

Lecture 3 Workbook

Data Dictionary objects Part 2

Contents

DIFFERENT TYPES OF VIEWS USED IN ABAP.....	2
HOW TO CREATE A MAINTENANCE VIEW	3
MAINTENANCE ATTRIBUTES.....	4
HOW TO CREATE A MAINTENANCE DIALOG	5
HOW TO CREATE A VIEW CLUSTER	6
HOW TO CREATE A RANGE TABLE	11
HOW TO CREATE TRANSACTION WITH PARAMETERS (FOR SM30)	13
HOW TO CREATE TRANSACTION FOR A VIEW CLUSTER (FOR SM34).....	15
HOW TO CREATE A SEARCH HELP	16
HOW TO CREATE A LOCK OBJECT	18

Different types of views used in ABAP

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.

The followings are different types of views:

- **Database View (transaction SE11)**

Database views implement an inner join, that is, only records of the primary table (selected via the join operation) for which the corresponding records of the secondary tables also exist are fetched.

Inconsistencies between primary and secondary table could, therefore, lead to a reduced selection set.

In database views, the join conditions can be formulated using equality relationships between any base fields. In the other types of view, they must be taken from existing foreign keys. That is, tables can only be collected in a maintenance or help view if they are linked to one another via foreign keys.

- **Help View (transaction SE54)**

Help views are used to output additional information when the online help system is called.

When the F4 button is pressed for a screen field, a check is first made on whether a matchcode is defined for this field. If this is not the case, the help view is displayed in which the check table of the field is the primary table. Thus, for each table no more than one help view can be created, that is, a table can only be primary table in at most one help view.

- **Projection View**

Projection views are used to suppress or mask certain fields in a table (projection), thus minimizing the number of interfaces. This means that only the data that is actually required is exchanged when the database is accessed.

A projection view can draw upon only one table. Selection conditions cannot be specified for projection views.

- **Maintenance View (transaction SE54)**

Maintenance views enable a business-oriented approach to looking at data, while at the same time, making it possible to maintain the data involved. Data from several tables can be summarized in a maintenance view and maintained collectively via this view. That is, the data is entered via the view and then distributed to the underlying tables by the system.

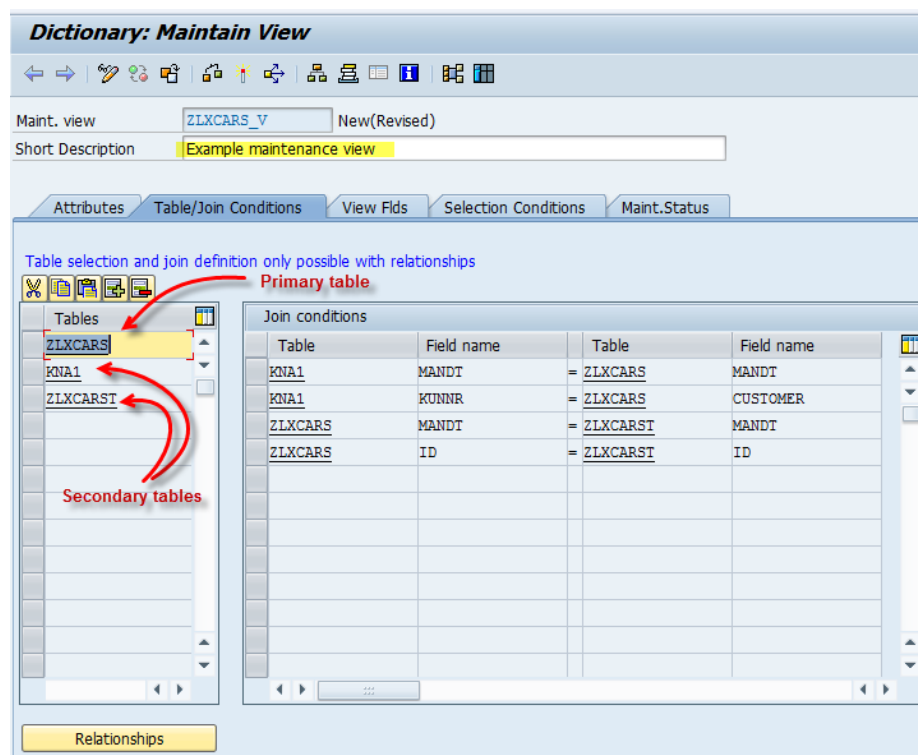
How to create a maintenance view

In a maintenance view user can watch data of multiple tables and also able maintain data across tables. Data is entered via the maintenance view and then distributed to the included tables of that view by the system.

For creation of a maintenance view you should use transaction SE11. Choose the View radio button and enter the name of the view. After that you may choose the type of the view.


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.

Enter the primary table of the view under Tables in the Tables/Join Conditions tab page.



Only those tables that are linked with the primary table (indirectly) with a foreign key can be included in the maintenance view.


If required, include more tables in the view. In a maintenance view you can only insert tables that are linked to one another with foreign keys.

Place the cursor on the primary table and choose Relationships. All existing foreign key relationships of the primary table are displayed. Select the required foreign key and choose  Copy. The secondary table used in such a foreign key is included in the view. The join conditions derived from the foreign keys are displayed.

You can also insert tables that are linked by foreign key with one of the secondary tables that was already inserted. To do this, place the cursor on the secondary table and choose Relationships. Then proceed as described above.

For maintenance and help views, there are certain restrictions on the foreign keys with which the tables can be included in the view. The foreign keys violating these conditions are displayed at the end of the list under the header Relationships with unsuitable cardinality.

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

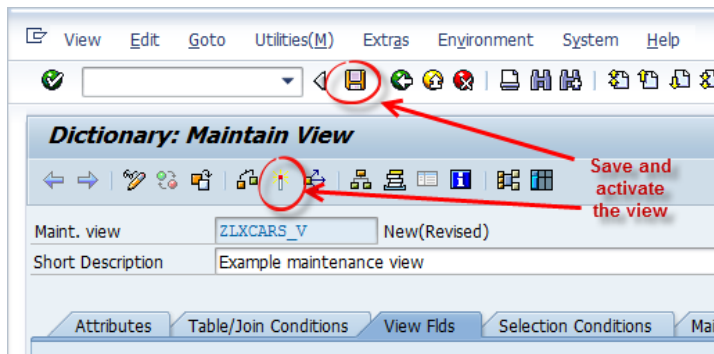
Choose Table fields. All the tables contained in the view are displayed in a dialog box. Select a table. The fields of the table are now displayed in a dialog box. You can copy fields by selecting them in the first column and choosing  Copy.

All key fields of the primary table must be included in a maintenance view. In addition, all key fields of secondary tables that are not involved in the foreign key (that is, which are not linked via a join condition to a key field already included in the view) must be included in the view.

This ensures that the records inserted with a maintenance view can be written correctly in the tables contained in the view.

On the Selection Conditions tab page, you can (optionally) formulate restrictions for the data records that can be displayed with the view.

The selection conditions define the data records that can be selected with the view.
In the Maintenance Status tab page, define the maintenance status of the view.



At activation, a log is written; it can be displayed with Utilities -> Activation log. If errors or warnings occurring when the view was activated, the activation log is automatically displayed.

Maintenance attributes

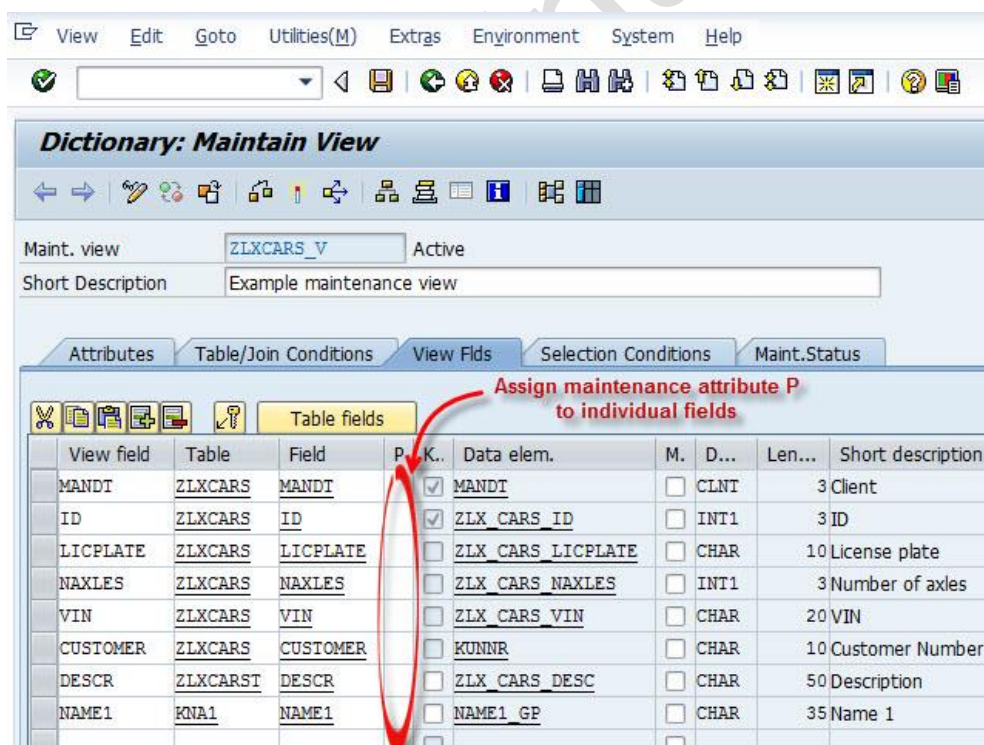
You can assign a maintenance attribute P to individual fields in the maintenance view definition in the Dictionary (transaction SE11). It can be R (read only), H (hidden), or S (subset).

