

Top Five ways to find a table and field within a transaction

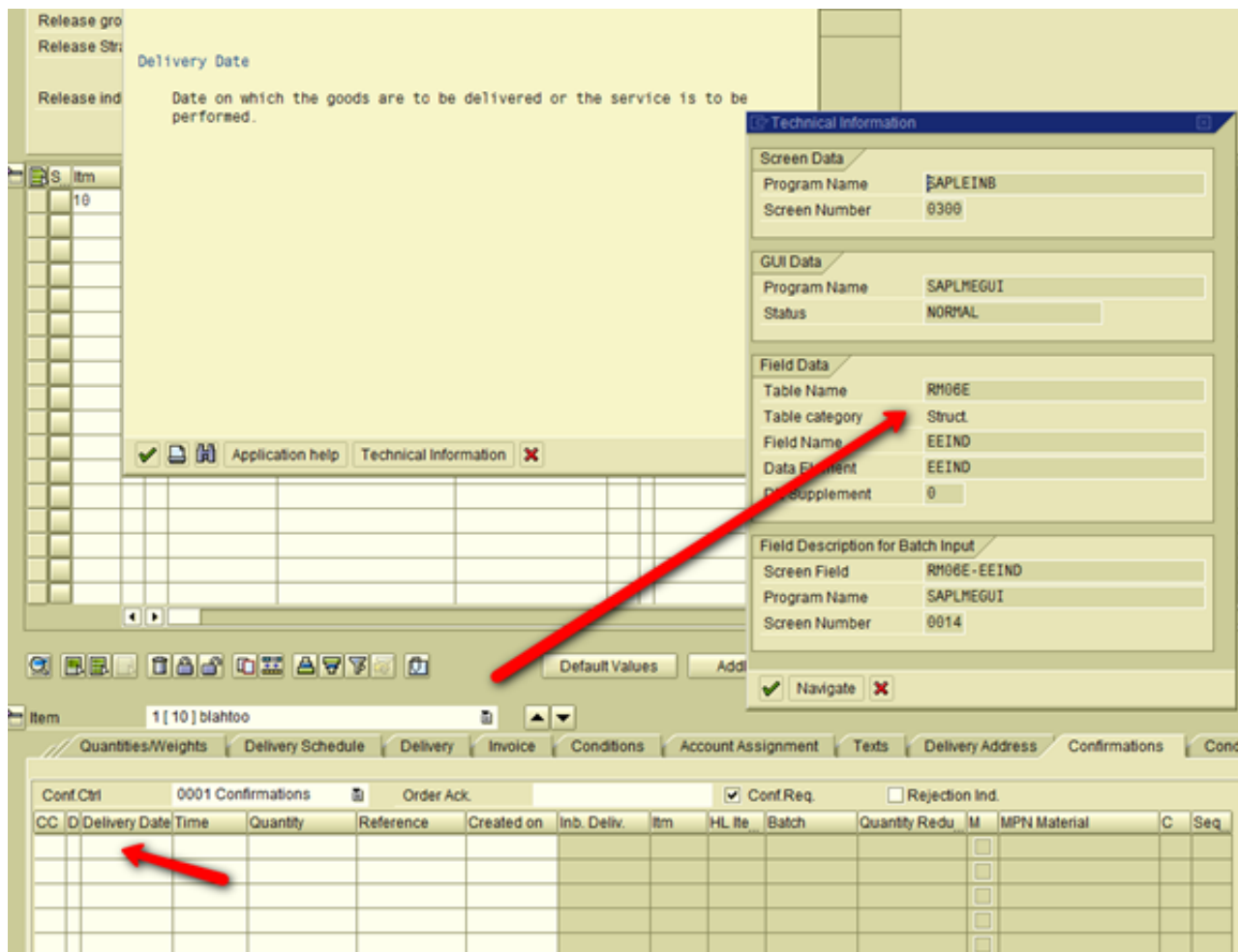
January 5th, 2010 | Categories: Article, Consulting Tools, Document Type, SAP, Technical | Tags: Database Table,SAP

Have you ever been frustrated trying to find which table and field a piece of data is stored in. You can see it on the screen, and the old faithful F1 – F9 results in some useless structure information. Or have you ever started looking at a piece of functionality you are unfamiliar with wanting to find the table structures behind it in SAP. Well this article shows my favorite five ways of digging under the hood to find out what's going on. Let me just preface this by saying I'm a purely functional consultant and I'm sure the more ABAP literate probably have their own ways. If so I'd appreciate hearing from you. But anyhow here's my top 5 (in no particular order).

#1 Use a Different Field

when you click on a field then hit F1-F9 and it shows a structure and not a real field, sometimes the easiest way to find the real transparent table behind is to try another field on the same area of the screen.

In the example below I want to know where SAP stores purchase order confirmation information. On the confirmation tab I click on the delivery date field and press F1 then F9 (technical information) in this case it shows me a structure and not a transparent table.



So trying another field like quantity I get the answer I'm looking for

Technical Information

Screen Data
 Program Name: SAPLEINB
 Screen Number: 0300



GUI Data
 Program Name: SAPMEGUI
 Status: NORMAL

Field Data
 Table Name: EKES
 Table category: Transparent table
 Field Name: MENGE
 Data Element: BBERG
 DE Supplement: 0

Field Description for Batch Input
 Screen Field: EKES-MENGE
 Program Name: SAPMEGUI
 Screen Number: 0014

✓ Navigate ✗

#2 Use Where Used on the Data Element

An alternative when faced with a structure is to use the  where used  on the data element. This can have mixed results but can often show a list of tables that can be quite useful.

From within the technical information pop up (F1-F9) click on the data element then press navigate

Dictionary: Display Data Element

Data element: EEEND
 Short Description: Delivered O...

Attributes | Data Type | Further Characteristics | Field Label

Elementary Type
 Domain
 Data Type: CHAR
 Length: 10
 Character String
 Decimal Places: 0

Predefined Type
 Data Type: ...
 Length: ...
 Decimal Places: ...

Reference Type
 Name of Ref. Type: ...

Reference to Predefined Type
 Data Type: ...
 Length: ...
 Decimal Places: ...

