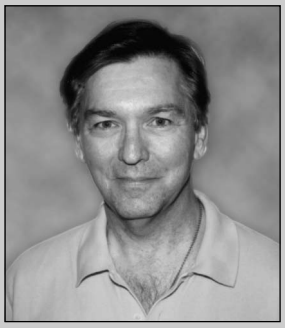
18 Techniques for Locating the Underlying Data of a Screen Field

Dennis Barrett



Dennis Barrett is an applications consultant with SAP America who focuses on Service Management/ Customer Service and the Service Provider solution. He has been consulting in computers for over 15 years, always blending applications and programming. He is a certified ABAP programmer, and is also the author of "SAP R/3 ABAP/4 Command Reference."

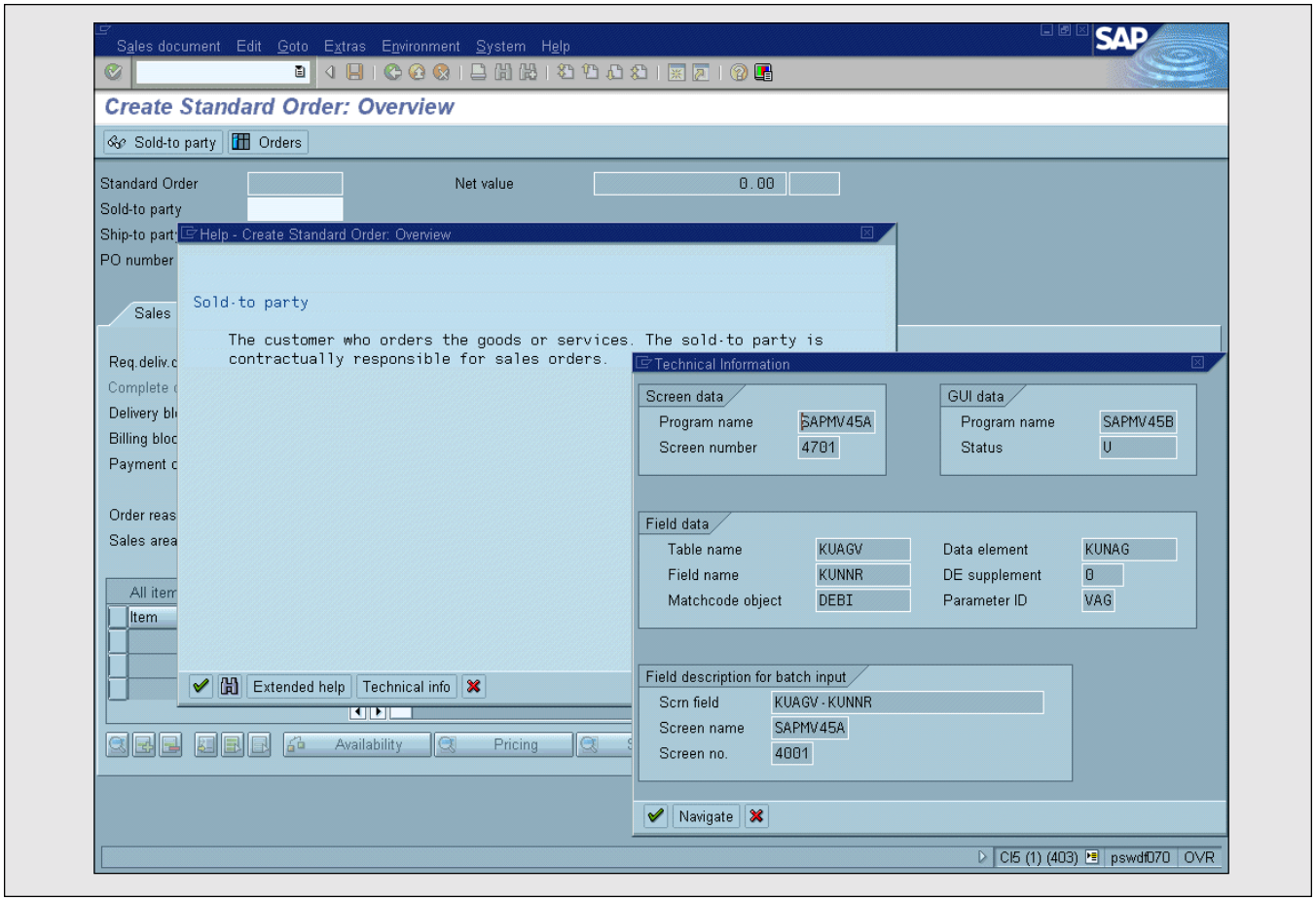
(complete bio appears on page 18)

Picture this. An Operations Manager wants a special report of her service orders. She gives you a sketch of what she wants it to look like (columns, rows, headings, grouping, subtotals, totals, and the like), and points to fields on the Service Management transaction screens that have the data she wants you to show in the report.

What do you do? Most likely, you would check to find any existing reports delivered with the system (or already written for this client) that provide the information the Operations Manager wants, or that can be copied and revised to fit her needs. If you don't find any, you might then look into the appropriate reporting system — in this case, the Plant Maintenance Information System (PMIS) — to see if you can adapt it. In this case, however, you can't find the report, and you can't cobble one together from existing ones. You must write a report or an ABAP Query to fulfill the requirements.

Now, suppose the Operations Manager asks if you can somehow add just one more little function to her Create Measurement Documents transaction: paste the associated sales order number into the MDoc text field. This scenario requires you to create an enhancement. In both the first and second scenario, you'll need access to specific data from the database tables. How do you find that data with just the screen fields as your guide? We all know the information associated with a transaction is stored in several (sometimes many) related tables. So, when you need several fields from a transaction for a report or an enhancement, you may need to find many of the transaction's tables and establish the links among them. Those links are often not obvious.

