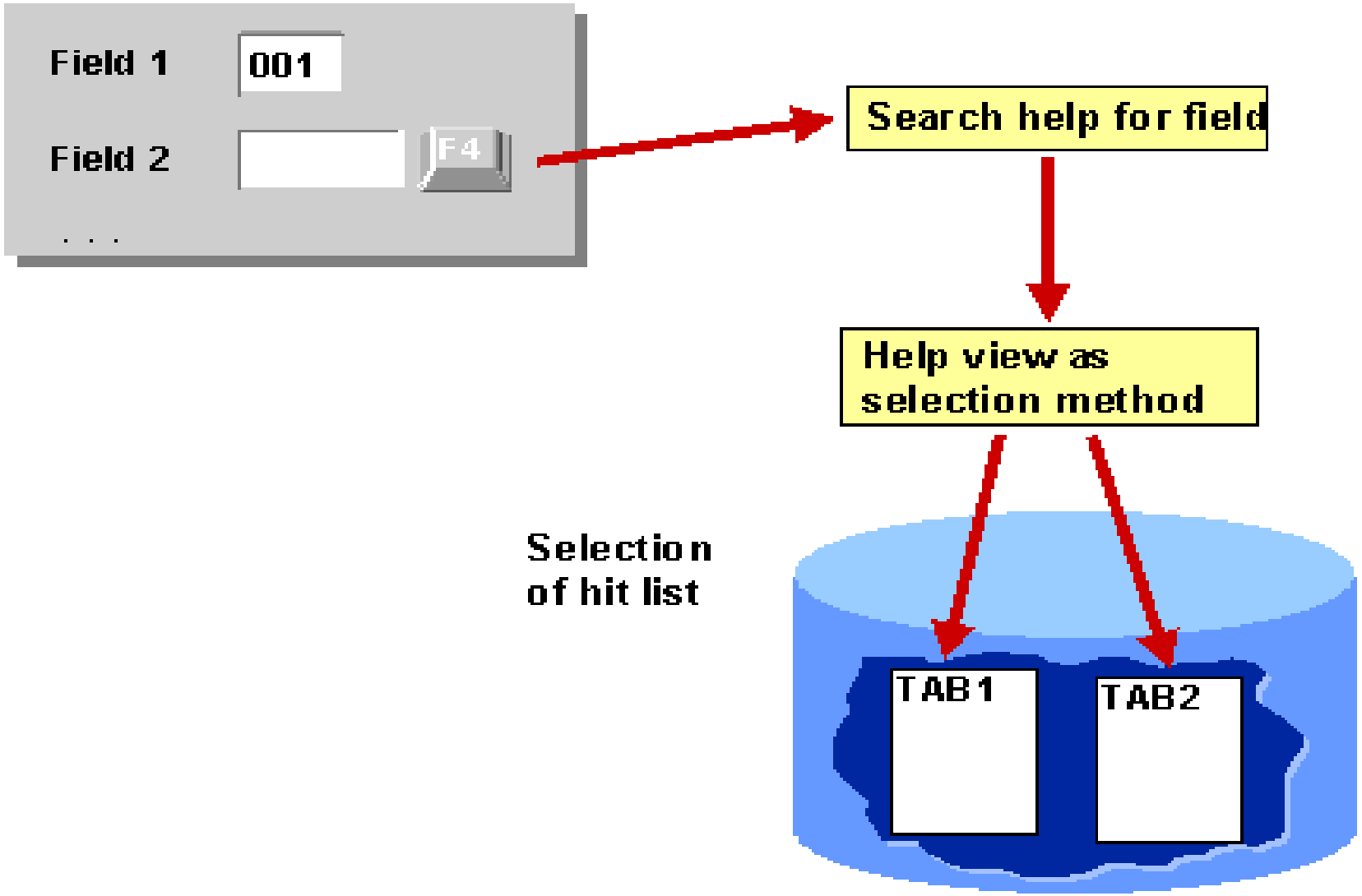
Result

The help view is activated. At activation, a log is written; it can be displayed with Utilities ® Activation log. If errors or warnings occurring when the view was activated, they are displayed directly in the activation log.

You have to create a help view if a view with outer join is needed as selection method of a search help.

The selection method of a search help is either a table or a view. If you have to select data from several tables for the search help, you should generally use a database view as selection method. However, a database view always implements an inner join. If you need a view with outer join for the data selection, you have to use a help view as selection method.

The creation method for Help view is similar to that of Database view.


Field 1
001
Field 2
F4
. . .
Search help for field
Help view as selection method
Selection of hit list
TAB1
TAB2

The input help (F4 help) is a standard function of the R/3 System. The user can display the list of all possible input values for a screen field with the input help.

This standard process can be completely defined by creating a search help in the ABAP Dictionary. This search help only has to be assigned to the screen fields in which they should be available.
There are two types of search help:
Elementary search helps describe a search path. The elementary search help must define where the data of the hit list should be read from (selection method), how the exchange of values between the screen template and selection method is implemented

(interface of the search help) and how the online input help should be defined (online behavior of the search help).

<u>Collective search helps</u> Combine several elementary search helps. A collective search help thus can offer several alternative search paths.

Each customer of a carrier or of a travel agency has a customer number. You want to find a search option for this customer number. The user must be offered two different search paths.

1> The user should be able to search for the customer number using the customer data, such as the name and address.

2>The user should be able to search for the customer number using existing customer bookings.

Two elementary search helps SCUSTOM_NAME (for searching with the customer data) and SCUSTOM_BOOK (for searching with the existing bookings) are created for the actual search paths.

These elementary search helps are included in the collective search help SCUSTOM.