Where-Used List Data Element

Data Element: EEEND

Used in

☒ Table Fields
☐ Struct. Fields
☐ View Fields
☐ Table types
☐ Search Helps
☐ Data elements
☐ Entity Type Attributes

☐ Programs
☐ Classes/Interfaces
☐ Web Dynpro Component
☐ BSP Applications
☐ Function Module Interfaces
☐ Service Definitions
☐ Package Interfaces

☐ Test Scripts (eCATT)
☐ Test Data (eCATT)

✓ In the Background Indirect Application Search Range ✗

Where-used Data Element BBERD in Table Fields (2 Hits)

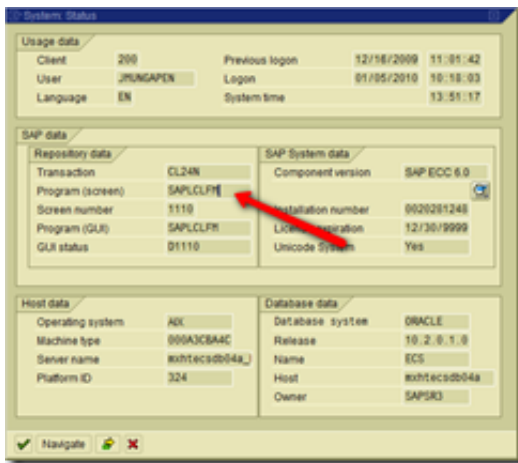
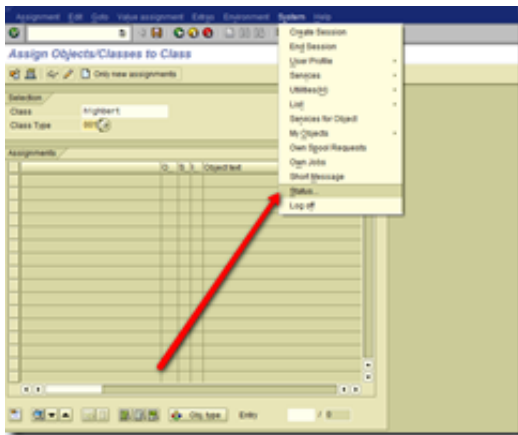
Table Fields	Short description
EKES	Vendor Confirmations
ERDAT	Creation Date of Confirmation
PERKS	Variant Vendor Confirmations
ERDAT	Creation Date of Confirmation

In the above example there were only two choices, I could then view both of these tables using SE16N to see if the data relating to my purchase order is there.

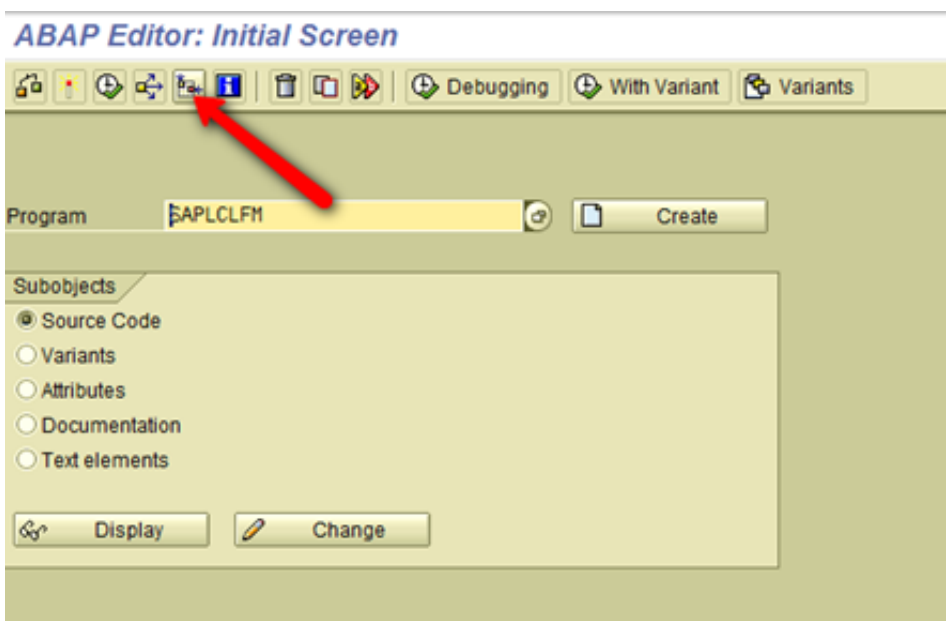
#3 Environmental Analysis


This is one of my current favorites for finding all the tables related to a transaction I might be unfamiliar with. Its a bit broader than finding a specific field and table on a screen, but very powerful and quick. For example if I wanted to find out how SAP stores object classification information i would proceed as follows. Find a suitable transaction that focuses on object classification, there are a number but I'll use CL24N for this example.

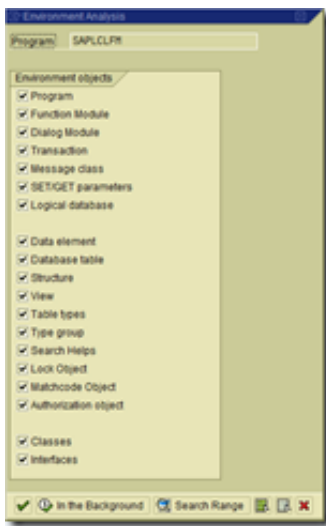
from within the transaction find out what the program name is using System > Status



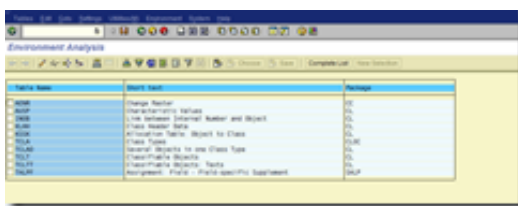
From here you can find the program name in this case SAPLCLFM. Copy the program name into the clipboard then in a new session enter it into an SE38 transaction.



From here you can select the environmental analysis option.  next you get a list of objects to analyze for.