- **R (read only)** - a field flagged R is not ready for input by the user on the screen. When a new data record is created, it must be filled automatically in the background. To fill a field automatically in the background, create a routine for an event.

- **H (hidden)** - a field flagged H is not displayed on the screen. This field must also be filled in the background by a routine at an event.

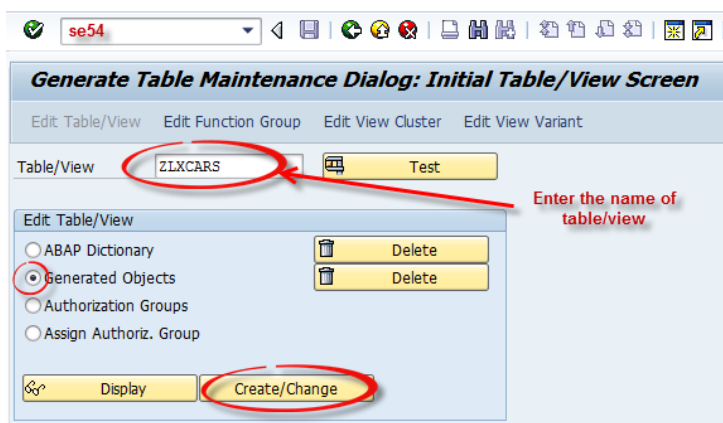
The system writes R and H fields to the database like all other fields when a new data record is saved.

- **S (subset)** - a field flagged S restricts the work area. The system only shows data records for which this subset field has the specified value. This restriction of the work area remains effective when you create new data records.

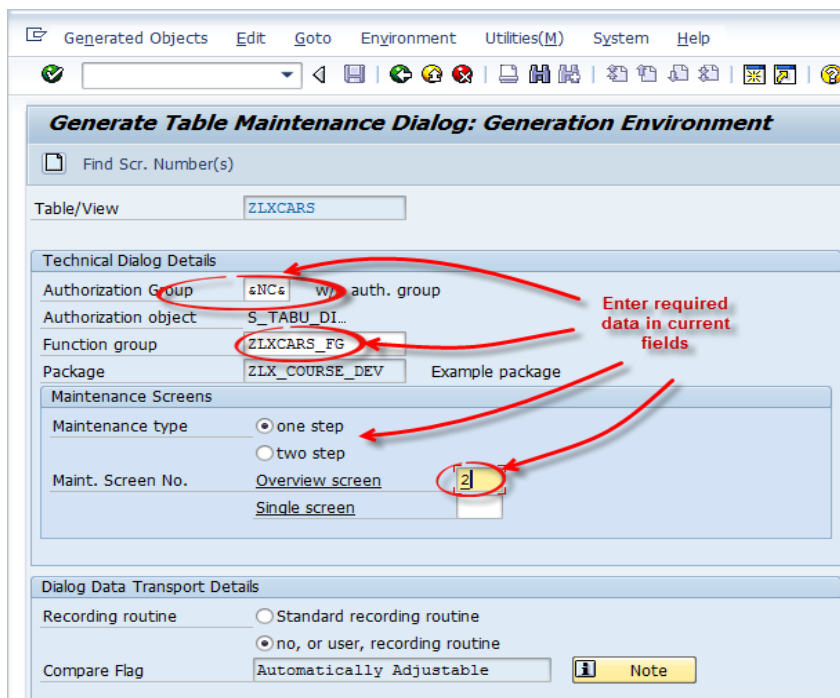


How to create a maintenance dialog

Maintenance dialogs can be created for SAP or customer tables or views. For creation of a maintenance dialog you should use transaction SE54. Also you can use transaction SE11. There you choose Table or View radio button, then click Change. In the next screen you should click on Utilities bar and then choose Table Maintenance Generator.



You should go to the maintenance screen for the generated object for the current table.



You must specify the following parameters in the maintenance dialog definition:

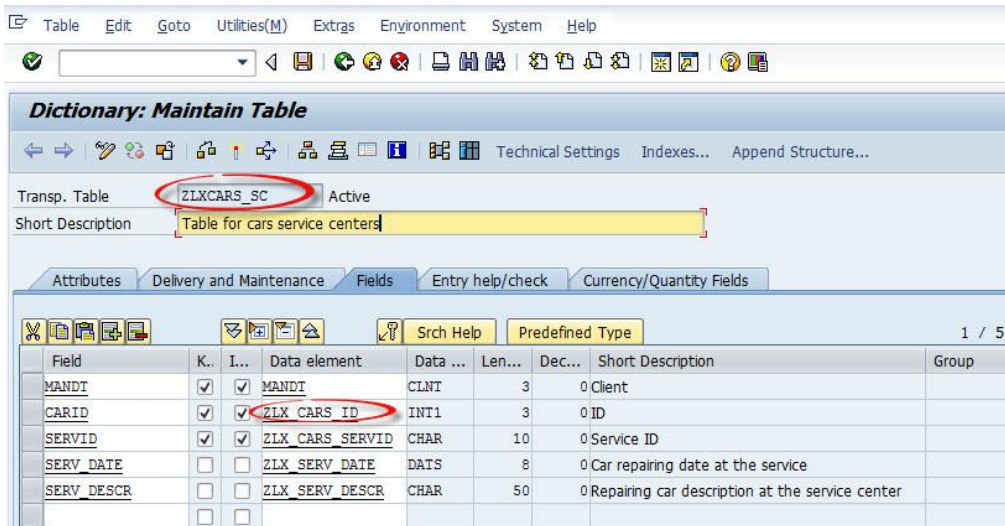
- **Function group:** the function group in which the tables/view-specific maintenance dialog components are generated. The function group is created if necessary.
- **Authorization group:** the users who are authorized to maintain the table/view contents.
- **Maintenance type:** one or two-step dialog.

One-step dialogs comprise only an overview screen containing all fields. In two-step dialogs, only the key and text fields with a length of more than 20 characters are displayed in the overview screen. All fields are offered in the detail screen.

- **Maintenance screens:** the internal number of each maintenance screen. You can get possible values in a search function.
- **Recording routine:** Specify whether and how the table/view contents maintained with the dialog are put in a transport.

How to create a view cluster

First of all you should define a dependent table. Use transaction SE16 to make entries to this table.



Dictionary: Maintain Table

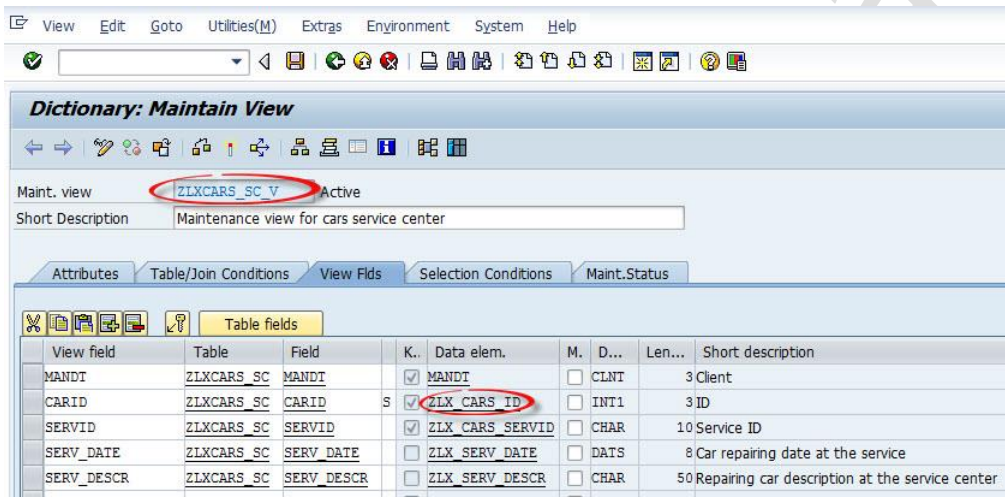
Transp. Table: ZLXCARS_SC Active

Short Description: Table for cars service centers

Attributes | Delivery and Maintenance | **Fields** | Entry help/check | Currency/Quantity Fields

Field	K..	I...	Data element	Data ...	Len...	Dec...	Short Description	Group
MANDT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	MANDT	CLNT	3	0	Client	
CARID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<u>ZLX_CARS_ID</u>	INT1	3	0	ID	
SERVID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	ZLX_CARS_SERVID	CHAR	10	0	Service ID	
SERV_DATE	<input type="checkbox"/>	<input type="checkbox"/>	ZLX_SERV_DATE	DATS	8	0	Car repairing date at the service	
SERV_DESCR	<input type="checkbox"/>	<input type="checkbox"/>	ZLX_SERV_DESCR	CHAR	50	0	Repairing car description at the service center	

After that you should create a maintenance view for this table using the method described above.