Figure 1 The Release 4.0B Technical Information Screen for the "Sold-to party" Field



I've come across these very situations on numerous occasions while working for several clients with the SD, MM, IM, WM, and Service Management (now called Customer Service) modules. Over time I have gathered suggestions from colleagues and developed some techniques myself to find the data I needed. I will share these techniques — 18 of them in all — with you now.

The Starting Gate and Finish Line

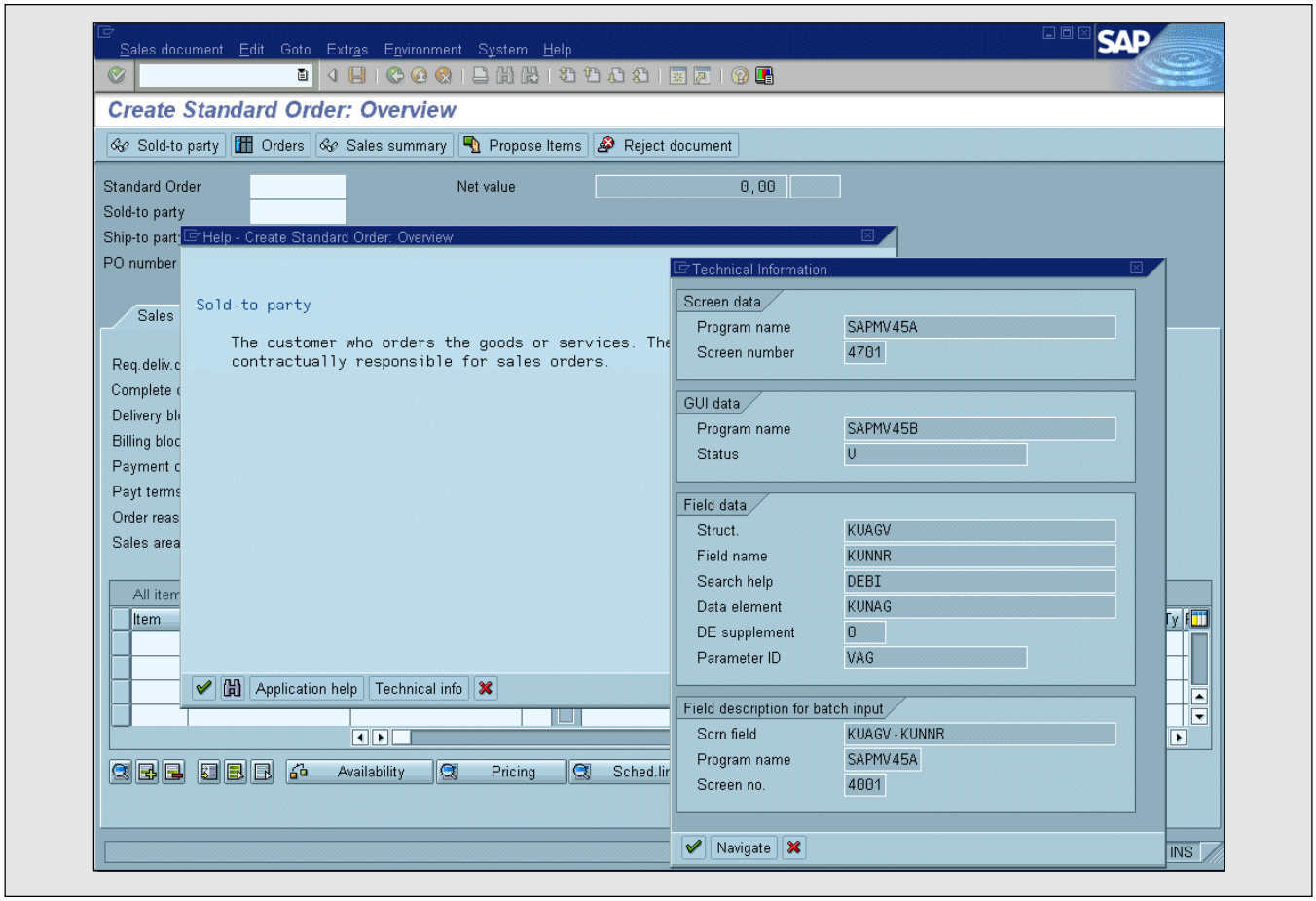
I will be presenting a wide variety of techniques for reaching the "finish line" of this exercise — namely, locating the table and field that stores the underlying data of a screen field. You're unlikely to need all of them; you'll probably be successful within the first four techniques. I include the remaining techniques because some fields can be very hard to chase down, and any one of these other tools just might do the trick.

The Starting Point

The starting point for this set of techniques is always the Technical Information screen, because it is here that you learn the screen field's name, and can discern whether or not the underlying data object is a structure, view, or table.

If the underlying data object is a table, your search is over. The field name shown on the Technical Information screen, plus this table name,

Figure 2 The Release 4.5B Technical Information Screen for the "Sold-to party" Field



arm you with the details you need. If the data object is a view, just one more click and you've got that table name. If the data object is a structure, you still have to unearth the name of the table.