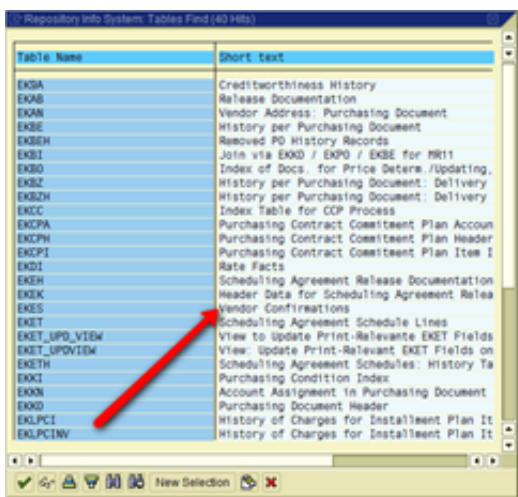
you can either just select tables or leave them all selected and execute. This gives the resulting tables used by the transaction code along with the package code which can be used in the SE80 method above.



I generally ignore the T tables as configuration tables and can quickly understand how the tables interact by looking at their data in Se16N.

#4 Pot Luck in Table Naming / SE80 Object Navigator

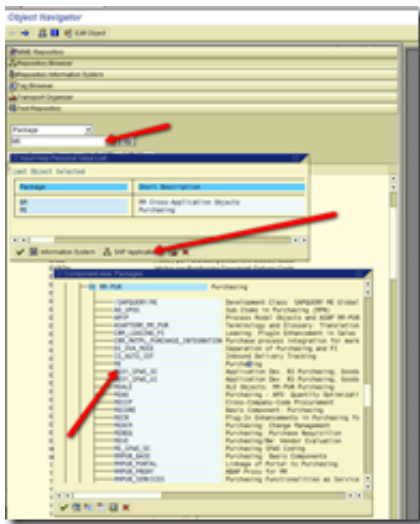
I often use this method to get a *feel* for what tables are related to a certain piece of functionality as SAP often uses a fairly consistent naming convention for tables. In my purchase order example I know that the PO header table is EKKO and that PO item table is EKPO, so within SE16N I can search for all tables starting with EK* to see if anything pops out



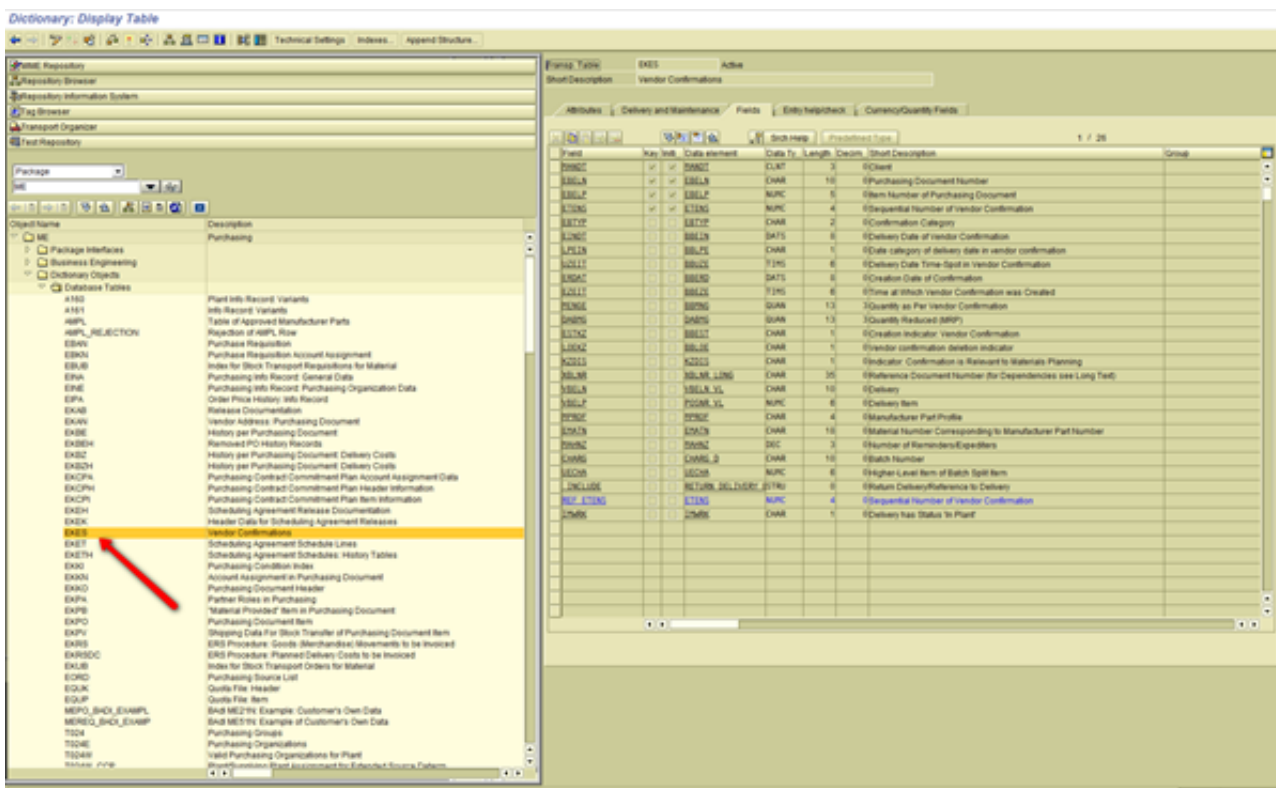
from the list above I can see that vendor confirmations is EKES.

A similar but more comprehensive way of achieving this is to use the SE80 Object navigator transaction in SAP, here you can find all the data dictionary related objects associated with some functionality in one place.

I tend to use the *package* as a way of finding objects and the application hierarchy to identify the correct package.