Dictionary: Maintain View

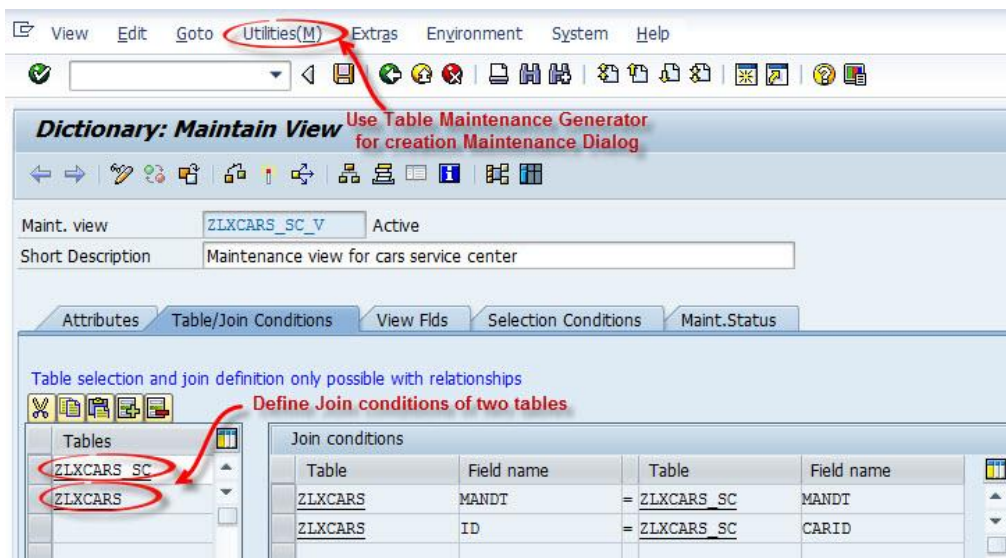
Maint. view: ZLXCARS_SC_V Active

Short Description: Maintenance view for cars service center

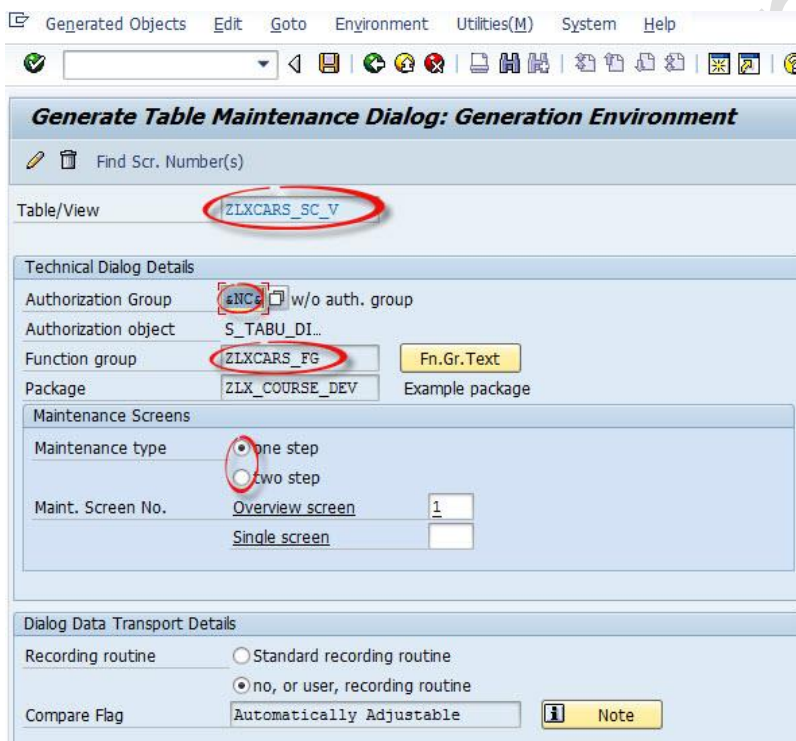
Attributes | Table/Join Conditions | **View Flds** | Selection Conditions | Maint.Status

View field	Table	Field	K..	Data elem.	M.	D...	Len...	Short description
MANDT	ZLXCARS_SC	MANDT	<input checked="" type="checkbox"/>	MANDT	<input type="checkbox"/>	CLNT	3	Client
CARID	ZLXCARS_SC	CARID	<input checked="" type="checkbox"/>	<u>ZLX_CARS_ID</u>	<input type="checkbox"/>	INT1	3	ID
SERVID	ZLXCARS_SC	SERVID	<input checked="" type="checkbox"/>	ZLX_CARS_SERVID	<input type="checkbox"/>	CHAR	10	Service ID
SERV_DATE	ZLXCARS_SC	SERV_DATE	<input type="checkbox"/>	ZLX_SERV_DATE	<input type="checkbox"/>	DATS	8	Car repairing date at the service
SERV_DESCR	ZLXCARS_SC	SERV_DESCR	<input type="checkbox"/>	ZLX_SERV_DESCR	<input type="checkbox"/>	CHAR	50	Repairing car description at the service center

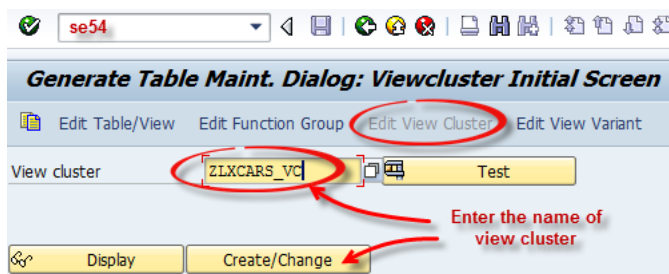
Pay attention to setting foreign keys to the table and defining Join Conditions.



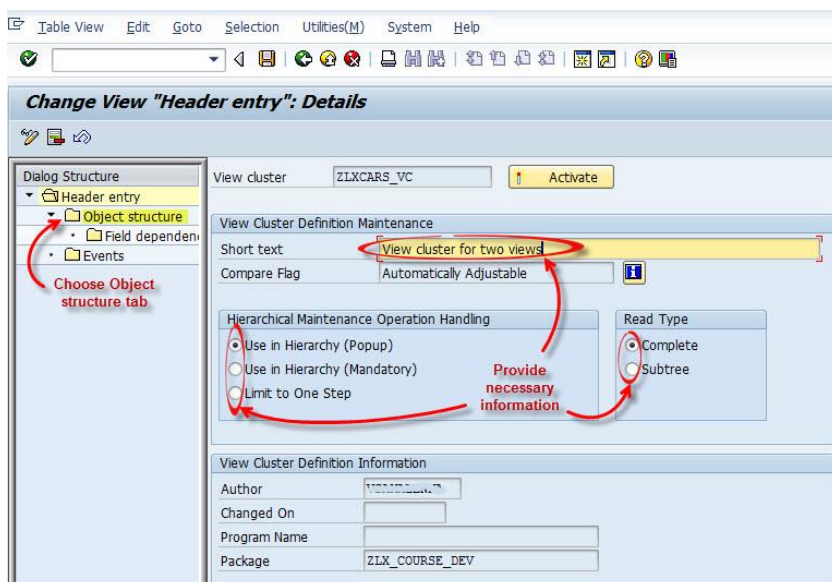
In the next step go to Utilities(M) tab and choose Table Maintenance Generator for generating Table Maintenance Dialog.