Technique #1: Check the Technical Information Screen

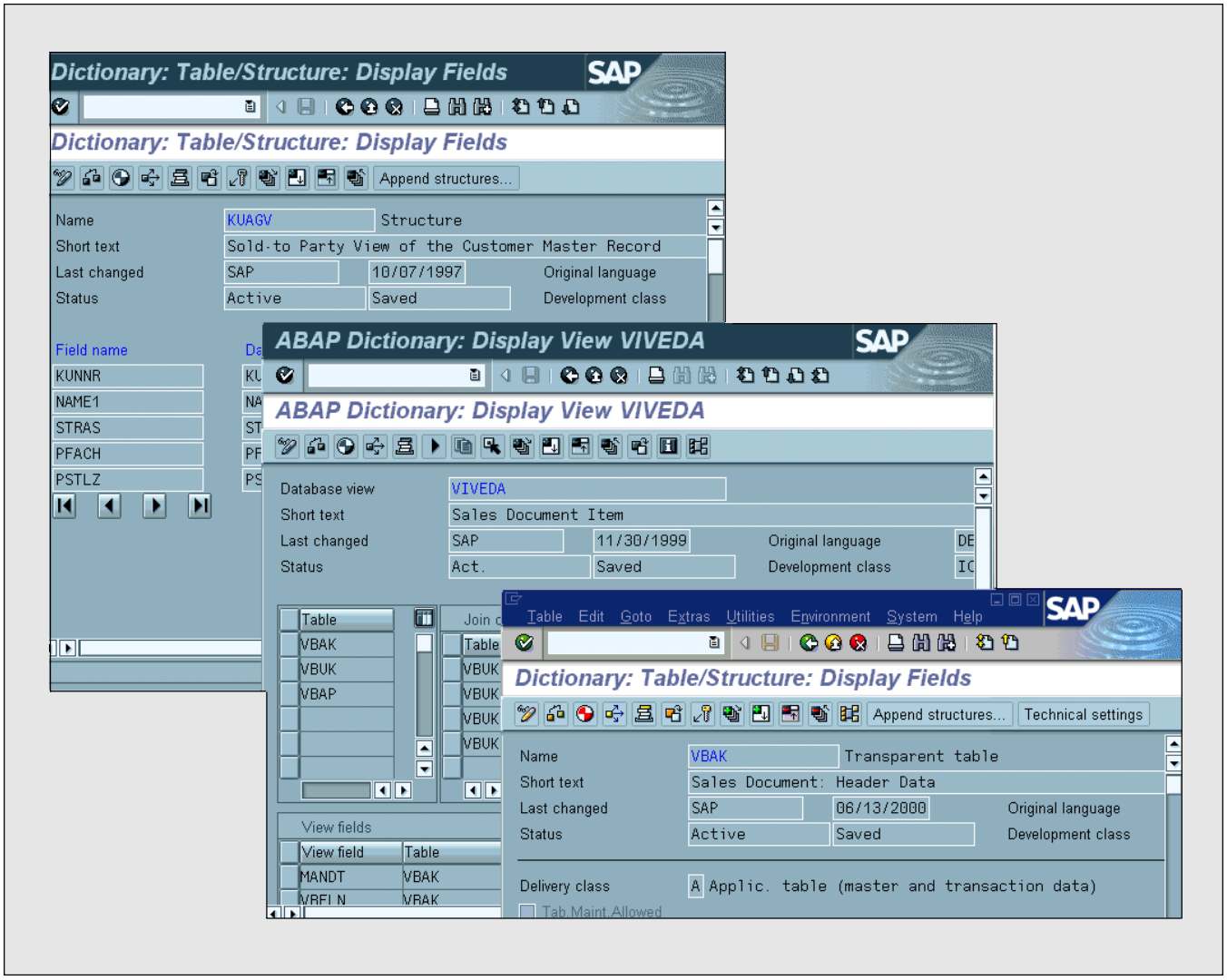
I always start my search for the underlying data object by getting the names of the field and Data Dictionary object from the transaction screen. Call up the Technical Information screen by positioning your cursor in the target screen field and then pressing **F1** or clicking on the **?** button. This will bring up the initial Help screen. From within the Help screen, press **F9** or click on the **Technical info** button.

Take a look now at **Figure 1**. It shows the Technical Information screen for the **Sold-to party** field in the Create Standard (Sales) Order screen for Release 4.0B. You see in the "Field data" block that the **Table name** is KUAGV and the **Field name** is KUNNR.[1] In Releases 4.0B and below, this screen calls the associated object a "table," whether it's a table, view, or structure. Notice in **Figure 2** that the 4.5B release (and above) informs you that KUAGV is actually a structure.

---

[1] The examples herein mention names of tables, structures, and fields that I found in my searches. You may find a different name in the same search — particularly a different structure's name. For example, the **Sold-to party** field in the Sales Order initial screen may be KUAGV-KUNNR or RV45S-KUNNR (or maybe something else). I'll show one name in this article; don't worry if you find another. It's a reflection of the continual enhancement of R/3, not (I hope) an error in the article.

Figure 3 *The Release 4.0B Data Dictionary Screens for Table "VBAK," View "VIVEDA," and Structure"KUAGV"*