In this example we can look at all the dictionary objects in package ME



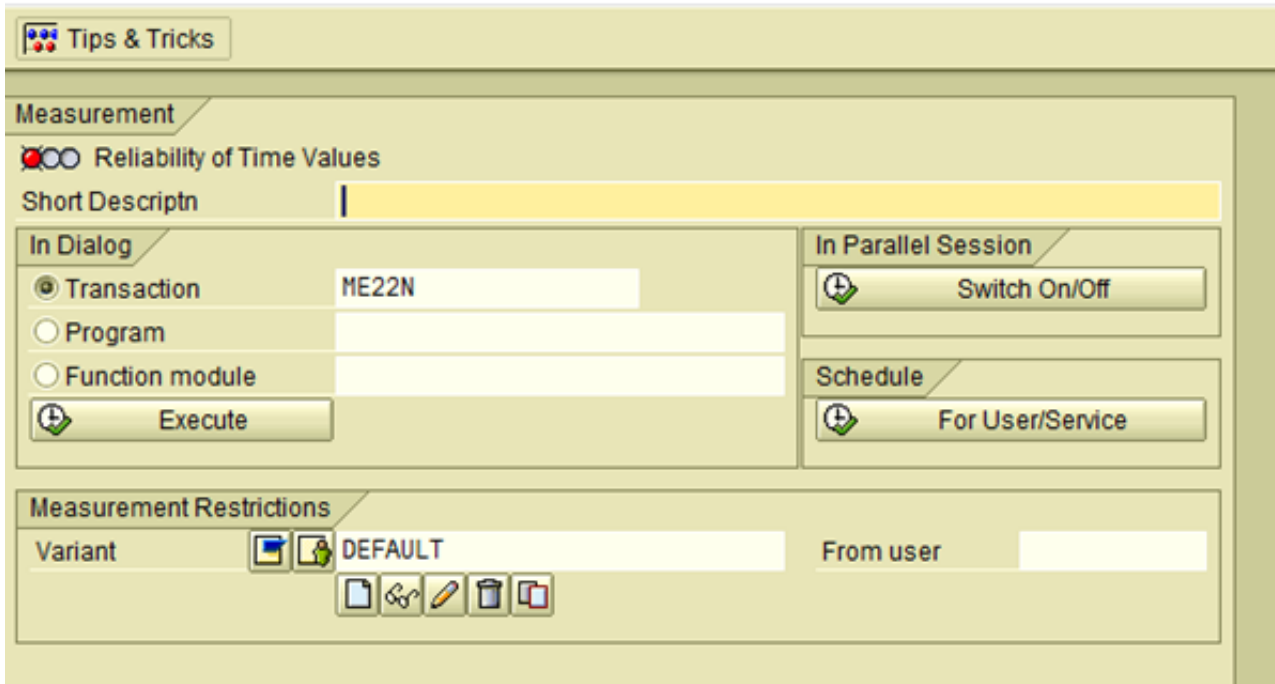
#5 Traces (Runtime Analysis and SQL trace)

If all else fails you can always trace the transaction to see what its up to and SAP provides two very useful traces that can be as useful for functional consultants as developers. About ten years ago I used to use these traces all the time, I must say i use them much more infrequently these days as I think the repository tools mentioned above have become more sophisticated (or maybe I have become more sophisticated). However sometimes performing a trace on a transaction gives you an idea what the program is up to.

SE30 Runtime Analysis

Executing transaction SE30 allows you to perform the desired transaction from within the trace

ABAP Runtime Analysis: Initial Screen



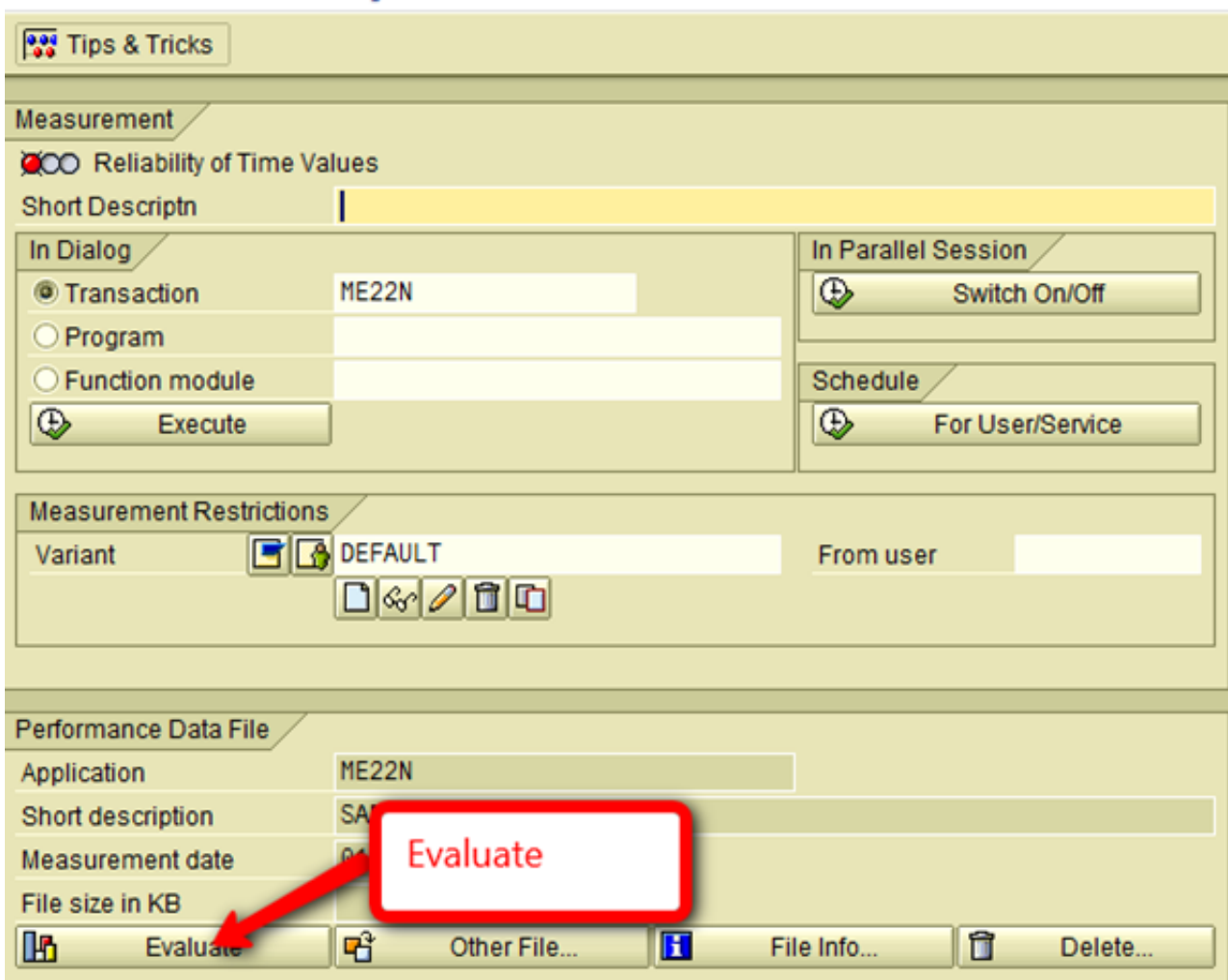
The screenshot shows the 'ABAP Runtime Analysis: Initial Screen'. At the top is a 'Tips & Tricks' button. Below it is the 'Measurement' section, which includes a 'Reliability of Time Values' toggle (currently off), a 'Short Descriptn' field, and two sub-sections: 'In Dialog' and 'In Parallel Session'. The 'In Dialog' section has radio buttons for 'Transaction' (selected), 'Program', and 'Function module', with a corresponding 'Execute' button. The 'In Parallel Session' section has a 'Switch On/Off' button. Below these is a 'Schedule' section with a 'For User/Service' button. The 'Measurement Restrictions' section at the bottom has a 'Variant' field set to 'DEFAULT' and a 'From user' field. A toolbar with icons for file operations is located below the variant field.