Elementary Search Help SCUSTOM_NAME

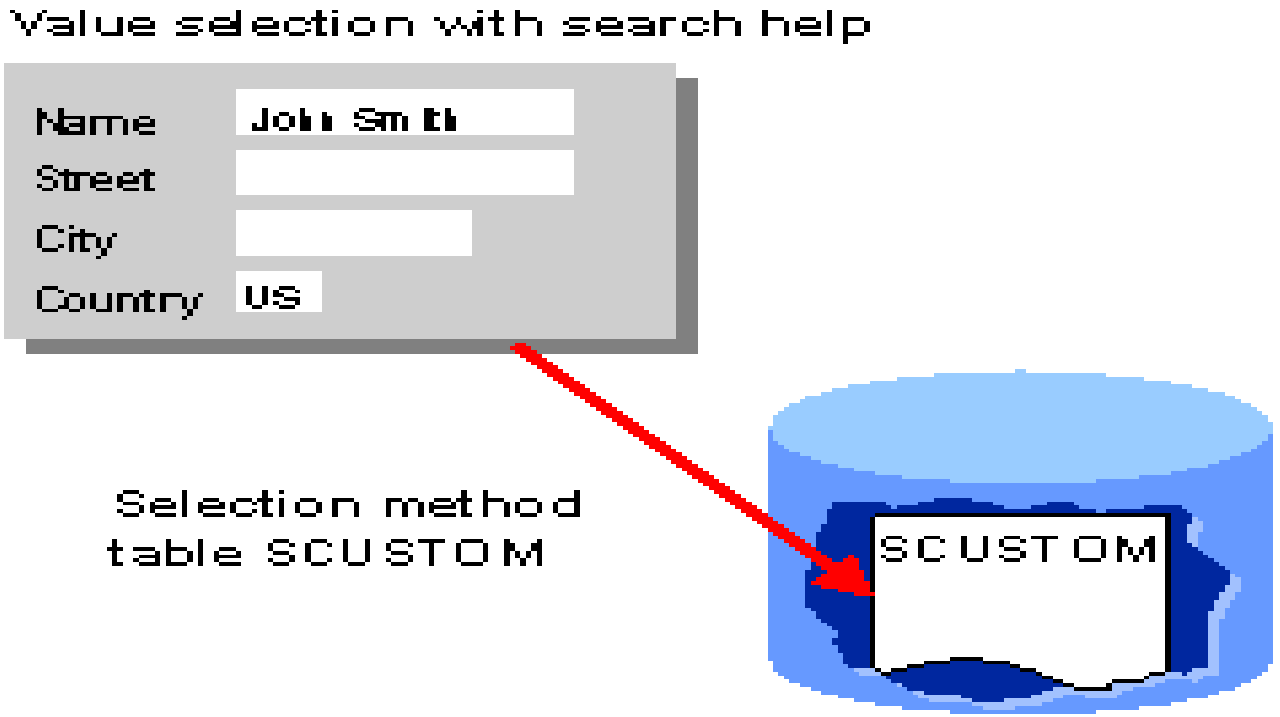
This elementary search help should enable you to search for the customer number using the name and address (street, city, country). All this data is contained in table SCUSTOM. Table SCUSTOM must therefore be selected as the selection method of the elementary search help.

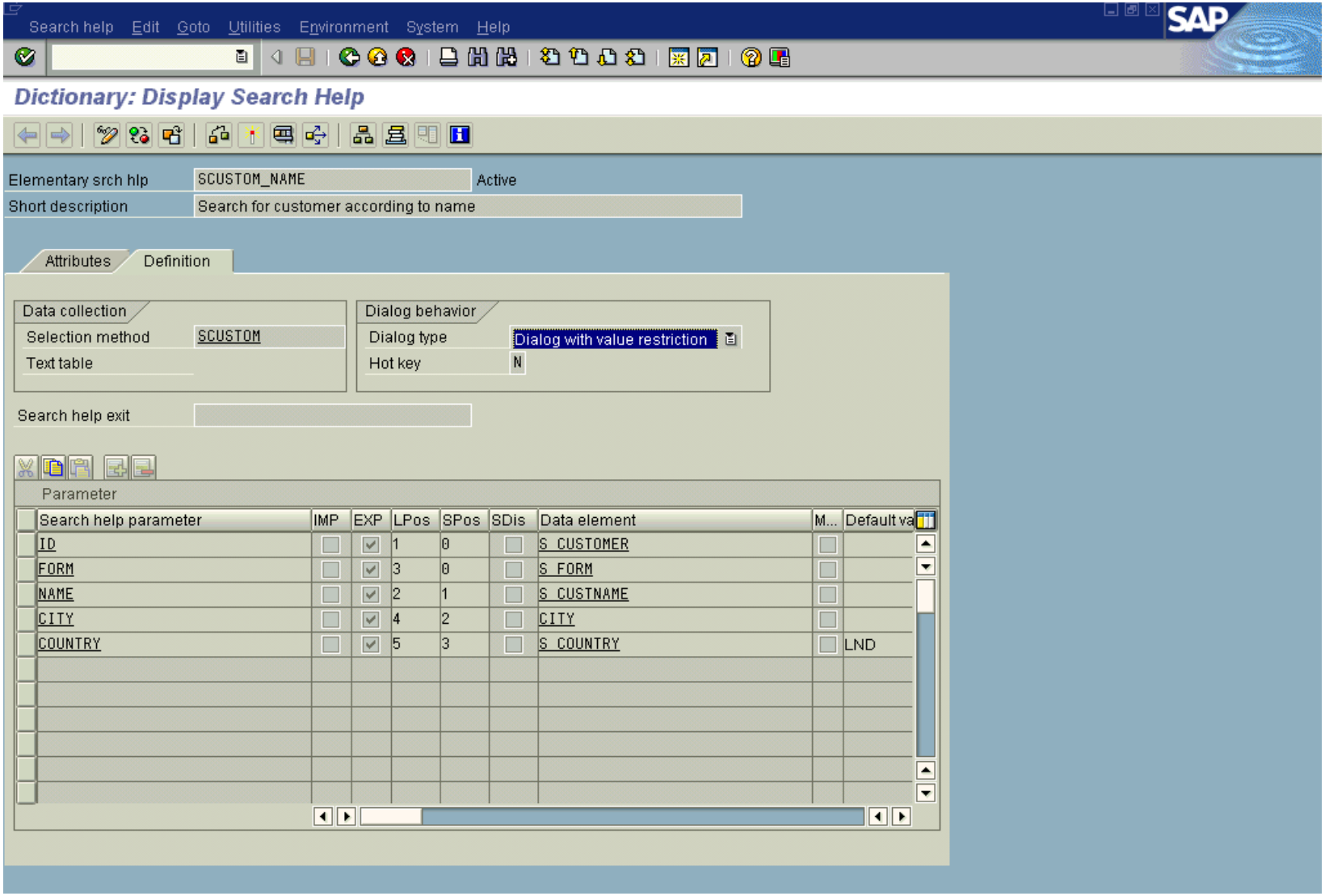
You now have to decide which fields of the selection method are needed for the input help process. These are the fields that should appear either in the dialog box for restricting values or in the hit list.