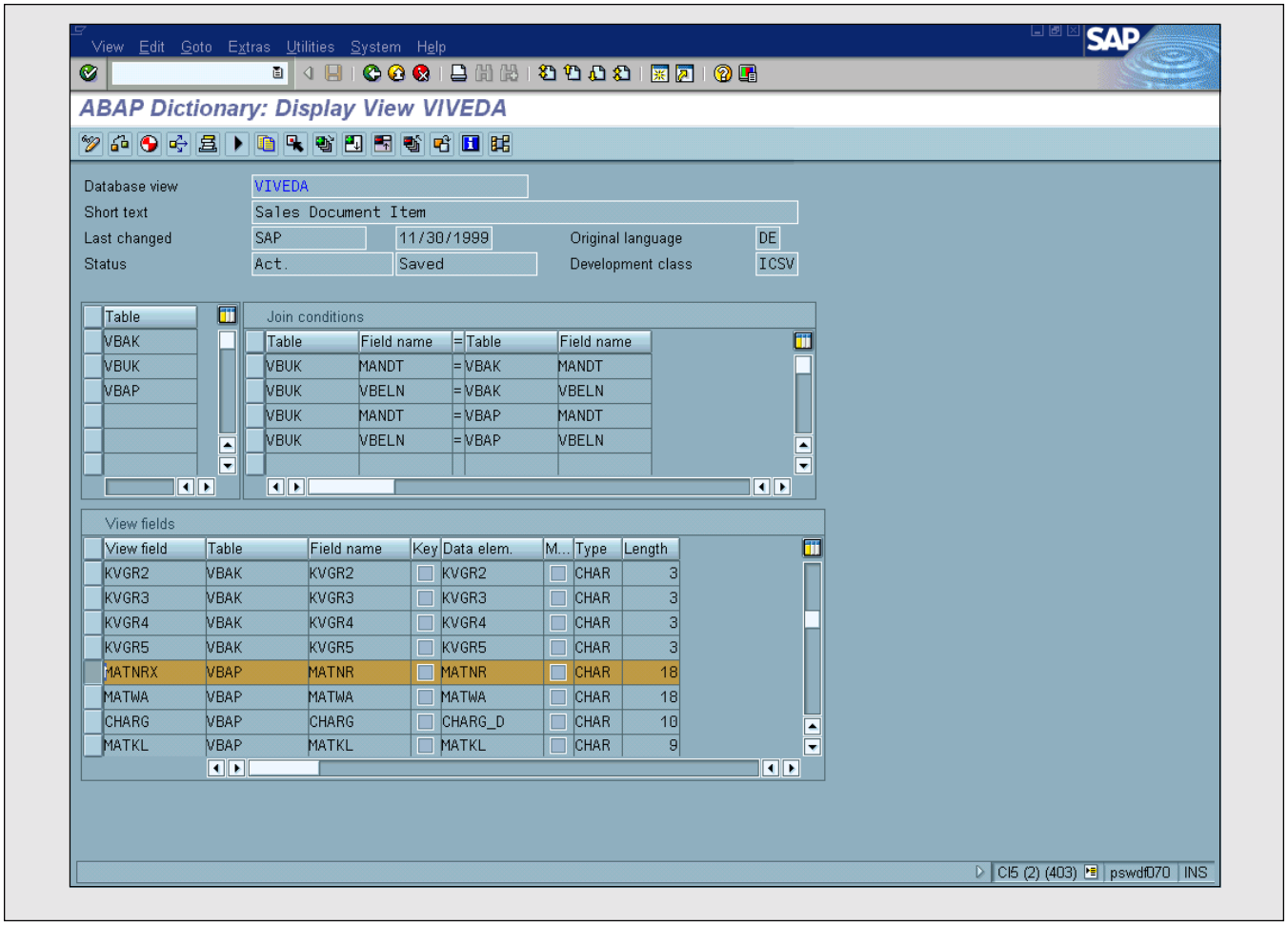
In Releases 4.0B and below, to ascertain whether or not an object is a structure you must take one more step. Drill into the **Table name** field in the Technical Information screen to see the Data Dictionary screen for the object.

Figure 3 shows those screens for the table VBAK, the view VIVEDA, and the structure KUAGV. Corresponding screens in Releases 4.5 and 4.6 look very similar. As you can see, the object type is easy to recognize in these screens.

If the underlying object is a table, you're home free. The information is stored in that table in the field with the same name as the one shown on the Technical Information screen. Success! When you write your report or ABAP Query, or when you program your enhancement, you can refer to the data using these table and field names.

If the object is a view, then scroll to the **View field** in the view with the same name as the one shown in the Technical Information screen. See **Figure 4**; the **Table** and **Field name** associated with the **View field** is your goal. Success again!

Figure 4 Locating the "Table" and "Field name" Associated with the "View field"



Tables, Views, and Structures

A screen field is associated with a table, view, or structure. While each of these is a Data Dictionary object, only the table actually stores data. This is why the goal of all these techniques is to find the table that actually stores the information that you can see displayed in the screen.

Table

Tables store data. If your screen field points to a table, or you are able to drill down to a table from a view, then you have found the storage location of your data, and you've reached the finish line! This table name and field name are what you need for your report or query.