Give the trace a name then enter the tcode you want to analyze and execute.

run though the transaction make sure you enter data in the field or area of the screen you want to track. Complete the transaction by saving and you will return to the trace screen.

now you can evaluate the trace

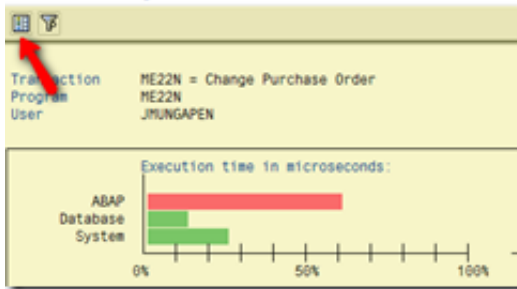
ABAP Runtime Analysis: Initial Screen




This screenshot is identical to the one above, but it includes a 'Performance Data File' section at the bottom. This section contains fields for 'Application' (ME22N), 'Short description' (SA), 'Measurement date' (01), and 'File size in KB'. Below these fields is a toolbar with four buttons: 'Evaluate', 'Other File...', 'File Info...', and 'Delete...'. A red arrow points from the 'Evaluate' button to a red-bordered box containing the word 'Evaluate' in red text.


The initial screen shows performance information which we are not interested in

Runtime Analysis Evaluation: Overview




select the hit list button  You now get a sequential list of what the transaction did, which can be overwhelming