In the dialog box for restricting values, the user should be able to restrict values with the customer's name and address, i.e. the fields for the street, city and country. These fields as well as the customer's number (the information to be found must always be in the hit list) should appear in the hit list. The fields ID, NAME, STREET, CITY and COUNTRY of table SCUSTOM must be included in the search help as parameters.

The parameter ID is declared to be an import parameter. A pattern entered in the corresponding field of a screen template can therefore be used directly for the value selection. Restrictions for the other parameters of the search help must be entered in the dialog box for

value selection.All the parameters of the search help are declared to be export parameters. As a result, all the parameters of the hit list can be returned to the screen template if the corresponding fields are available there.





Elementary Search help

Procedure

1> In the initial screen of the ABAP Dictionary, select object class Search help, enter the name of the search help and choose Create. A dialog box appears in which you must select the type of search help.

2> Select Elementary search help and choose .

The maintenance screen for elementary search helps appears.

3> Enter an explanatory text in the field Short text.

You can for example find the search help at a later time using this short text.

4> In the Definition tab page enter the selection method of the search help. You can enter the name of a table or a view (database view , projection view or help view) here. If you enter a table that has a text table, the name of the text table is automatically entered in the corresponding field.

5> Using the input help (F4 help), select fields of the selection method as parameter in the Search help parameter area. Select the fields that should be used in the dialog box for value selection or in the hit list.

If the selection method is a table that has a text table, both the fields of the table and the fields of the text table are offered in the

input help.

The data element of the parameter is automatically copied from the selection method. The data element defines the output attributes and the F1 help of the parameter in the hit list and in the dialog box for value selection.

You can assign the parameter another data element. To do so, select the Mod flag. The Data element field is now ready for input. Then select a data element with the input help (F4 help). Only data elements whose data type, length and number of decimal places is the same as those of the previous data element can be assigned. This removes the link between the data element of the search help parameter and the data element of the selection method field having

the same name. If you cancel the Mod flag, the data element of the assigned table field is used again.

6> Define the attributes of the search help parameters.

Select the IMP flag if it is an import parameter. Select the EXP flag if it is an export parameter.

You can define the dialog for the input help with the fields LPos, SPos and SDis. Enter the parameter position in the hit list in LPos. If you enter nothing or the value 0 here, the parameter is not displayed in the hit list.

Enter the parameter position in the dialog box for value selection in SPos. If you enter nothing or the value 0 here, the parameter is

not displayed in the dialog box for value selection.

Set the SDis flag if the parameter should be a pure display field in the dialog box for value selection. The user is thus informed that the contents of the parameter restrict the value, but he cannot change this restriction. This makes sense for example when the parameter is an import parameter or if it has a default value.

You can assign the parameter a default value in the Default value field.

7> Select the dialog type of the search help.

The dialog type defines how the hit list is displayed in the input help.

8> Save your entries.

A dialog box appears in which you have to assign the search help a development class.

9> Choose Activate .

Dialog types for a Search help

The dialog type of an elementary search help defines how the hit list is displayed when the input help is called.