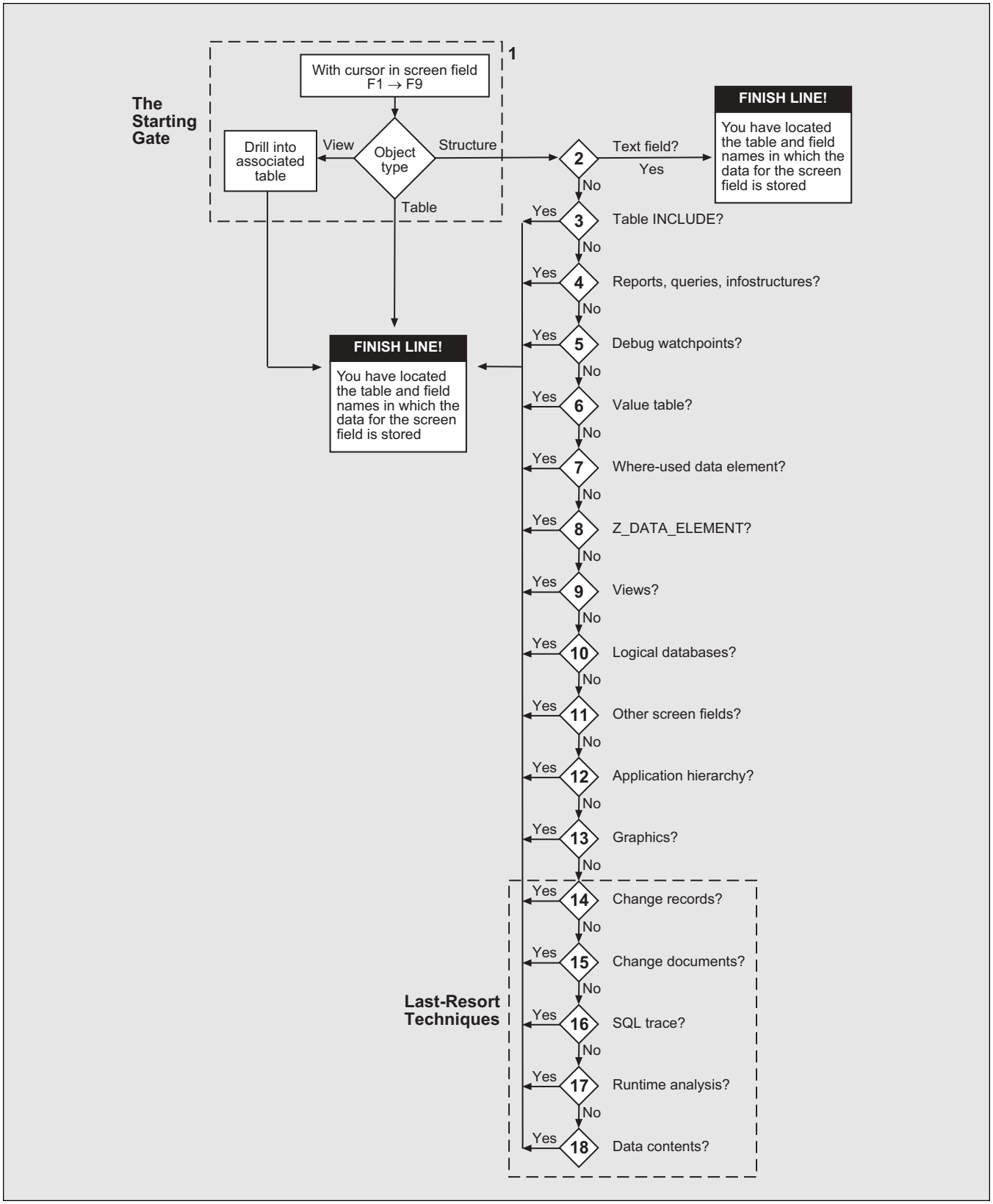
View

Views used in screens are linked groups of tables. If the field is in a view, you can drill down to the underlying table that contains the data.

Structure

Structures are Data Dictionary objects that have fields but do not carry data. A structure is simply an array of variables defined in the Data Dictionary, and are like empty tables. If the screen field is in a structure, you'll have to keep looking. Remember, only a table actually stores data.

Figure 5 The Suggested Sequence of Techniques



If the object is a structure, then you have more digging to do, and you must proceed to one or more of the remaining 17 techniques described here. These techiques follow an order that should get you to the finish line in the fewest number of steps. Remember, as soon as you find the associated table, you're done with the search. Look at the flowchart in **Figure 5** to see my suggested sequence of techniques.

Racing Toward the Finish Line: Techniques 2-13

Remember, the finish line, or goal, of this exercise is to locate the table and field names in which the underlying data for a screen field is stored. At this point, we know only that the underlying data object is a structure. Techniques 2-13, which I detail in this section, will most likely hold the answer. If techniques 2-13 do not hold the answer, try techniques 14-18 — the measures of last resort! I suggest you hold off trying these last five tactics until you've exhausted the other possibilities. If none of these 18 techniques work, you have an unusually difficult problem; I wish you good luck. I hope you find your table quickly and easily.

Technique #2: Text Fields

If the screen label is "Short text", or the field name is STTXT, or its data element is CO_STTXT, then you are looking for the source of a text field. As **Figure 6**