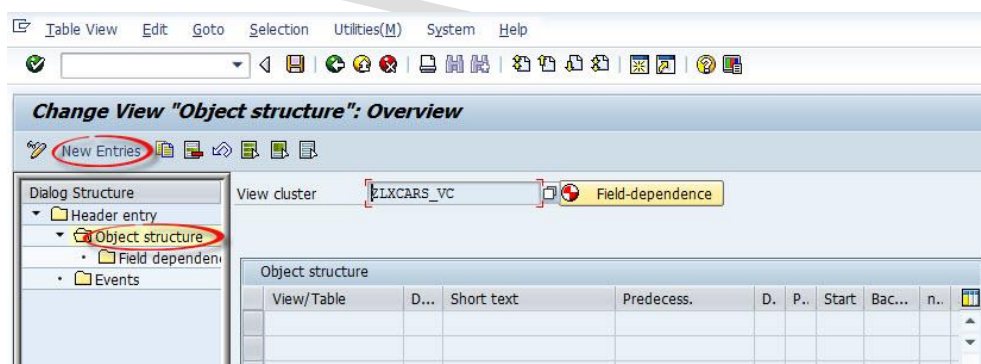
Only after that you can go to transaction SE54 to create a view cluster.



A view cluster is a group of maintenance dialogs which are collected in one maintenance unit for business or technical reasons. They allow related data in more than one table/view to be maintained consistently.



After providing all necessary information in Header entry screen go to Object Structure tab and press New entries button.



Whereas only 1:1 relationships can be processed in maintenance views (except for language-dependent texts), key extensions and relationships of cardinality N:M can also be handled in view clusters. Maintenance dialogs with no key, or partial key-dependence, can also be combined into view clusters.

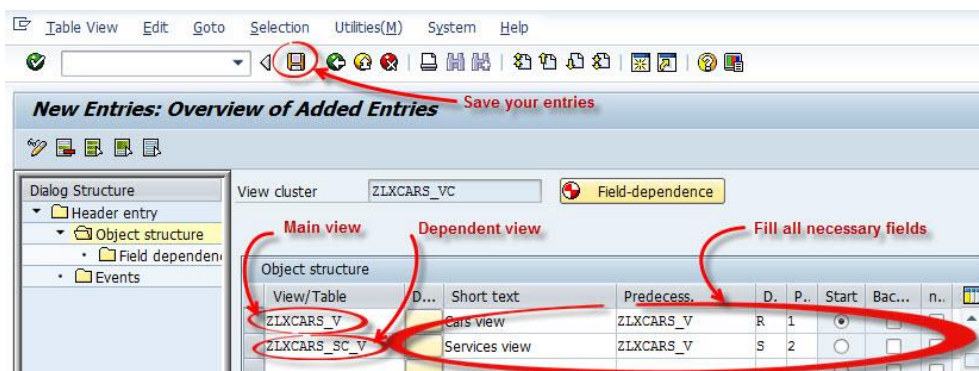
Grouping dialogs into one maintenance unit has the following advantages for data maintenance:

- **Navigation:** A view cluster contains convenient standard navigation between the individual maintenance dialogs. This simplifies the maintenance of the data in a view cluster.
- **Consistency:** The view cluster ensures data consistency when deleting, copying, saving, retrieving and manually transporting data. For example when an entry in a higher-level view is deleted, it automatically ensures that all dependent entries in lower-level views are also deleted.

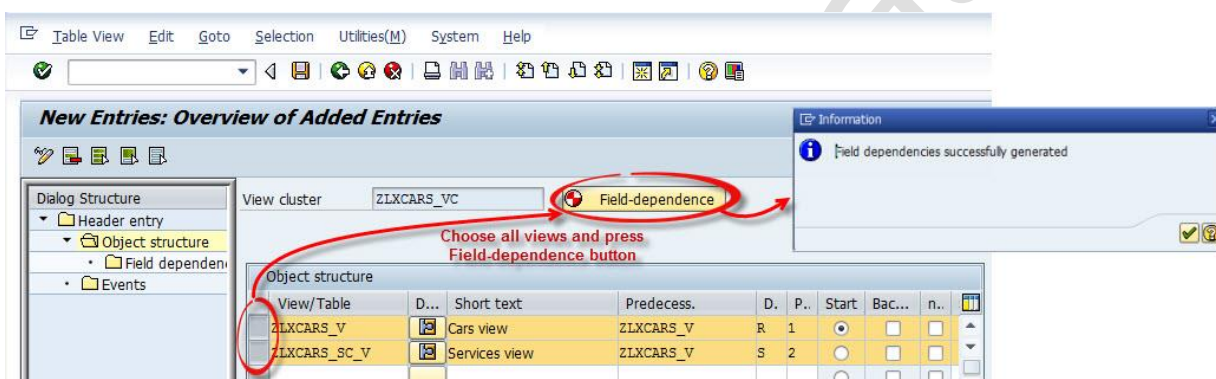
You can first split tables with a lot of fields into various views and then collect them into a view cluster. Use tabs to format very wide tables with a lot of fields.

Navigation in a view cluster is usually based on the hierarchy of the tables/views on which the dialogs are based. A view cluster usually consists of one or more root dialogs and the at most 14 maintenance dialogs which depend on them. A view at a lower level has one or more additional fields in its key compared its higher-level view. Each data record at the higher level has several dependent data records below it.

Provide all necessary data for each maintenance dialog and save the changes.



For Object structure define Field-dependence. The Information screen should appear.



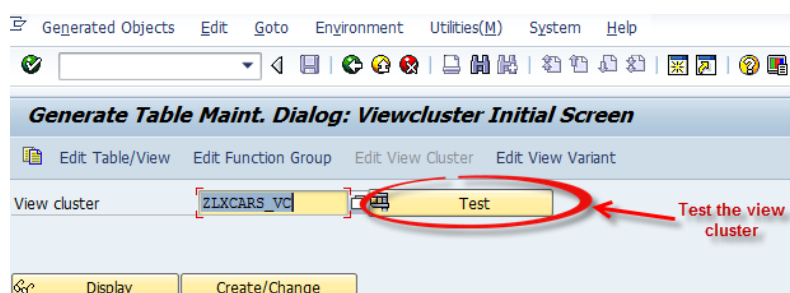
Then go to Header entry tab and activate the view cluster.

Each maintenance dialog is an independent unit consisting of an overview screen or an overview and a detail screen, depending on the dialog type. As well as the data to be maintained, the overview screen contains a navigation area, in which you can go to higher or lower level maintenance dialogs.

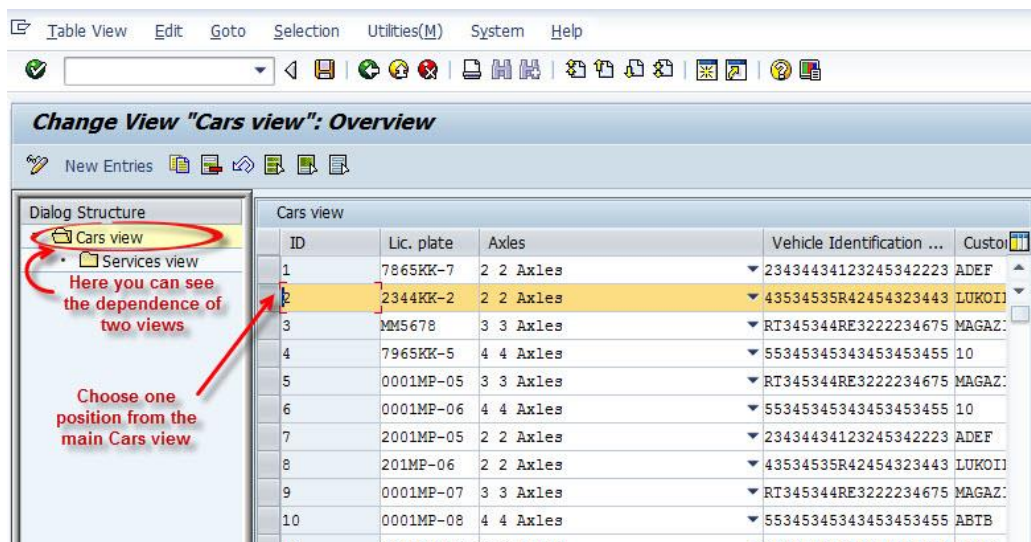
One-level maintenance dialogs only contain one screen, the list screen, in which all existing data records are displayed in a table. In a two-level maintenance dialog, double-clicking on a data record in the list screen takes you to the detail screen, which displays all the fields in the selected data record.

When you generate the maintenance dialog, flag the key fields which are identical in the view and its higher-level in the cluster, with the maintenance attribute S. The system fills the key fields when you go from the upper to the lower level, with the values of the data record selected in the upper level.