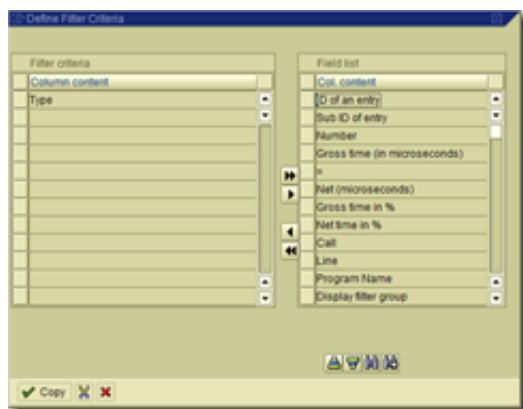
Runtime Analysis Evaluation: Hit List



No.	Gross	Net	Gross (%)	Net (%)	Call	Program Name	Type	No.	Filter
1	4,109,679	0	100.0	0.0	Runtime analysis		Sys.		
1	4,105,300	23,499	99.9	0.6	Call Transaction ME22N	SAPRS38T			
1	4,060,172	474	99.0	0.0	Program SAPPO_GUI				
1	4,058,398	1,145	98.8	0.0	Call Func. LCL_MAINTAIN	RY_REPO_GUI			
1	2,588,590	90	63.0	0.0	Call M. LCL_APPLICATION=>INIT	SAPLREGUI			
1	2,415,126	216	58.8	0.0	Call M. LCL_TRANSACTION_PO=>OPEN	SAPLREGUI			
2	2,263,734	19,817	55.1	0.5	Call M. CL_PO_HEADER_HANDLE_MM=>PO_READ	SAPLREGUI			
1	1,722,607	54	41.9	0.0	Call Func. REPO_DOC_IN=>SIZE	CL_PO_HEADER_HANDLE_MM=====CP			
1	1,717,693	77	41.8	0.0	Call M. CL_PO_HEADER_HANDLE_MM=>PO_INITIALIZE	CL_PO_HEADER_HANDLE_MM=====CP			
1	1,717,433	12,523	41.8	0.3	Perform(Ext) REPO_REFRESH	SAPLRECOM1			
1	1,468,917	1,240,282	35.7	30.2	Perform REFRESH_TABLES	SAPLREPO			
1	1,175,956	53	28.6	0.0	Call M. CL_WINDOW_MM=>SEND	SAPLREGUI			
1	1,171,560	32	28.5	0.0	Perform(Ext) CALL_SCREEN	CL_WINDOW_MM=====CP			
1	1,171,528	148,827	28.5	3.6	Call Screen 0014	SAPLREGUI			
44	576,729	34,980	14.0	0.9	PBO Dynpro SAPLREGUI	SAPLREGUI	Sys.		
42	539,289	2,683	13.1	0.1	PBO Dynpro SAPLREVIEWS	SAPLREVIEWS	Sys.		
1	535,544	103	13.0	0.0	Dynpro Entry CALL_SCREEN	SAPLREGUI			
2	499,789	144	12.2	0.0	Call Func. REPO_DOC_READ	CL_PO_HEADER_HANDLE_MM=====CP			
2	490,838	48	11.9	0.0	Perform BESTELLUNG_LESEN	SAPLRECOM1			
2	481,791	18	11.7	0.0	Perform(Ext) BESTELLUNG_LESEN	SAPLRECOM1			
2	481,683	665	11.7	0.0	Perform LESEN_BELEG	SAPLREPO			
16	405,807	301	9.9	0.0	Call M. CL_WINDOW_MM=>HANDLE_EVENT	SAPLREGUI			
2	391,838	36	9.5	0.0	Module(PAI) FC00E_EXIT				
2	387,683	56	9.4	0.0	Call M. CL_COMPOSITE_SCREEN_VIEW_MM=>EXECUTE	CL_WINDOW_MM=====CP			
15	387,483	256	9.4	0.0	Call M. CL_COMPOSITE_SCREEN_VIEW_MM=>EXECUTE	CL_COMPOSITE_SCREEN_VIEW_MM=====CP			
1	386,770	141	9.4	0.0	Perform LESEN_LIEFERANT	SAPLREPO			
2	375,379	30	9.1	0.0	Call M. CL_CONTROLLER_MM=>EXECUTE	CL_COMPOSITE_SCREEN_VIEW_MM=====CP			
2	375,356	76	9.1	0.0	Call M. CL_COMMAND_PROCESSORS_MM=>IF_COMMAND_MM=>EXECUTE	CL_CONTROLLER_MM=====CP			
2	372,771	117	9.1	0.0	Call M. LCL_DOCUMENT_CMD=>IF_COMMAND_MM=>EXECUTE	CL_COMMAND_PROCESSORS_MM=====CP			
1	372,552	63	9.1	0.0	Call M. LCL_APPLICATION=>LEAVE	SAPLREGUI			
1	370,851	183	9.0	0.0	Call Func. VENDOR_MASTER_DATA_SELECT_00	SAPLREPO			
14	361,393	581	8.8	0.0	Call M. CL_BADI_FLT_DATA_TRANS_AND_DB=>ACT_INPS_PER_FLT_VAL	CL_EXITHANDLER=====CP			
1	360,404	23	8.8	0.0	Perform RETAIL_REFERENCE_SITE_SELECT	SAPLW06			
1	360,381	43	8.8	0.0	Call M. CL_EXITHANDLER=>GET_INSTANCE	SAPLW06			
5	360,129	106	8.8	0.0	Call Func. SXE_INPS_RELEASED_FOR_CUSTOMER	CL_BADI_FLT_DATA_TRANS_AND_DB=====CP			
1	359,940	260	8.8	0.0	Call Func. PA_GET_EXTENSION_ACTIVITY	SAPLSENE			
1	355,168	56	8.6	0.0	Call M. LCL_TRANSACTION_PO=>CLOSE	SAPLREGUI			
1	354,932	907	8.6	0.0	Call M. CL_PO_HEADER_HANDLE_MM=>PO_CLOSE	SAPLREGUI			
1	348,720	824	8.5	0.0	Call Func. PA_GET_INACTIVE_PACKAGES	SAPLPA_PACKAGE_SERVICES			
1	347,276	11,808	8.5	0.3	Call M. CL_PO_HEADER_HANDLE_MM=>REFRESH_RWIN	CL_PO_HEADER_HANDLE_MM=====CP			
1	339,411	5,436	8.3	0.1	Perform RECOMPUTE_INACTIVE_PACKAGES	SAPLPA_PACKAGE_SERVICES			
6	307,681	1,300	7.5	0.0	PBO Dynpro SAPLREACTVI	SAPLREACTVI	Sys.		
16	273,887	11,015	6.7	0.3	Call Func. PA_GET_DESCENDANT_PACKAGES	SAPLPA_PACKAGE_SERVICES			
82	265,106	265,106	6.5	6.5	Fetch TDEV	SAPLPA_PACKAGE_SERVICES	DB	OpenS	
1	235,499	1,294	5.7	0.0	Call Func. RWIN_CHECK	CL_PO_HEADER_HANDLE_MM=====CP			
1	234,205	67	5.7	0.0	Call Func. RWIN_CHECK_SUBSET	SAPLRWIN			
1	234,138	174	5.7	0.0	Perform FILL_IT_BUFFER	SAPLRWIN			
1	202,067	8	4.9	0.0	Call M. LCL_APPLICATION=>INIT_FIRST_PLUGIN	SAPLREGUI			
3	200,067	123	4.9	0.0	Call M. LCL_PLUGIN_MGR=>GET_PLUGIN	SAPLREGUI			
1	199,876	288	4.9	0.0	Call M. LCL_PLUGIN_PO=>CONSTRUCTOR	SAPLREGUI			
2	192,230	6,848	4.7	0.2	Call Func. VB_INIT	SAPLREPO			
1	187,347	29	4.1	0.0	Call M. LCL_APPLICATION=>RESET_VIEWS	SAPLREGUI			
1	187,096	27	4.1	0.0	Call M. LCL_DOCUMENT_VIEW=>IF_MODEL_HOLDER_MM=>SET_MODEL	SAPLREGUI			
1	187,042	21	4.1	0.0	Raise E. LCL_DOCUMENT_VIEW=>IF_SUBJECT_MM=>CHANGED	SAPLREGUI			
89	187,021	426	4.1	0.0	Call M. CL_SCREEN_VIEW_MM=>IF_OBSERVER_MM=>HANDLE_SUBJECT_CHANGED	SAPLREGUI			
27	164,783	167	4.0	0.0	Call M. CL_SCREEN_VIEW_MM=>HANDLE_EVENT	CL_MODEL_VIEW_MM=====CP			
1	149,189	68,400	3.6	1.7	Call Func. CLAP_DOB_INIT_CLASSIFICATION	SAPLV01Z			
8	147,882	139	3.8	0.0	Call M. CL_BASIC_MODEL_VIEW_MM=>PBO	CL_SCREEN_VIEW_MM=====CP			
2	144,879	91	3.5	0.0	Call M. LCL_TRANSACTION_PO=>GET_GOS_MANAGER	SAPLREGUI			
2	144,725	306	3.5	0.0	Call M. CL_GOS_MANAGER=>CONSTRUCTOR	SAPLREGUI			
1	135,054	21	3.3	0.0	Call M. LCL_PO_ITEM_TABLE_VIEW=>SUBJECT_CHANGED	CL_SCREEN_VIEW_MM=====CP			
1	134,888	17	3.3	0.0	Call M. LCL_PO_ITEM_TABLE_VIEW=>SET_MODEL	SAPLREGUI			

Not being a developer I only look for a few things in this list such as Fetch and Select statements as they generally relate to table reads and writes and these are all of the type  . (I'm sure an ABAP savvy person might have a better criteria) you can filter the results using the ALV filter button.