Figure 6 Click the "Maintain long text" Icon to Get to the Long Text Screen

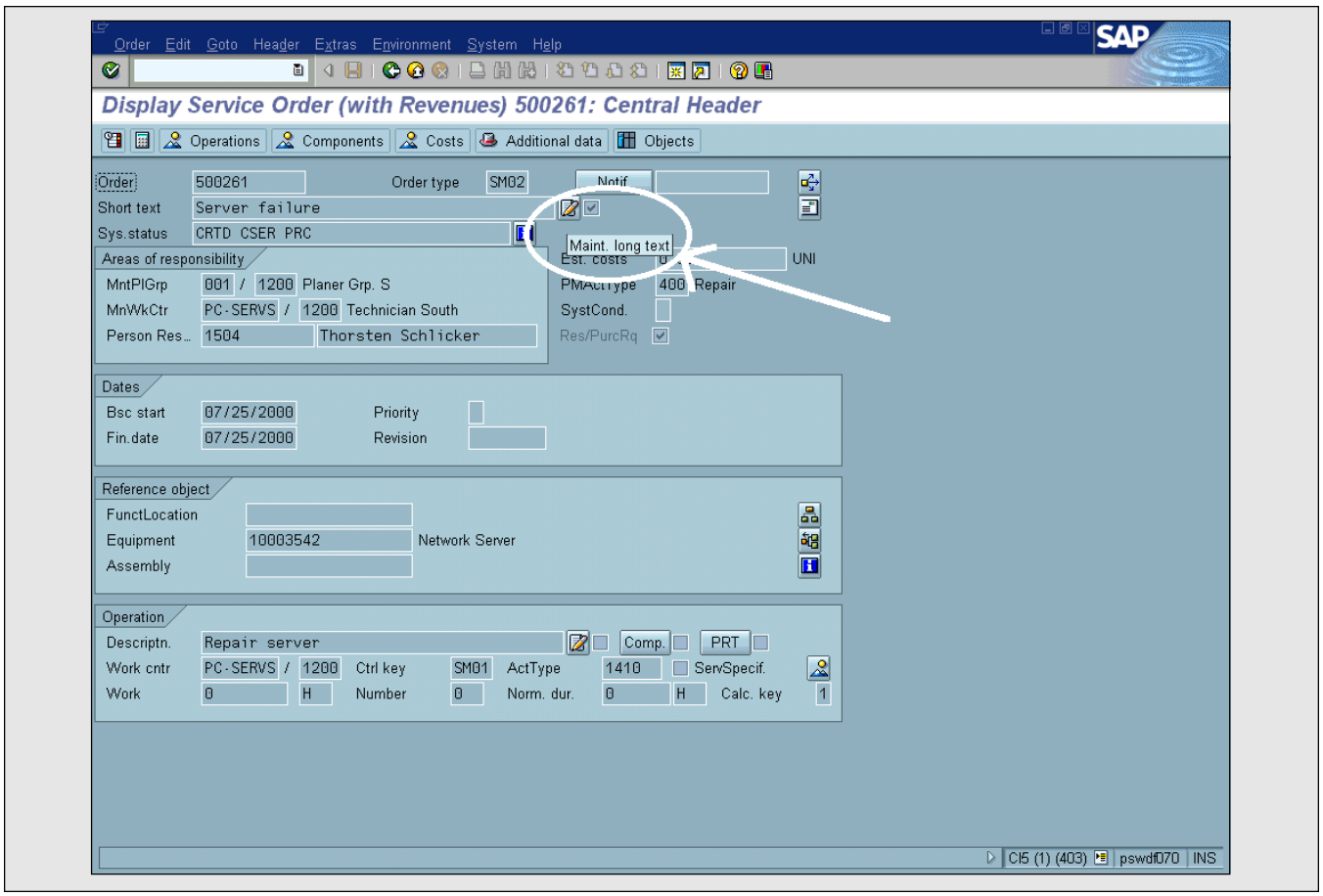
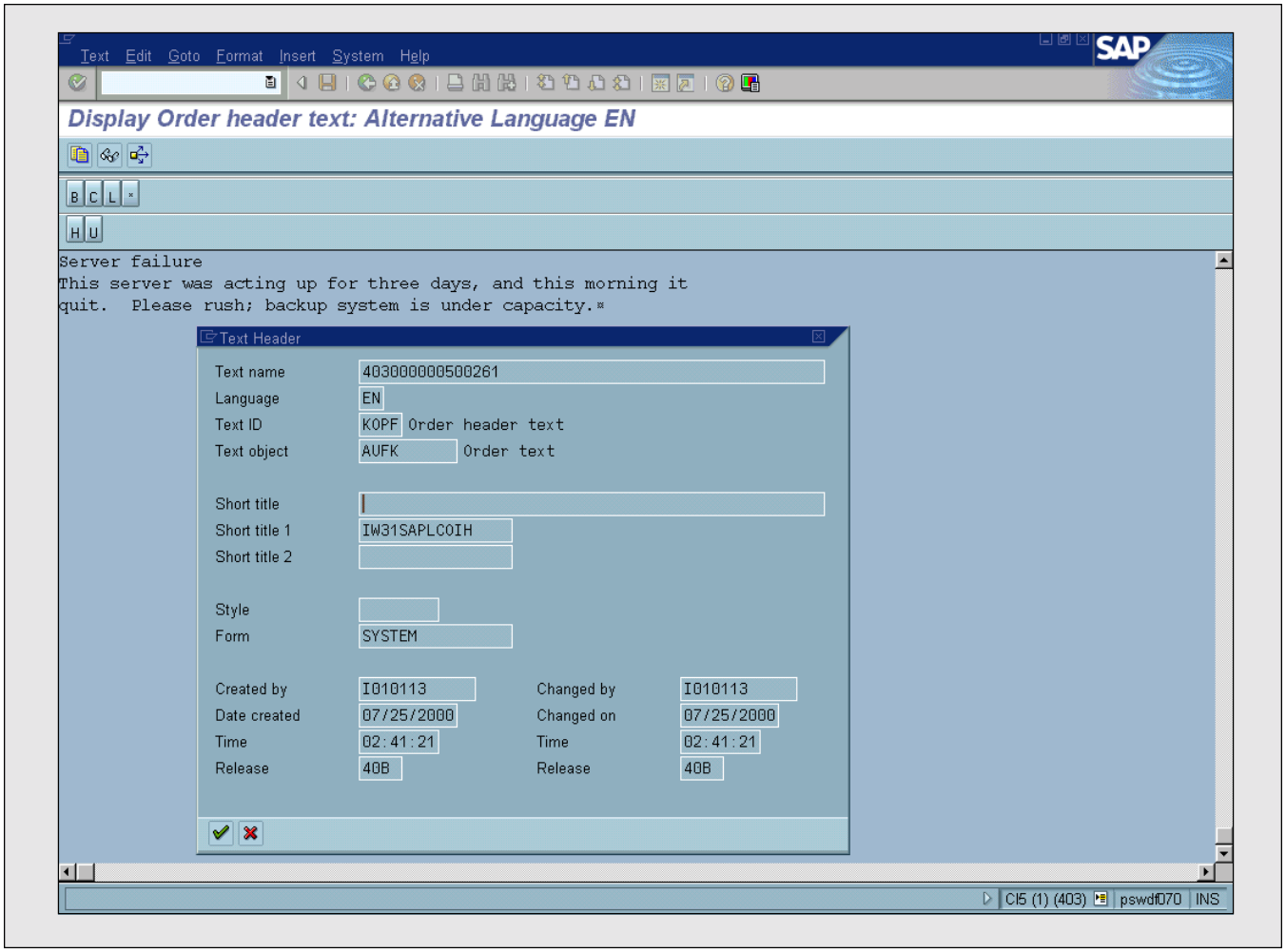


Figure 7 Retrieving the Text Name, Text ID, and Text Object Parameters



shows, the **Short text** field often has a "Maintain long text" icon stuck at the end that may look like a sheet of paper with a pencil. Click on that icon to get to the Long text screen.

Texts are stored all over the place in R/3, but frequently in tables STXH and STXL, which are the text header and text line, respectively. Their texts are stored in raw (i.e., non-ASCII) form, so you must use function modules to get them. There are several function modules for processing these texts, including Read_Text, which reads text lines from the database. In the Long text screen, follow **GoTo → Header** (see **Figure 7**) to get the **Text name**, **Text ID**, and **Text object** parameters that you'll need to execute the function module.

If the field is not a text field, then you must continue to search.