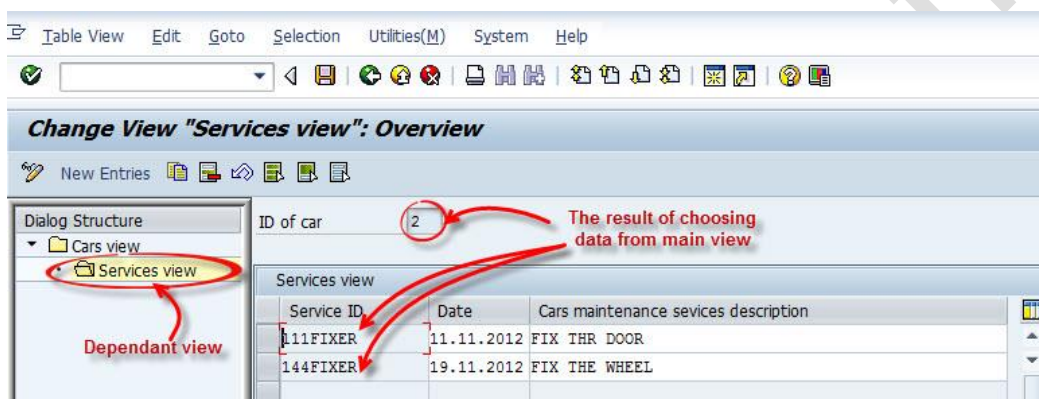
From the initial screen of the transaction SE54 give the view cluster name and click on Test button.



You can observe all the existing entries of the main view.



Choosing the row from the main view you can see the data from dependent view.

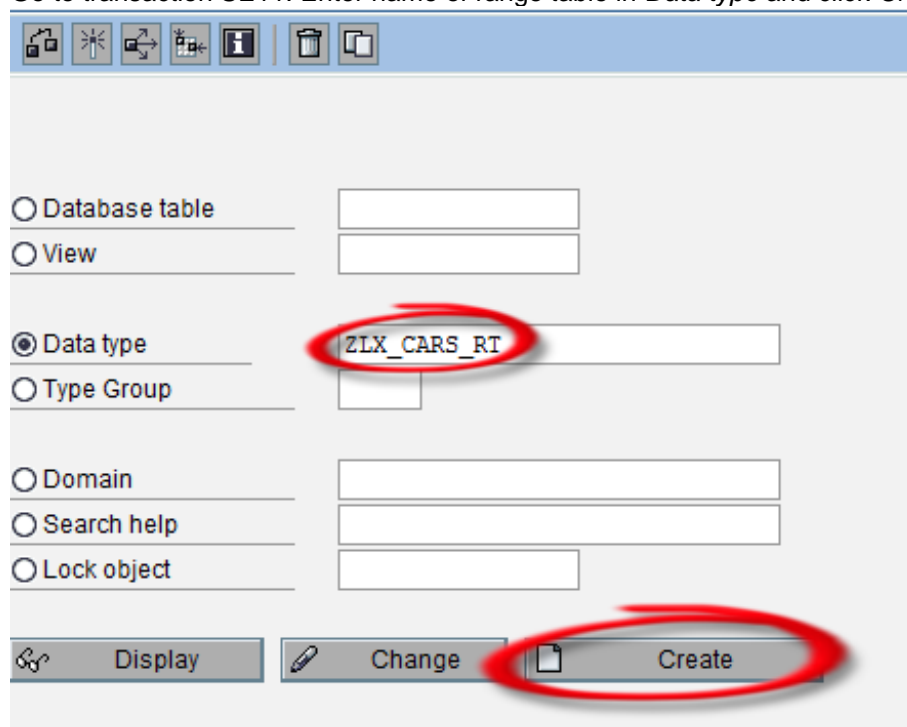


How to create a Range table

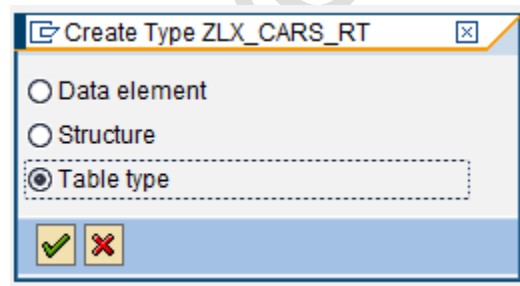
A ranges table type is a special case of a table type. A ranges table type describes the structure of an internal table for administering complex areas, i.e. the type of an internal ranges table in the ABAP program. Ranges tables can be used for example in logical conditions (IN operators) in the SELECT, IF, WHILE and CHECK statements or to pass data to selection tables. The row type of a ranges table type has a fixed structure. The row type consists of the 4 components SIGN (sign), OPTION (comparison operator), LOW (lower limit) and HIGH (upper limit) in this order. The type of components LOW and HIGH is defined by an elementary associated type. It can be defined by specifying a data element or by directly defining the data type, number of places and if necessary the number of decimal places.

- You can pass value to sign as either 'I' (Include) or 'E' (Exclude)
- Option can be used for operand like 'EQ' (Equal to), 'NE' (Not Equal To), 'BT' (Between)
- Low is the actual value that you need to pass in the table
- High option is not needed unless using the BT option

Go to transaction SE11. Enter name of range table in *Data type* and click *Create* button.



Select Table type



Provide *Short text* and select menu Edit->Define as range table type.

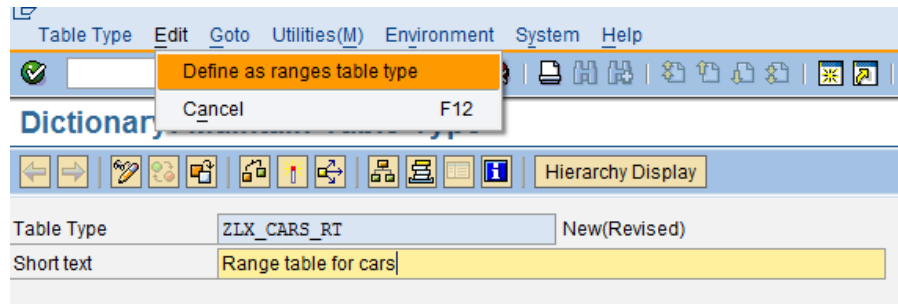
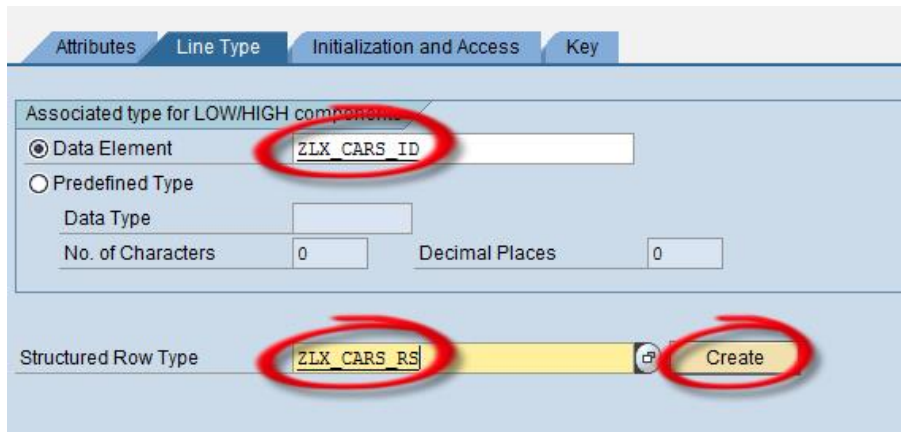


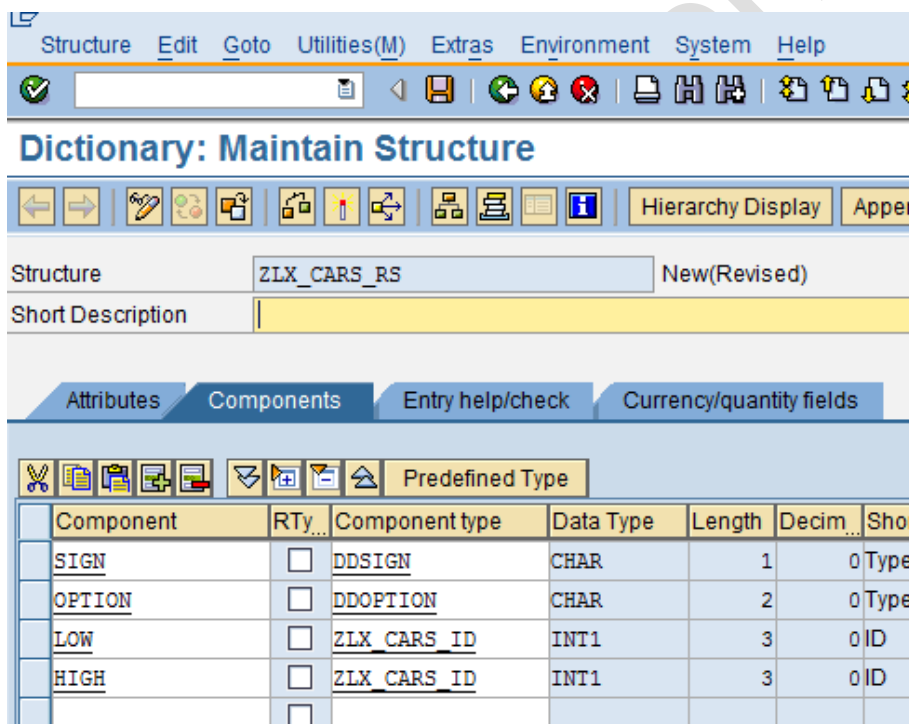
Table Type: ZLX_CARS_RT New(Revised)
Short text: Range table for cars

Enter *Data Element* and *Structured Row Type*, click *Create* button



Associated type for LOW/HIGH components:
☒ Data Element: ZLX_CARS_ID
☐ Predefined Type
Data Type:
No. of Characters: 0 Decimal Places: 0
Structured Row Type: ZLX_CARS_RS Create

Now you can see the screen



Structure: ZLX_CARS_RS New(Revised)
Short Description:

Component	RTy...	Component type	Data Type	Length	Decim...	Shor
SIGN	<input type="checkbox"/>	DDSIGN	CHAR	1	0	Type
OPTION	<input type="checkbox"/>	DDOPTION	CHAR	2	0	Type
LOW	<input type="checkbox"/>	ZLX_CARS_ID	INT1	3	0	ID
HIGH	<input type="checkbox"/>	ZLX_CARS_ID	INT1	3	0	ID

Provide *Short Description*, save and activate Range table and Range string.