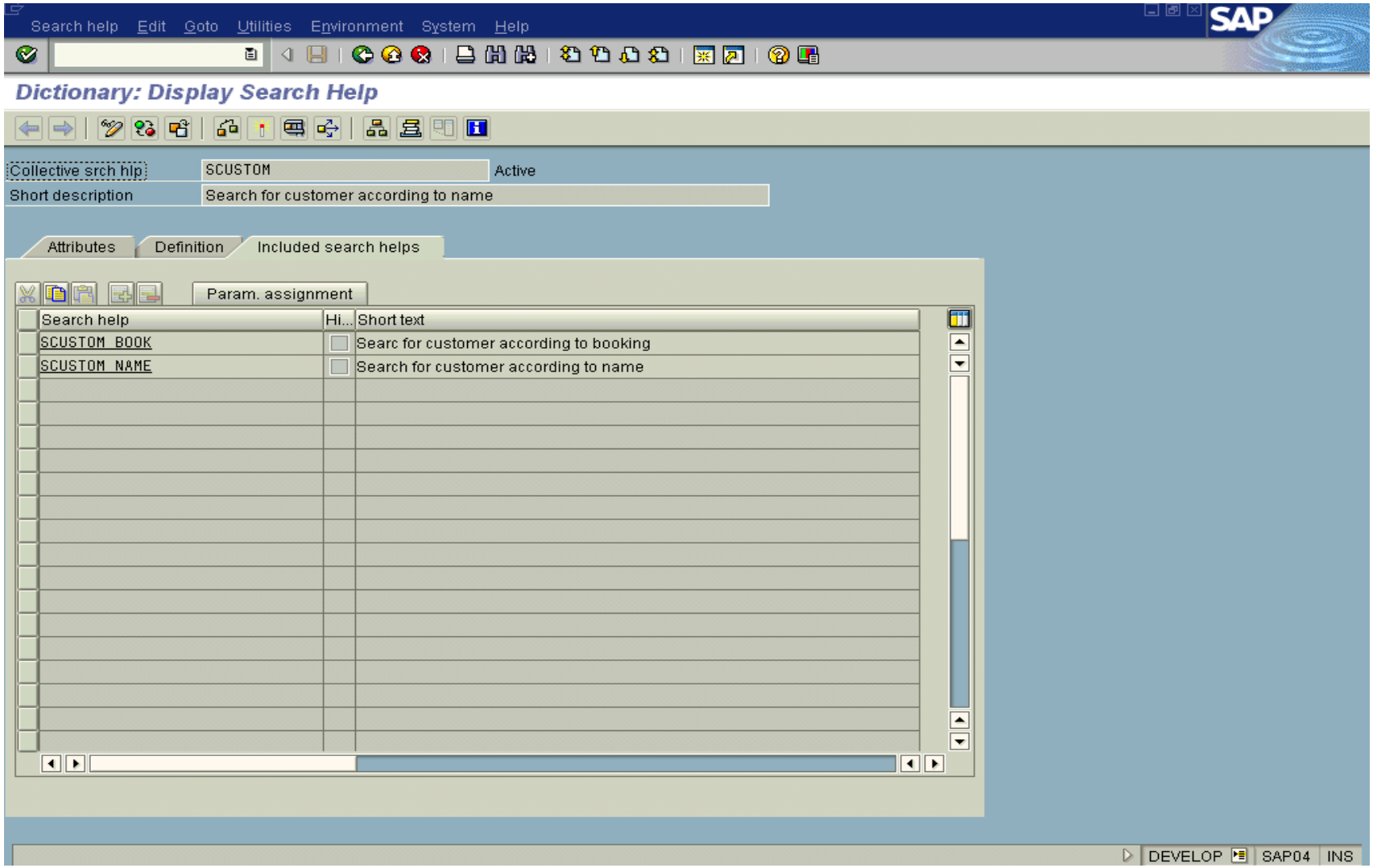
The following dialog types are possible:

1> Immediate value display: The hit list is immediately displayed when the input help is called. This is only meaningful if the hit list usually only contains a few entries.

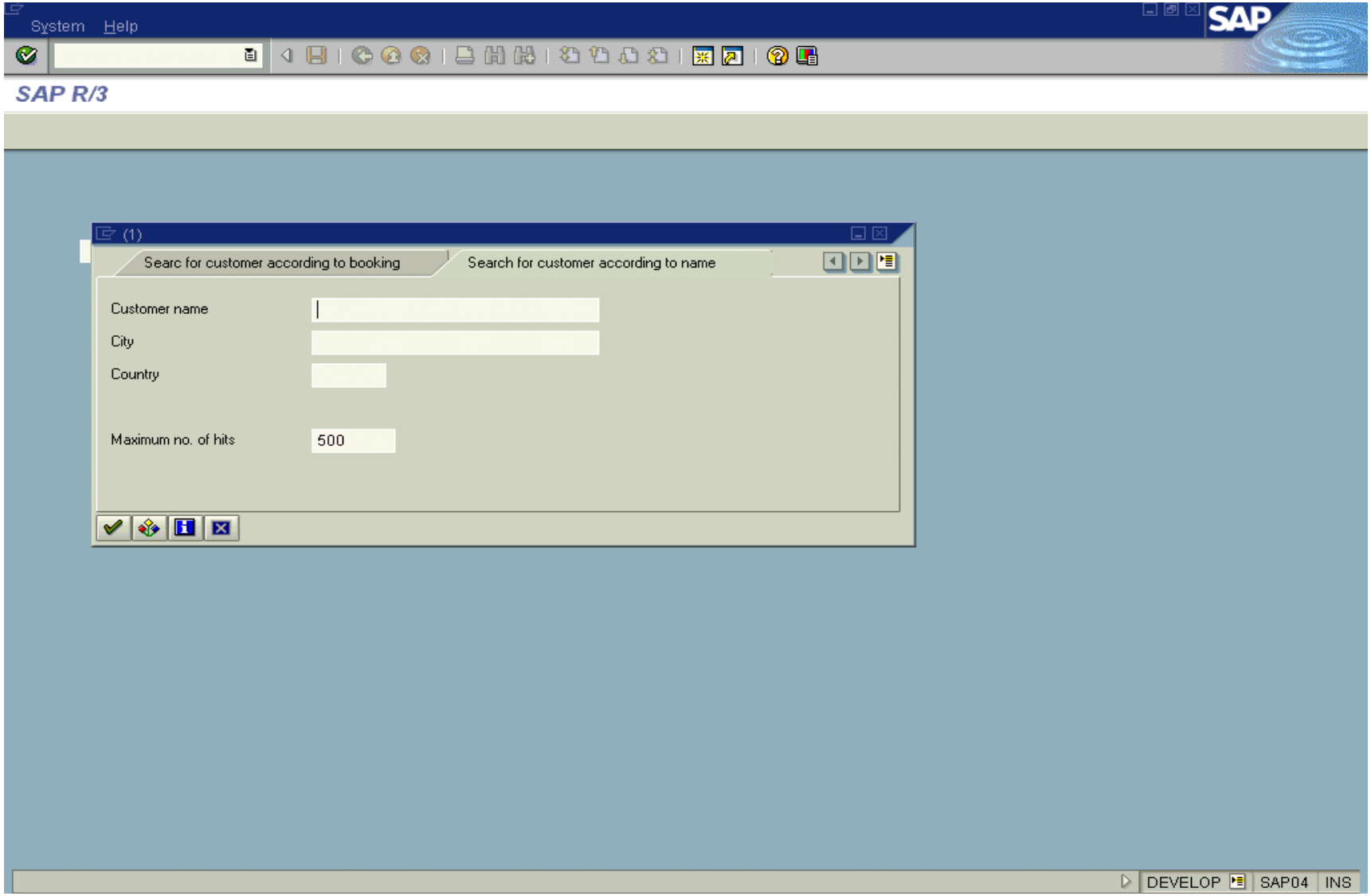
2> Complex dialog with value restriction: The dialog window for restricting values is offered immediately. Choose this option if the list of possible entries is usually very large. If the user limits the amount of data to be processed, the hit list will become more