Select the type column for filtering



then enter your criteria of **DB**

Runtime Analysis Evaluation: Hit List

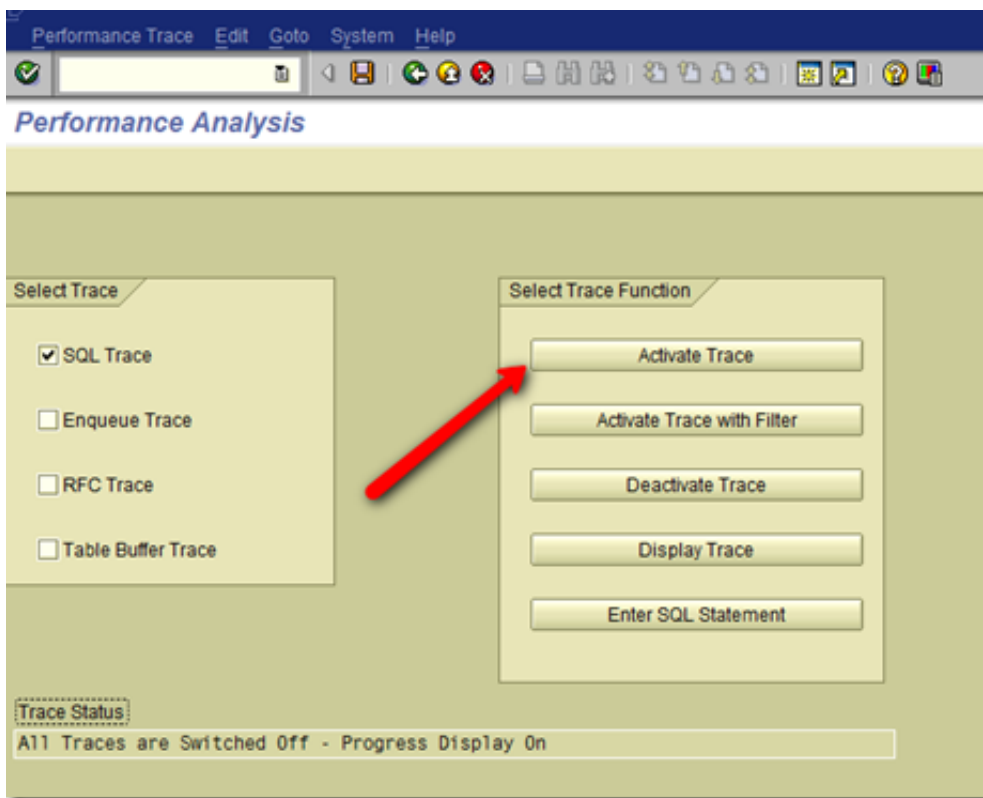
No.	Gross	Net	Gross (%)	Net (%)	Call	Program Name	Type	No. Filter
82	265,106	= 265,106	0.5	0.5	Fetch TDEV	SAPLPA_PACKAGE_SERVICES	DB	OpenS
1	118,601	= 118,601	2.9	2.9	Open Cursor TRMPR	SAPLRWIN	DB	OpenS
1	35,241	= 35,241	0.9	0.9	Delete PAKDATBUF	SAPLPA_PACKAGE_SERVICES	DB	OpenS
1	19,596	= 19,596	0.5	0.5	Select Single T006B	SAPLSCVU	DB	OpenS
1	18,082	= 18,082	0.4	0.4	Select Single T7880	SAPLRHMR	DB	OpenS
1	16,301	= 16,301	0.4	0.4	Open Cursor TRWCA	SAPLRWIN	DB	OpenS
1	11,955	= 11,955	0.3	0.3	Open Cursor TRWCD	SAPLRWIN	DB	OpenS
3	10,483	= 10,483	0.3	0.3	Select Single TPAER	SAPLEKPA	DB	OpenS
1	8,318	= 8,318	0.2	0.2	Commit Work	SAPLPA_PACKAGE_SERVICES	DB	OpenS
2	6,615	= 6,615	0.2	0.2	Modify ESDUS	SAPMLSO	DB	OpenS
1	6,302	= 6,302	0.2	0.2	Open Cursor TRWCI	SAPLRWIN	DB	OpenS
3	5,713	= 5,713	0.1	0.1	Select Single T163K	SAPLMEPO	DB	OpenS
1	4,974	= 4,974	0.1	0.1	Select Single T001W_EXT	SAPLAIPLOC01	DB	OpenS
82	4,815	= 4,815	0.1	0.1	Open Cursor TDEV	SAPLPA_PACKAGE_SERVICES	DB	OpenS
2	4,546	= 4,546	0.1	0.1	Open Cursor TPRG	SAPLMEPO	DB	OpenS
4	4,490	= 4,490	0.1	0.1	Select Single T162	SAPLMEXF	DB	OpenS
1	4,284	= 4,284	0.1	0.1	Select Single T160	RM_MEPO_GUI	DB	OpenS
5	4,254	= 4,254	0.1	0.1	Open Cursor DOFTX	SAPLSOIFRUNTIME	DB	OpenS
1	4,097	= 4,097	0.1	0.1	Select Single V_ADDR_USR	SAPLSUSM	DB	OpenS
1	3,731	= 3,731	0.1	0.1	Select Single T006A	SAPLSCVU	DB	OpenS
2	3,322	= 3,322	0.1	0.1	Fetch T16FB	SAPLMEREL	DB	OpenS
1	3,271	= 3,271	0.1	0.1	Select Single T001W	SAPLAIPLOC03	DB	OpenS
1	3,063	= 3,063	0.1	0.1	Select Single T161T	SAPLMEXF	DB	OpenS
1	2,916	= 2,916	0.1	0.1	Fetch NAST	SAPLVMSG	DB	OpenS
1	2,817	= 2,817	0.1	0.1	Open Cursor T163Y	SAPLEINH	DB	OpenS
1	2,708	= 2,708	0.1	0.1	Select Single T161	SAPLMEXF	DB	OpenS
1	2,642	= 2,642	0.1	0.1	Fetch PAKDATBUF	SAPLPA_PACKAGE_SERVICES	DB	OpenS
1	2,540	= 2,540	0.1	0.1	Select Single LFA1	SAPLWY01	DB	OpenS
2	2,464	= 2,464	0.1	0.1	Fetch EKPO	SAPLMEPO	DB	OpenS
1	2,459	= 2,459	0.1	0.1	Select Single EKK0	SAPLME01	DB	OpenS
4	2,266	= 2,266	0.1	0.1	Select Single USR05	CL_USER_SETTINGS_MM=====CP	DB	OpenS
1	2,265	= 2,265	0.1	0.1	Array Insert PAKDATBUF	SAPLPA_PACKAGE_SERVICES	DB	OpenS
1	1,977	= 1,977	0.0	0.0	Select Single T163	SAPLMEPROP	DB	OpenS
3	1,898	= 1,898	0.0	0.0	Select Single T162	SAPLMEPO	DB	OpenS
1	1,840	= 1,840	0.0	0.0	Open Cursor T161M	SAPLMEPO	DB	OpenS
1	1,825	= 1,825	0.0	0.0	Select Single USR02	SAPLSUU1	DB	OpenS
1	1,819	= 1,819	0.0	0.0	Fetch KONV	SAPLV61A	DB	OpenS
1	1,809	= 1,809	0.0	0.0	Select Single MMPURCI_DETAILS	SAPLMEPO	DB	OpenS
1	1,748	= 1,748	0.0	0.0	Open Cursor EKPO	SAPLMEPO	DB	OpenS
1	1,730	= 1,730	0.0	0.0	Open Cursor TC08M	SAPLMECO	DB	OpenS
49	1,689	= 1,689	0.0	0.0	Fetch ESDUS	SAPMLSO	DB	OpenS
2	1,687	= 1,687	0.0	0.0	Fetch STXH	SAPLSTXD	DB	OpenS
1	1,660	= 1,660	0.0	0.0	Open Cursor T16FG	SAPLEBNF	DB	OpenS
1	1,659	= 1,659	0.0	0.0	Select Single TMCNV	SAPLONCV	DB	OpenS
1	1,659	= 1,659	0.0	0.0	Select Single T16CA	SAPLMEDCHM	DB	OpenS
2	1,655	= 1,655	0.0	0.0	Select Single V_USEREXST	RHUSER00	DB	OpenS
1	1,620	= 1,620	0.0	0.0	Select Single T024	SAPLMEXF	DB	OpenS
1	1,615	= 1,615	0.0	0.0	Select Single T024E	SAPLMEXF	DB	OpenS
5	1,589	= 1,589	0.0	0.0	Fetch DOFTX	SAPLSOIFRUNTIME	DB	OpenS
1	1,585	= 1,585	0.0	0.0	Open Cursor T024Z	SAPLMEQR	DB	OpenS
1	1,553	= 1,553	0.0	0.0	Open Cursor PACT	SAPLPRIO_HANDLER	DB	OpenS
1	1,512	= 1,512	0.0	0.0	Open Cursor SWOTLI	SAPLSWOR	DB	OpenS
2	1,511	= 1,511	0.0	0.0	Open Cursor STXH	SAPLSTXD	DB	OpenS
2	1,504	= 1,504	0.0	0.0	Select Single T160	SAPFMEX	DB	OpenS
1	1,496	= 1,496	0.0	0.0	Select Single THMPURCFOLDERS	SAPLMEPO	DB	OpenS
1	1,472	= 1,472	0.0	0.0	Select Single USR21	SAPLSUSM	DB	OpenS
1	1,455	= 1,455	0.0	0.0	Open Cursor TWLAD	SAPLMMDA	DB	OpenS
1	1,393	= 1,393	0.0	0.0	Select Single WMPARAMS	SAPLSHTM	DB	OpenS
1	1,263	= 1,263	0.0	0.0	Open Cursor TC08F	SAPLMECO	DB	OpenS
2	1,257	= 1,257	0.0	0.0	Select Single T160V	SAPLMEPO	DB	OpenS
1	1,180	= 1,180	0.0	0.0	Select Single USREFUS	SAPLSUU1	DB	OpenS
2	1,144	= 1,144	0.0	0.0	Fetch V_USERNAME	SAPLRHUS	DB	OpenS
1	1,138	= 1,138	0.0	0.0	Open Cursor LFM2	SAPLWY06	DB	OpenS
1	1,112	= 1,112	0.0	0.0	Fetch EKBE	SAPLME07	DB	OpenS
2	1,086	= 1,086	0.0	0.0	Fetch EKKN	SAPLMEPO	DB	OpenS