You can use Range table in select statement.

Create Range table:

```
DATA : lt_cars TYPE TABLE OF zlxcars,
grt_cars_id TYPE zlx_cars_rt,
grs_cars_id LIKE LINE OF grt_cars_id.
```

Now you need assign data to the range table.

First method: assign data from select-option.

```
MOVE-CORRESPONDING s_id TO grs_cars_id.
```

```
APPEND grs_cars_id TO grt_cars_id.
```

Or with this syntax:

```
grt_cars_id = s_id[].
```

Second method: fill all fields by hands.

```
grs_cars_id-sign = 'I'.
grs_cars_id-option = 'EQ'.
grs_cars_id-low = '1'.
```

```
APPEND grs_cars_id TO grt_cars_id.
```

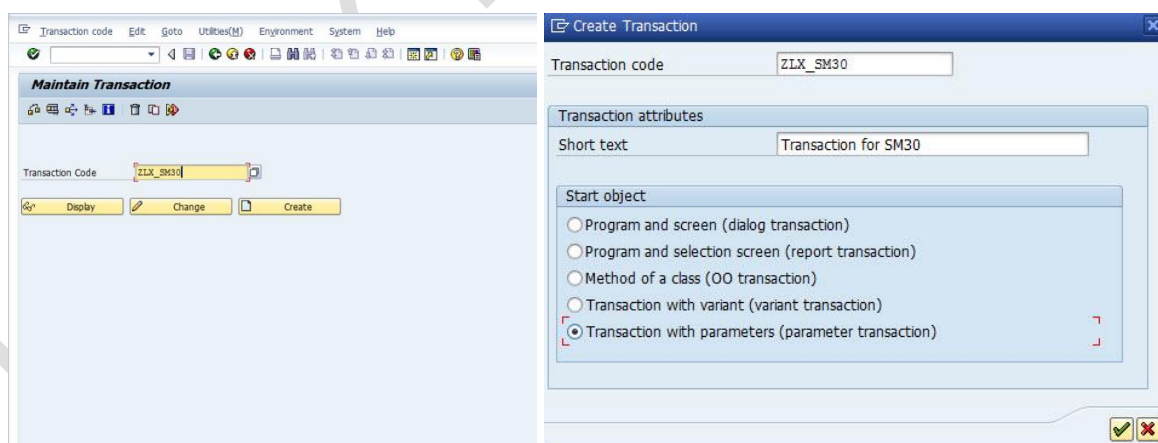
Now you can use your Range table in SELECT statement.

```
SELECT *
INTO CORRESPONDING FIELDS OF TABLE lt_cars
FROM zlxcars
WHERE zlxcars~id IN grt_cars_id.
```

How to create transaction with parameters (for SM30)

To create Transaction, select the menu path Tools → ABAP Workbench → Development → Other Tools → Transactions from the SAP menu of SAP Easy Access (or start Transaction SE93). Enter desired transaction code and create:

1. Put "Short text"
2. Select option "Parameter Transaction"



2. Enter SM30 as the default value for the transaction
3. Check "Skip initial screen"
4. Scroll down. In the table control at the bottom enter the following default parameters:
 1. VIEWNAME with VALUE = <custom table view>
 2. UPDATE with VALUE = X
5. Save

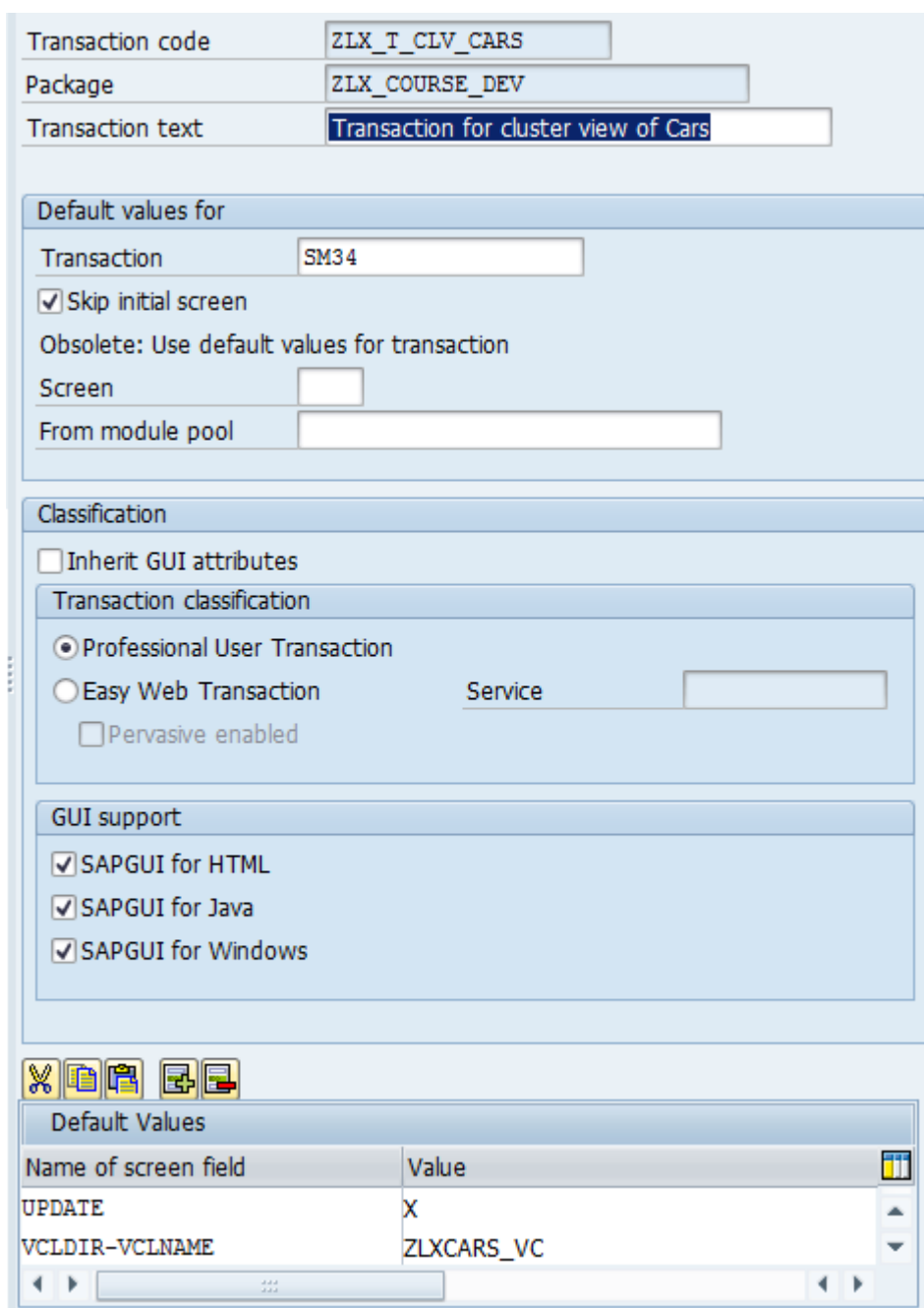
Other valid parameter values are:

PARAMETER	VALUE	ACTION
SHOW	X or <space>	Display
TRANSPORT	X or <space>	Transport
UPDATE_LTD	X or <space>	Maintain subset
SHOW_LTD	X or <space>	Display subset
TRANSP_LTD	X or <space>	Transport subset
TVIMV-VARIANT	Variant name	Use variant (only with UPDATE_LTD, SHOW_LTD and TRANSP_LTD)