comprehensible and the system load during value selection will be reduced.

3> Dialog depending on number of values: If the hit list contains less than 100 entries, it is displayed immediately. If the hit list contains more than 100 entries, the dialog box for restricting values is displayed.



Collective Search Help SCUSTOM



F4 for Collective Search Help

Procedure

1> In the initial screen of the ABAP Dictionary, select object class Search help, enter the name of the search help and choose Create. A dialog box appears in which you must select the type of search help.

2> Select Collective search help and choose .

The maintenance screen for collective search helps is displayed.

3> Enter an explanatory text in the field Short text.You can for example find the search help at a later time using this short text.

4> In the Definition tab page enter the parameters of the collective search help. Select the Imp flag if it is an import parameter. Select the Exp flag if it is an export parameter.
Define the types for the parameters of a collective search help by assigning a data element. Enter the name of the data element that describes the contents of the search help parameter in the Data element field.
You can assign the parameter a default value in the Default value field.

5> In exceptions it could be necessary to change the standard process defined by the search help. You can implement the deviation from the standard using a search help exit.
In this case enter the name of the search help exit in the corresponding field.

6> On the Included search helps tab page, define the search helps that you want to include in the collective search help.You can include elementary search helps and collective search helps. Use the Hide flag to control whether an included search help should appear in the dialog box for selecting the elementary search help. If the flag is set, the search help is not offered.

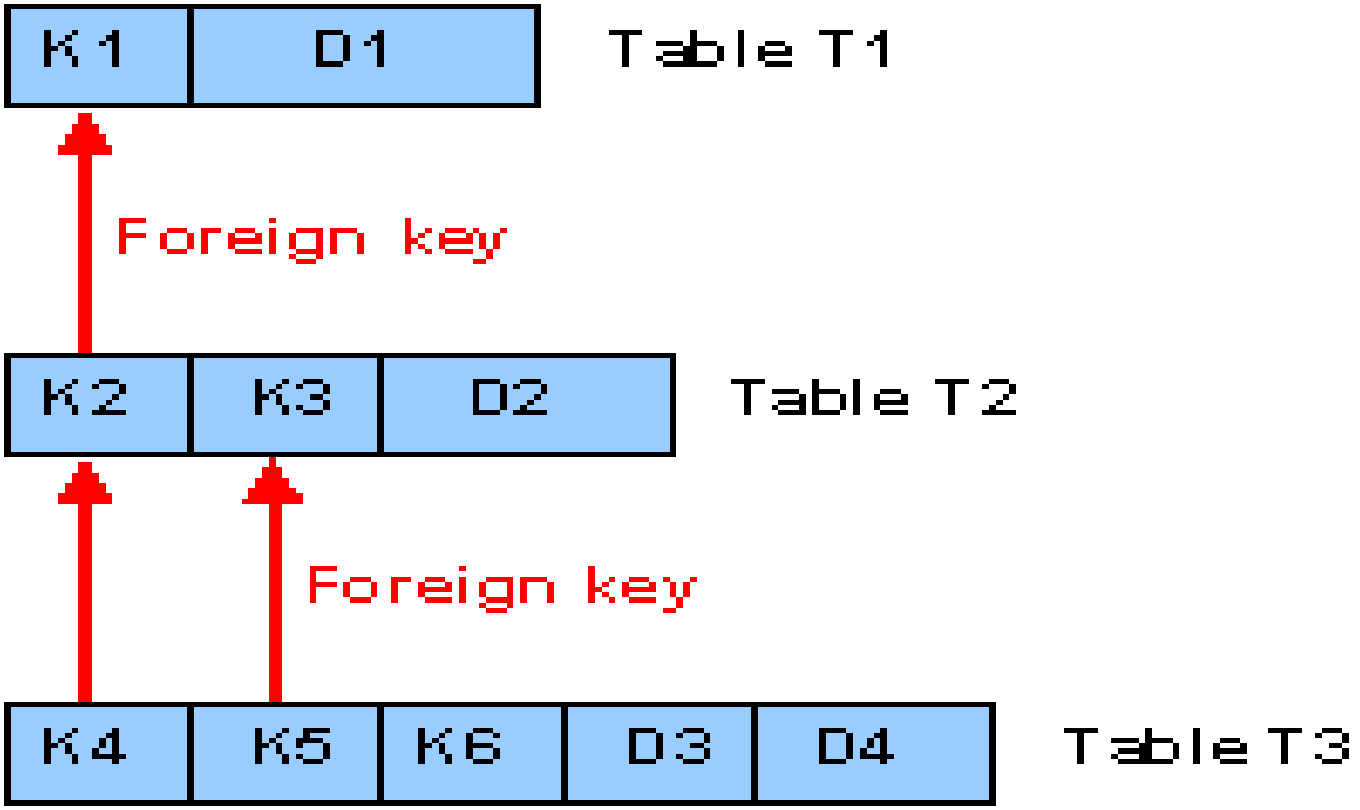
7> Save your entries.
A dialog box appears in which you have to assign a development class to the search help.