Technique #3: "INCLUDE" Table

A structure may contain one or more INCLUDEs. An INCLUDE may actually be a table or another structure. So, if the structure you are working with includes an INCLUDE, and that INCLUDE *is* a table that contains your field, then you have won again. For example, in a Sales Order Item Overview, the Condition Type is listed as KOMV-KSCHL, but KOMV is a structure. KOMV does contain the INCLUDE KONV, which *is* a table that contains the field KSCHL. That table and field are your goal, and

we need not go any further. If the structure doesn't contain an INCLUDE, or if the INCLUDE is not a table, then we must keep searching. The next technique you should apply is described in technique 4 because it's easy and, quite likely, will get you your answer.

Technique #4: Reports, Queries, Infostructures

If you know of any ABAP Queries, already-existing reports, or Infostructures that use the field you need, then you can dig into them to find the source table. Before you started this search, you may have found a report or query that didn't provide the result that the Operations Manager needs, but that uses the field you are looking for. Now that you have the field name from the Technical Information screen, you can review those reports and queries to see if the search has already been done for you. If you find your field in one of those reports or queries, you can examine the code and find the table that it uses to pull that field; you will have successfully concluded your search.

If you don't find your table and field here, then you must continue searching. Technique 5 is a powerful one for those who know how to use it, and can very often find the result.

You may have found a report or query that uses the field you are looking for. Now that you have the field name, you can review those reports and queries to see if the search has already been done for you. If you find your field in one of those reports or queries, you can examine the code and find the table that it uses to pull that field, which successfully concludes your search.

Technique #5: Debug Watchpoint

If you are searching in Release 4.0B or above, and want to go to the heavy artillery immediately, then get the **Program name** (under "Screen data") from the Technical Information dialog box and debug the program. Go to **Tools → ABAP Workbench → ABAP Editor → Program=[the program name] → Debugging**. Click on the **Watchpoint** button on the task bar and enter the program and field names into the **Local watchpoint** fields, then click on **Enter** and **F8**. Any change in the variable triggers the watchpoint, presenting you with the code immediately below the line that changed the variable. Read the code above the watchpoint carefully, and you can probably find the name of the Data Dictionary object that the field is in. Again, it may be a table, view, or structure, but you are getting very close.

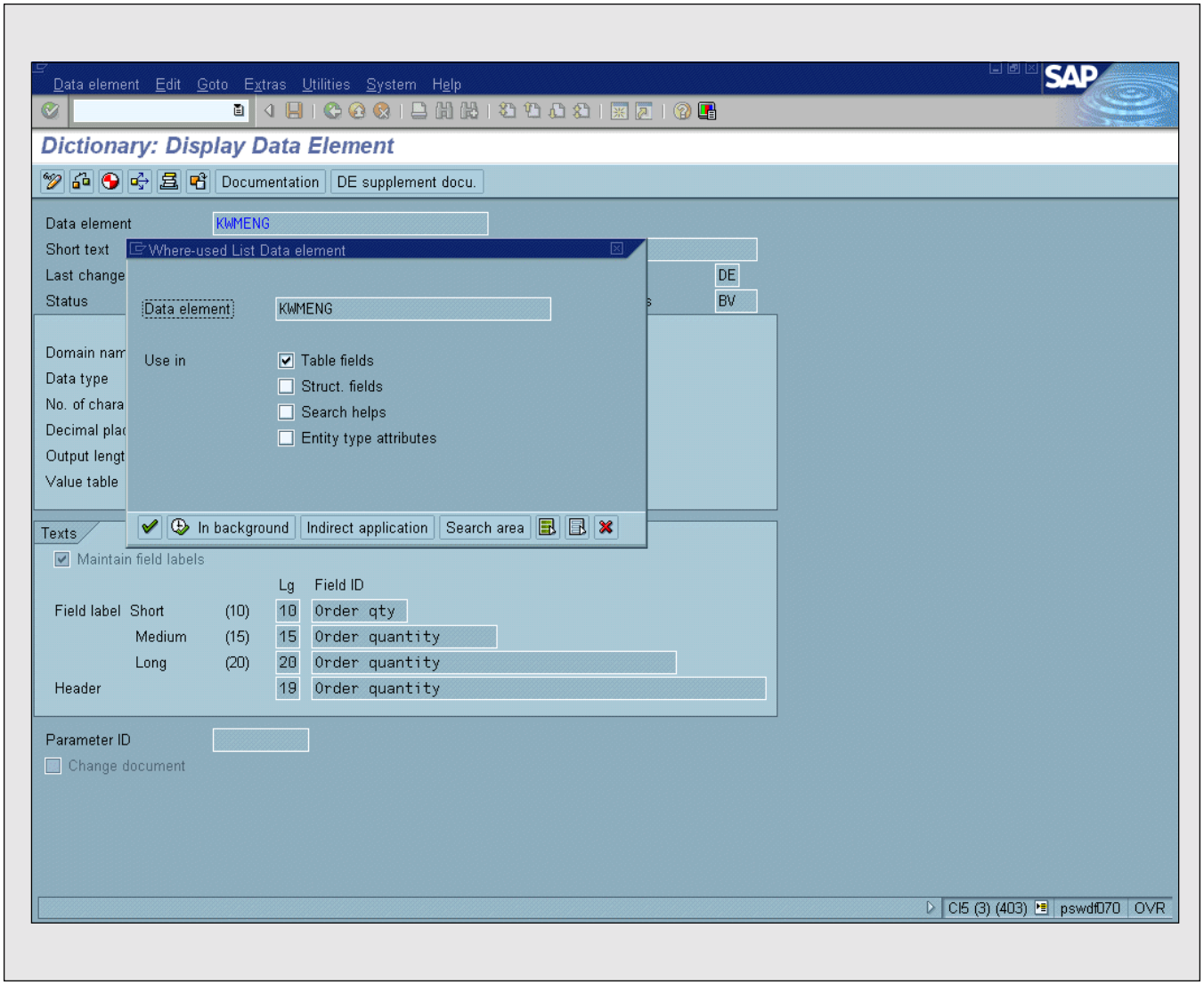
Ken Greenwood (author of *Teach Yourself ABAP/4 in 21 Days*) says he always jumps immediately to this technique if the Technical Information screen doesn't produce the table. Ken further suggests that this technique finds the table right away — with the only exception being the IMPORT statement. If the watchpoint doesn't trigger, then debug again using a breakpoint on IMPORT and you will find it in a maximum of three steps.