How to create transaction for a view cluster (for SM34)

To create Transaction for a view cluster you need to start a transaction SE93. Enter desired transaction code and press 'Create':

1. Put explanatory text.
2. Select option "Parameter Transaction".
3. Enter **SM34** as the default value for the transaction
4. Check "Skip initial screen"
5. Scroll down. In the table control at the bottom enter the following default parameters:
 1. UPDATE with VALUE = X
 2. **VCLDIR-VCLNAME** with VALUE = <custom view cluster>
6. Save

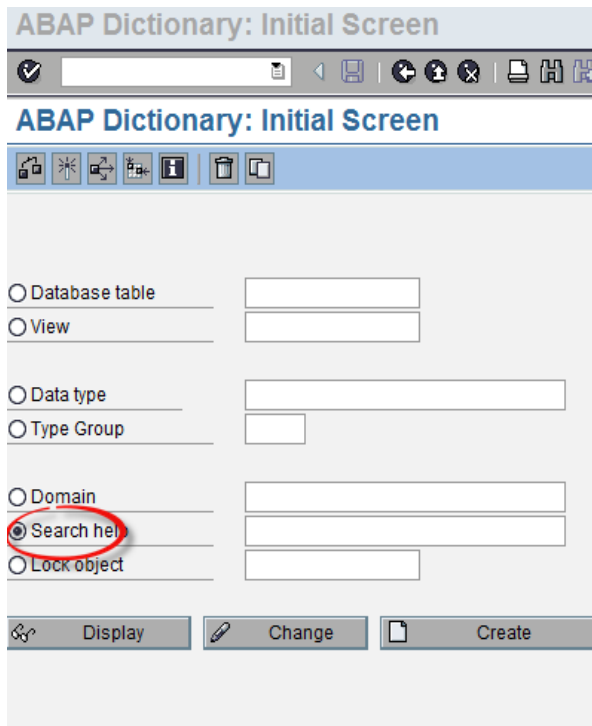


The screenshot shows the SAP SE93 transaction creation interface. The top section contains fields for Transaction code (ZLX_T_CLV_CARS), Package (ZLX_COURSE_DEV), and Transaction text (Transaction for cluster view of Cars). Below this is the 'Default values for' section, where Transaction is set to SM34, 'Skip initial screen' is checked, and 'Obsolete: Use default values for transaction' is selected. The 'Classification' section shows 'Professional User Transaction' selected. The 'GUI support' section has 'SAPGUI for HTML', 'SAPGUI for Java', and 'SAPGUI for Windows' all checked. At the bottom, the 'Default Values' table is visible, containing the following data:

Name of screen field	Value
UPDATE	X
VCLDIR-VCLNAME	ZLXCARS_VC

How to create a search help

Go to transaction SE11. Choose the radio button Search help.



ABAP Dictionary: Initial Screen

Database table

View

Data type

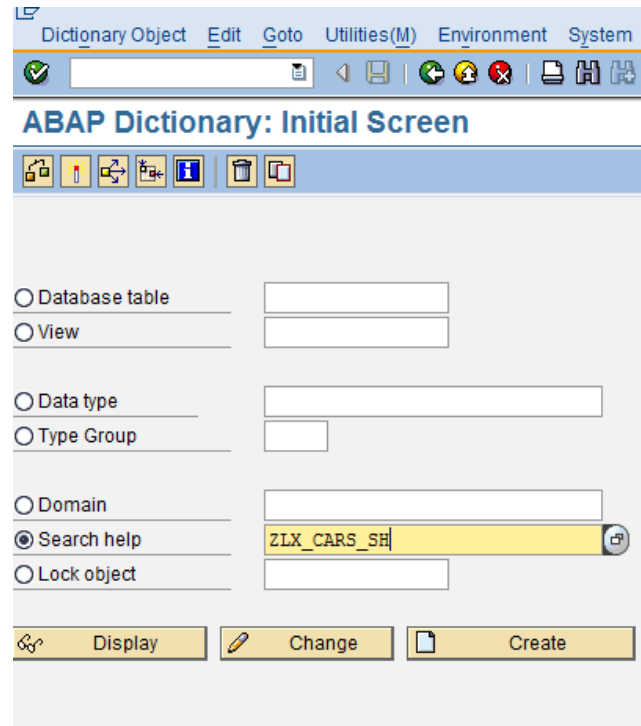
Type Group

Domain

☒ Search help

Lock object

Display Change Create



ABAP Dictionary: Initial Screen

Database table

View

Data type

Type Group

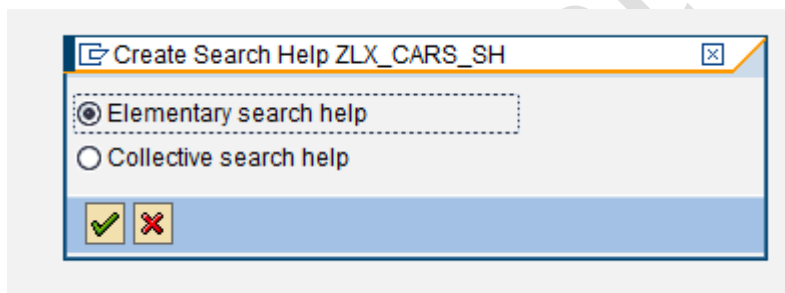
Domain

☒ Search help

Lock object

Display Change Create

Provide the Search help name. Select the Create button.
Select Elementary search help.



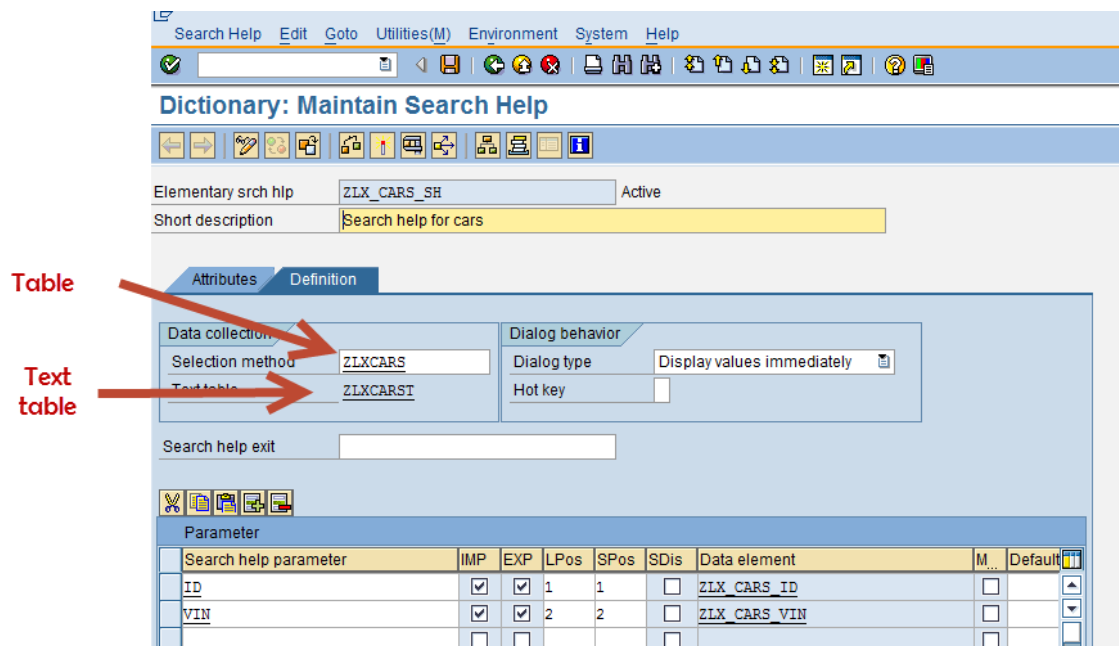
Create Search Help ZLX_CARS_SH

☒ Elementary search help

☐ Collective search help

✓ ✗

Provide the Short description (You can enter the name of a table or a 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), the Selection method. Provide the fields.