8> Choose Activate.

The R/3 System synchronizes simultaneous access of several users to the same data records with a lock mechanism. When interactive transactions are programmed, locks are set and released by calling function modules . These function modules are automatically generated from the definition of lock objects in the ABAP Dictionary.

Structure of a Lock Object

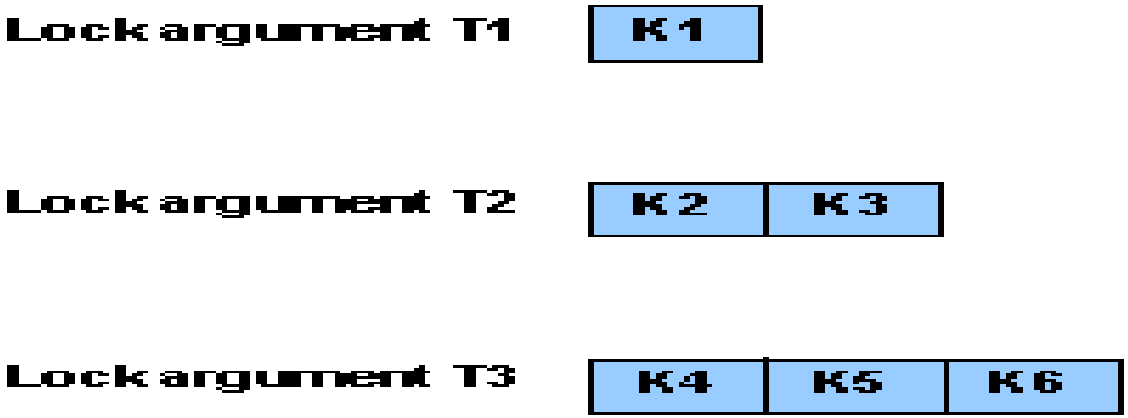
The tables in which data records should be locked with a lock request are defined in a lock object together with their key fields. When tables are selected, one table (the primary table) is first selected. Further tables (secondary tables) can also be added using foreign key relationships.


K1
D1
Table T1
Foreign key
K2
K3
D2
Table T2
Foreign key
K4
K5
K6
D3
D4
Table T3

Lock Arguments

The lock argument of a table in the lock object consists of the key fields of the table.

The lock argument fields of a lock object are used as input parameters in the function modules for setting and removing locks generated from the lock object definition. When these function modules are called, the table rows to be locked or unlocked are specified by defining certain values in these fields. These values can also be generic. The lock argument fields therefore define which subset of the table rows should be locked.



The simplest case of a lock object consists of exactly one table and the lock argument of the table is the primary key of this table. Several tables can also be included in a lock object. A lock request therefore can lock an entire logical object, and not only a record of a table. Such a logical object can be for example a document comprising an entry in a header table and N entries in a position table.

Call the lock function module with K1=2
and K3=1 (K6 unspecified)

Table T1

K1	D1
1	...
2	...
3	...
...	

Table T2

K2	K3	D2
1	1	...
2	1	...
2	2	...
3	1	...
3	3	...
...		

Table T3

K4	K5	K6	D3	D4
1	1	A	...	...
2	1	A	...	...
2	1	B	...	...
2	1	C	...	...
2	3	A	...	...
2	3	B	...	...
3	1	A	...	...
...				

Red entries are locked

Lock Mode

The lock mode controls whether several users can access data records at the same time. The lock mode can be assigned separately for each table in the lock object. When the lock is set, the corresponding lock entry is stored in the lock table of the system for each table.

Access by more than one user can be synchronized in the following ways:

Exclusive lock: The locked data can only be displayed or edited by a single user. A request for another exclusive lock or for a shared lock is rejected.

Shared lock: More than one user can access the locked data at the same time in display mode. A request for another shared lock is

accepted, even if it comes from another user. An exclusive lock is rejected.

Exclusive but not cumulative: Exclusive locks can be requested several times from the same transaction and are processed successively. In contrast, exclusive but not cumulative locks can be called only once from the same transaction. All other lock requests are rejected.

1> Select object type Lock object in the initial screen of the ABAP Dictionary, enter an object name and choose Create.

The name of a lock object should begin with an E (Enqueue).

The maintenance screen for lock objects is displayed.

2> Enter an explanatory short text in the field Short text.

You can then use the short text to find the lock object at a later time, for example with the R/3 Repository Information System.

3> Enter the name of the primary table of the lock object.

All other tables in the lock object must be linked with the primary table using foreign keys.

4> Select the lock mode of the primary table in the field below it.

The lock mode is used as the default value for the corresponding parameters of the function modules generated from the lock object.

5> Choose Add if you want to lock records in more than one table with the lock object.

A list of all the tables linked with the primary table using valid foreign keys is displayed. Select the appropriate table. The lock mode of the primary table is copied as lock mode. You can change this setting as required, for example you can assign the lock mode separately for each table.

Similarly, you can add a table linked with the secondary table just added with foreign keys. To do this, place the cursor on the name of the secondary table and choose Add.