you now have a pretty exhaustive list of tables that were hit during the trace (so make sure you populate values in the fields you're trying to track down when doing the trace.)

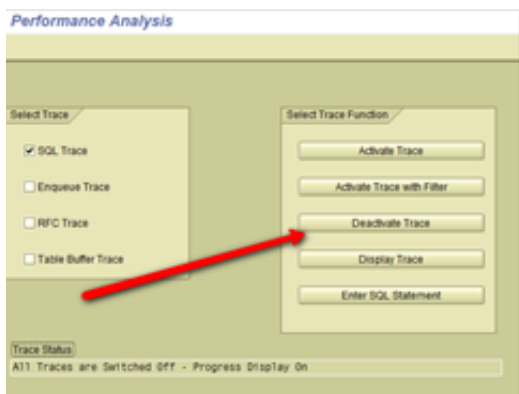
ST05 SQL Trace

This is a similar form of trace to the runtime analysis, however it works using two sessions, it also generates a lot of data so you need to prep what you trace before hand.

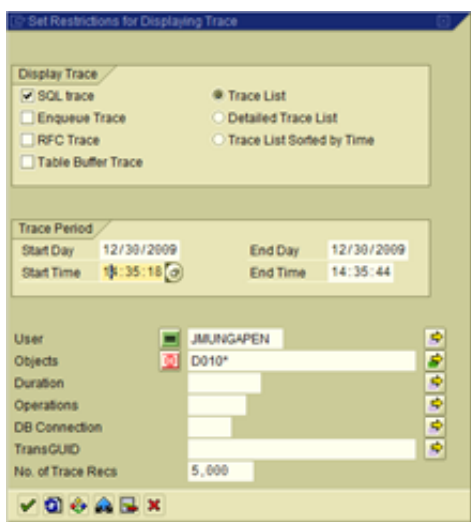
in one session set up the transaction you want to trace, then open a new session and select ST05



Select the SQL trace and activate. Alt Tab to your other session, enter some data in the area you want to trace then save that transaction. Immediately alt tab back to the trace and deactivate it



now you can display the trace



somewhere in the resulting mess of a report will be the data you changes and the resulting table objects