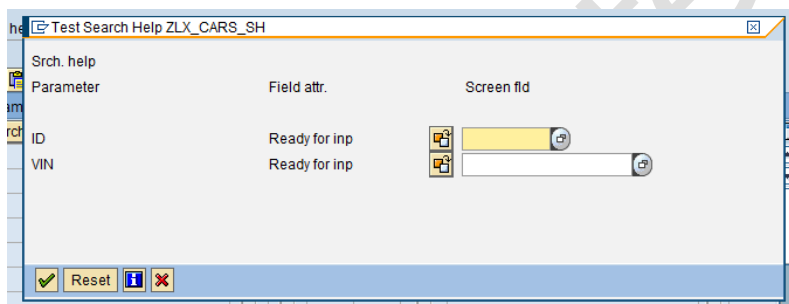
Table

Text table

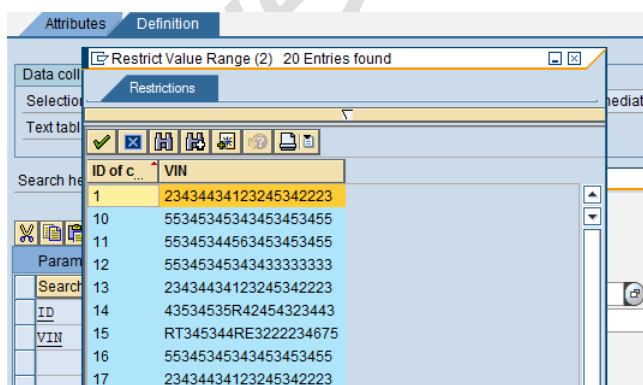
Search help parameter	IMP	EXP	LPos	SPos	SDis	Data element	M...	Default
ID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	1	1	<input type="checkbox"/>	ZLX_CARS_ID	<input type="checkbox"/>	
VIN	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	2	2	<input type="checkbox"/>	ZLX_CARS_VIN	<input type="checkbox"/>	

Save it and activate it.

Then execute it. The Test Search Help Screen will pop up. Choose necessary parameter using F4 button.



The screen with Restrict Value Range will pop up.

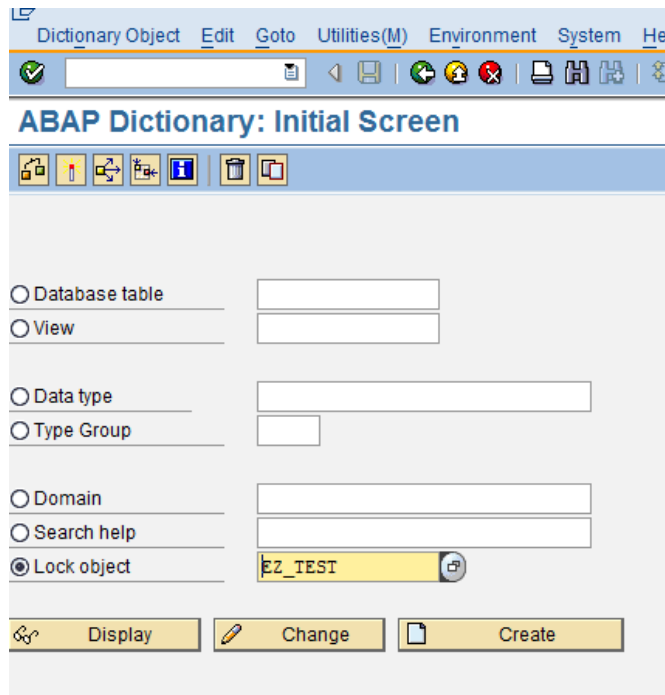


ID of c...	VIN
1	23434434123245342223
10	55345345343453453455
11	55345344563453453455
12	55345345343433333333
13	23434434123245342223
14	43534535R42454323443
15	RT345344RE3222234675
16	55345345343453453455
17	23434434123245342223

This is the output for elementary search help.

How to create a Lock object

SAP provides you with the ability to restrict access to data while the table is being updated. Go to transaction SE11. Select Lock object and provide name (Lock object name should always start from 'E').

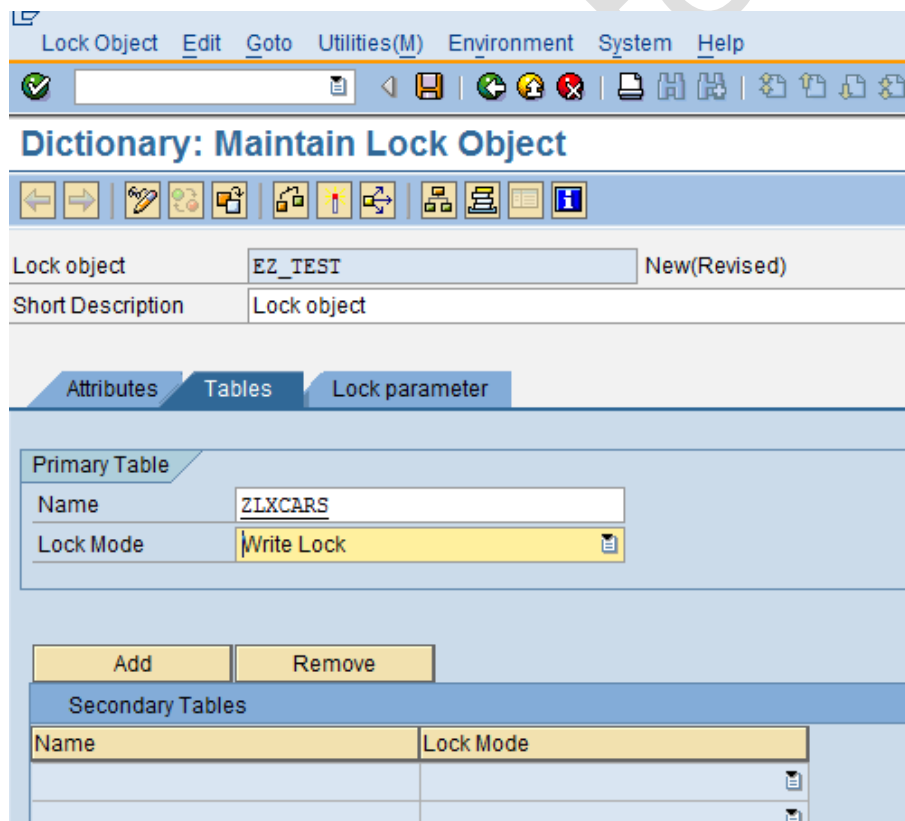


ABAP Dictionary: Initial Screen

☐ Database table
☐ View
☐ Data type
☐ Type Group
☐ Domain
☐ Search help
☒ Lock object

Display Change Create

Provide short description. Open tab Tables. Provide table name and select Lock Mode.



Dictionary: Maintain Lock Object

Lock object: EZ_TEST New(Revised)
 Short Description: Lock object

Attributes Tables Lock parameter

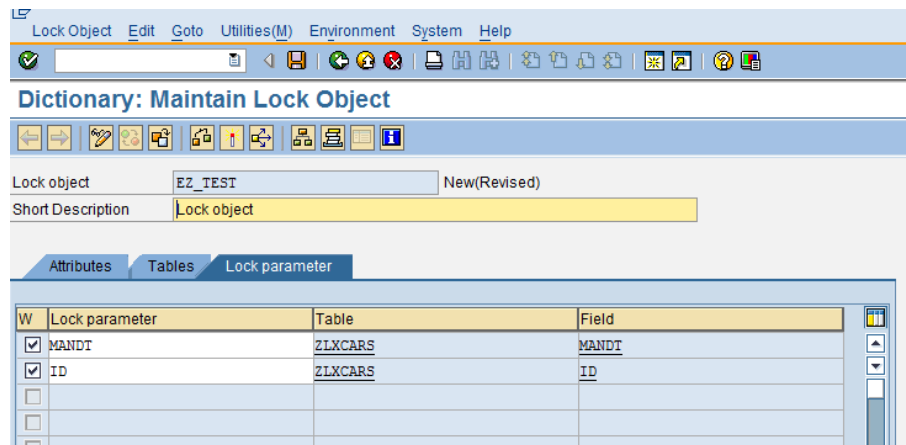
Primary Table
 Name: ZLXCARS
 Lock Mode: Write Lock

Add Remove

Secondary Tables

Name	Lock Mode

Open tab Lock parameter. Here you can see all the key fields from your table.



Now save and activate the Lock object. SAP creates two function modules ENQUEUE_<lockobjectname> and DEQUEUE_<lockobjectname> are generated from its definition to set and release locks. Open source code and add the following code in-order to create the table lock. This function module must be called before any update takes place. If a lock has already been taken out it will display the appropriate message.

CALL FUNCTION 'ENQUEUE_EZ_TEST'
EXPORTING

```
mode_zlxcars      = 'E'
* MANDT           = SY-MANDT
* ID              =
* X_ID            = ''
* _SCOPE          = '2'
* _WAIT           = ''
* _COLLECT        = ''
* EXCEPTIONS
* FOREIGN_LOCK    = 1
* SYSTEM_FAILURE  = 2
* OTHERS          = 3.
```

IF sy-subrc <> 0.

MESSAGE s000(ztest).

ENDIF.

The following code will remove the lock for the specific table entries.

CALL FUNCTION 'DEQUEUE_EZ_TEST'
EXPORTING

```
mode_zlxcars      = 'E'
* MANDT           = SY-MANDT
* ID              =
* X_ID            = ' '
* _SCOPE          = '3'
* _SYNCHRON       = ' '
* _COLLECT        = ' '.
```