If no lock mode is assigned to a table, no lock is set for the entries in this table when the generated function modules are called. You should not assign a lock mode if a secondary table was only used to define a path between the primary table and another secondary table with foreign keys.

6> Save your entries.

A dialog box appears in which you have to assign the lock object a development class.

7> You can define whether the function modules generated from the lock object should be RFC-enabled on the Attributes tab page.

If you set the Allow RFC flag, the generated function modules can be called from within another system with Remote Function Call.

If you permit Remote Function Calls for an existing lock object, you must ensure that the generated function modules are called from within an ABAP program with parameters appropriate for the type. You should therefore check all programs that use the associated function modules before activating the lock object with the new option.

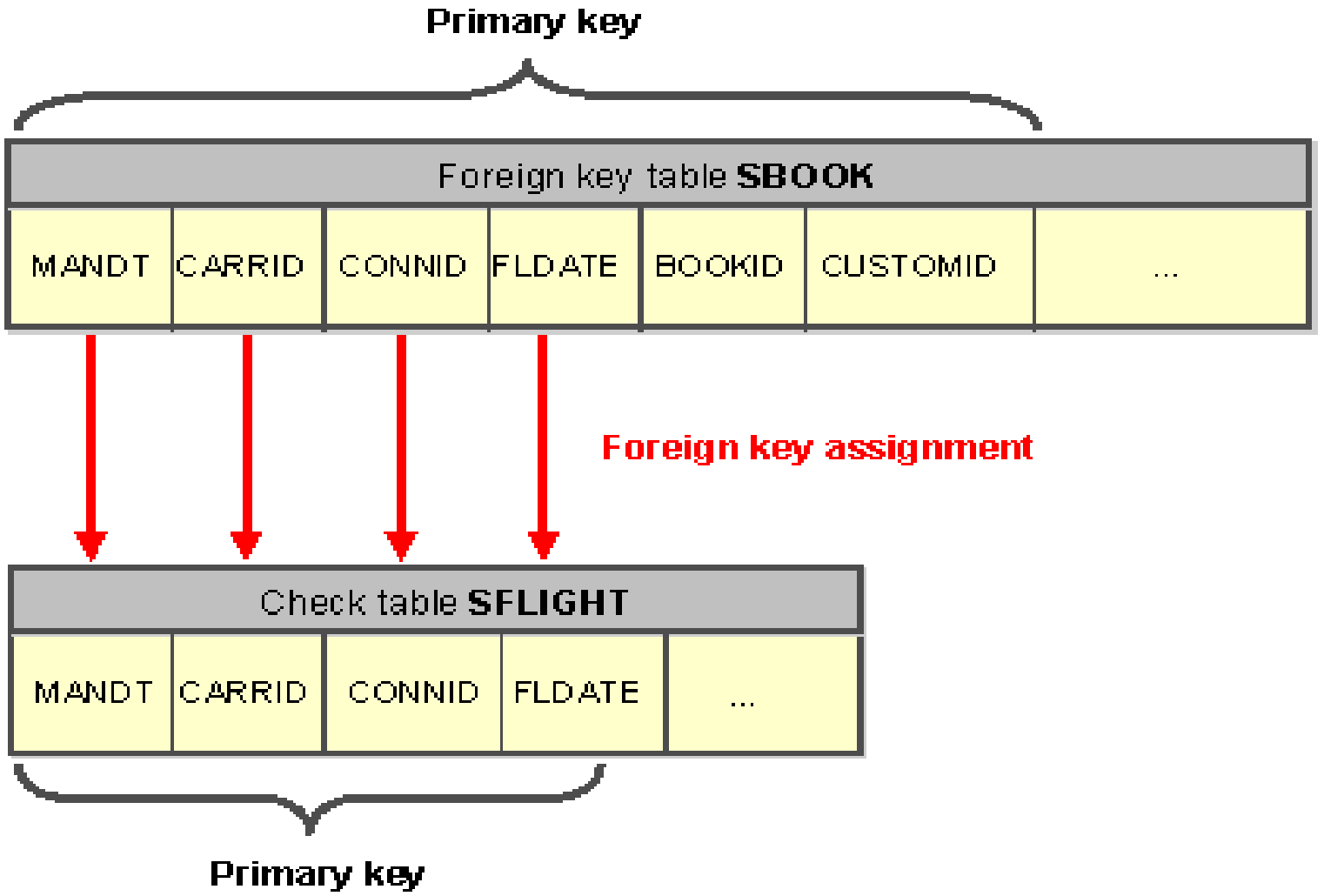
8> Choose Activate.

Result

When you activate the lock object, the two function modules ENQUEUE_<lockobjectname> and DEQUEUE_<lockobjectname> are generated from its definition to set and release locks.

When booking flights it is important to prevent flights from being overbooked. For this reason, you have to lock the particular flight as well as all the bookings existing for this flight during processing. You can do this with lock object E_BOOKING.

The flights are recorded in table SFLIGHT and the bookings for the flights in table SBOOK. The two tables are linked with a foreign key. Lock object E_BOOKING must therefore contain table SFLIGHT as primary table and table SBOOK as further table.


Primary key
Foreign key table SBOOK
MANDT
CARRID
CONNID
FLDATE
BOOKID
CUSTOMID
...
Foreign key assignment
Check table SFLIGHT
MANDT
CARRID
CONNID
FLDATE
...
Primary key

The lock argument of table SFLIGHT thus contains the fields MANDT, CARRID, CONNID, and FLDATE. The lock argument of table SBOOK thus contains the fields MANDT, CARRID, CONNID, FLDATE, BOOKID and CUSTOMID.

Select exclusive lock mode, that is the locked data can only be displayed and edited by one user.

When the lock object is activated, the following function modules are generated from its definition:

ENQUEUE_ E_BOOKING (set locks)

DEQUEUE_ E_BOOKING (release locks)

```
CALL FUNCTION 'ENQUEUE_E_BOOKING'
    exporting
         mode_sflight    = 'E'
         mode_sbook      = 'E'
         mandt           = sy-mandt
         carrid          = 'LH '
         connid          = 400
         fldate          = '19981129'
         bookid          = 0
         customid        = 0
         x_carrid        = ' '
         x_connid        = ' '
         x_fldate        = ' '
         x_bookid        = ' '
         x_customid      = ' '
         _scope          = '2'
         _wait           = 'X'
         _collect        = ' '
     exceptions
          foreign_lock    = 1
          system_failure  = 2
          others          = 3.
```

Lock modes

Lock parameters

Lock behavior when copying initial value

Pass lock to update program

Behavior in conflict situations

Lock container

Calling Enqueue Function module

With this call, flight LH 400 on Nov. 29,1998 is exclusively (lock mode E) locked in table SFLIGHT together with all the bookings entered in table SBOOK for this flight (since the initial value 0 is transferred for BOOKID and CUSTOMID).

The lock is sent to the update program (_SCOPE = '2'). If there is a lock conflict, another attempt is made to set the lock after a certain time (_WAIT = 'X').

```
CALL FUNCTION 'DEQUEUE_E_BOOKING'
     exporting
          mode_sflight = 'E'
          mode_sbook   = 'E'
          mandt        = sy-mandt
          carrid       = 'LH'
          connid       = 400
          fldate       = '19981129'
          bookid       = 0
          customid     = 0
          x_carrid     = ' '
          x_connid     = ' '
          x_fldate     = ' '
          x_bookid     = ' '
          x_customid   = ' '
          _scope       = '3'
          _synchron    = ' '
          _collect     = ' '.
```

Lock modes

Lock parameters

Lock behavior when copying initial value

Pass lock to update program

Synchronous deletion of lock entry

Lock container

Calling Dequeue Function module

The existing exclusive lock entries for flight LH 400 are deleted in table SFLIGHT and the bookings for this flight are deleted in table SBOOK. The request to delete the lock entries is passed on to the update program (_SCOPE = ‘3’).

Field Names of the Lock Object

The keys to be locked must be passed here.

A further parameter X_<field> that defines the lock behavior when the initial value is passed exists for every lock field <field>. If the initial value is assigned to <field> and X_<field>, then a generic lock is initialized with respect to <field>. If <field> is assigned the initial value and X_<field> is defined as X, the lock is set with exactly the initial value of <field>.

Parameters for Passing Locks to the Update Program

A lock is generally removed at the end of the transaction or when the corresponding DEQUEUE function module is called.

Parameter _SCOPE controls how the lock or lock release is passed to the update program. You have the following options:

_SCOPE = 1: Locks and lock releases are not passed to the update program. The lock is removed when the transaction is ended.

_SCOPE = 2: The lock or lock release is passed to the update program. The update program is responsible for removing the lock. The interactive program with which the lock was requested no longer has an influence on the lock behavior.

_SCOPE = 3: The lock or lock release is also passed to the update program. The lock must be removed in both the interactive program and in the update program. This is the standard setting for the DEQUEUE function module.

Controlling Lock Transmission

Parameter _COLLECT controls whether the lock request or lock release should be performed directly or whether it should first be written to the local lock container. This parameter can have the following values:

Initial value: The lock request or lock release is sent directly to the lock server.

X : The lock request or lock release is placed in the local lock container. The lock requests and lock releases collected in this lock container can then be sent to the lock server at a later time as a group by calling the function module FLUSH_ENQUEUE.

Behavior for Lock Conflicts (ENQUEUE only)

The ENQUEUE function module also has the parameter _WAIT.

This parameter determines the lock behavior when there is a lock conflict.

You have the following options:

Initial value: If a lock attempt fails because there is a competing lock, the exception FOREIGN_LOCK is triggered.

X : If a lock attempt fails because there is a competing lock, the lock attempt is repeated after waiting for a certain time. The exception FOREIGN_LOCK is triggered only if a certain time limit has elapsed since the first lock attempt. The waiting time and the time limit are defined by profile parameters.

Controlling Deletion of the Lock Entry (DEQUEUE only)

The DEQUEUE function module also has the parameter _SYNCHRON.

If X is passed, the DEQUEUE function waits until the entry has been removed from the lock table. Otherwise it is deleted asynchronously, that is, if the lock table of the system is read directly after the lock is removed, the entry in the lock table may still exist.

Exceptions of the ENQUEUE Function Module

FOREIGN_LOCK: A competing lock already exists. You can find out the name of the user holding the lock by looking at system variable SY-MSGV1.

SYSTEM_FAILURE: This exception is triggered when the lock server reports that a problem occurred while setting the lock. In this case, the lock could not be set.

SUMMARY

- **The ABAP Dictionary centrally describes and manages all the data definitions used in the system. The ABAP Dictionary is completely integrated in the ABAP Development Workbench.**
- **Tables are defined in the ABAP Dictionary independently of the database.**
- **Views are logical views on more than one table.**
- **A domain defines the value range of all table fields and structure components that refer to this domain.**
- **Lock Objects are used to synchronize access to the same data by more than one user.**

EXERCISE

Using SE11 create an user defined table with the following fields,

- **Employee No. (Primary Key)**
- **Name**
- **Designation**
- **Salary**

Try to add few records into that table.

Note: Follow the steps explained in the presentation for the creation of the table.