If you can't find the table reference you need in the code, then you must continue your search, and the next natural technique for you to use is described in technique 6.

Technique #6: Value Table

If debugging didn't give you what you need or you don't want to debug a program, then back out to the Technical Information dialog box and double-click on the **Data element** field under "Field data". This will show you the value table, if there is one. If you're just looking for master data like customer name — whose field name is KUNNR — then a value table will probably be defined, and it will give you what you want. You'll find that transaction data like Order Quantity — whose field name is KWMENG — has no value table, and you get to keep looking. From here on, techniques 7-13 all have about the same likelihood of revealing your table name. Use them in any order you like. You'll

Figure 8 "Where-used List Data element" Request Screen in Release 4.0B



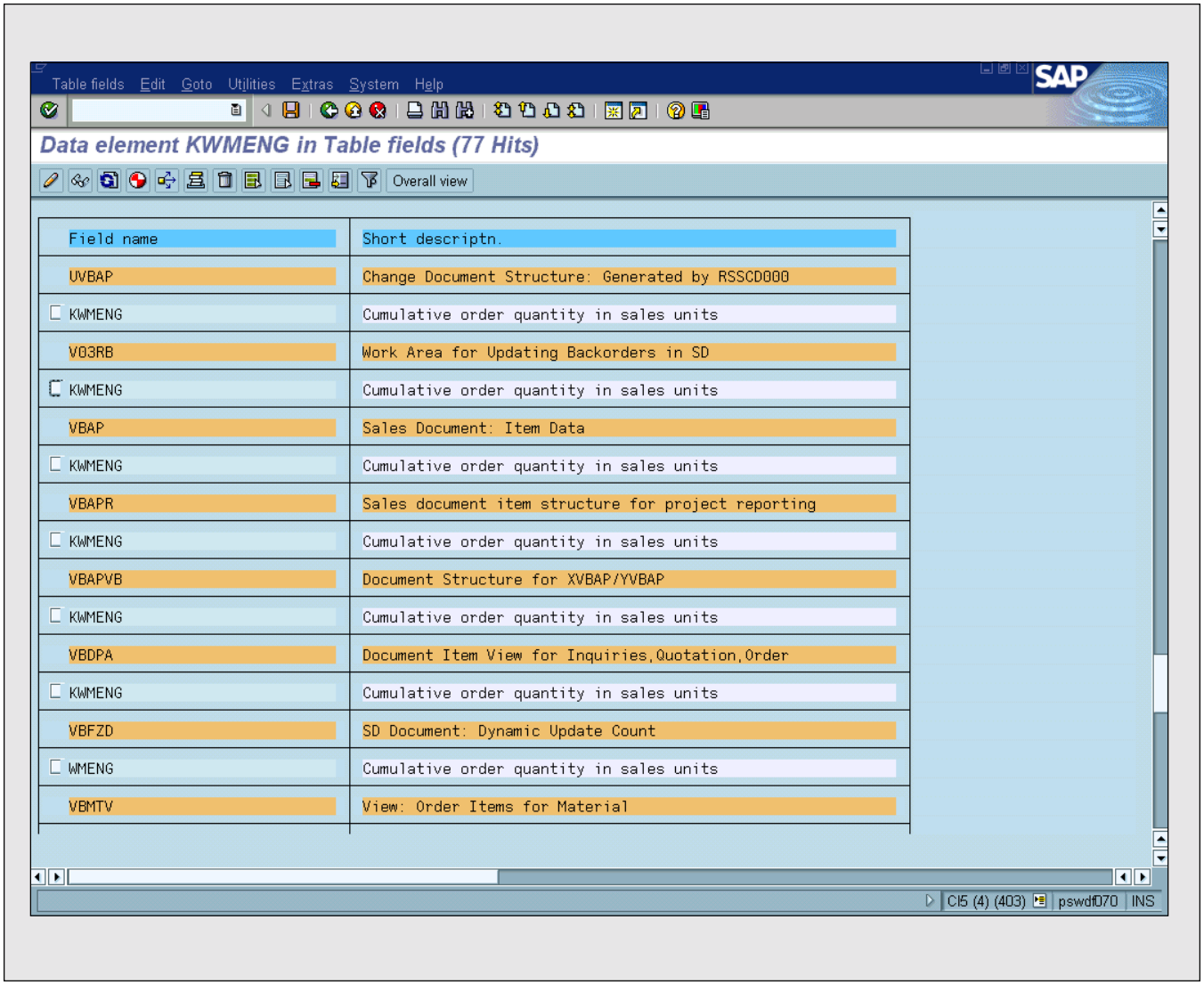
probably find that you like some more than others, so use those first.

> *If debugging didn't give you what you need or you don't want to debug a program, then back out to the Techical Information dialog box and double-click on the Data element field under "Field data". This will show you the value table, if there is one.*

Technique #7: "Where-Used" Data Element

When R/3 moves data from one table (or structure) to another, it often (not always) moves it into a target field having the same data element as the source field. From the Technical Information dialog box, double-click on the **Data element** field under "Field data" and follow **Utilities** → **Where-used List** → **Table fields** → **Enter** (see **Figure 8**). This will list the tables, views, and structures that use this data element. If your field is a commonly used

Figure 9 *Data Element "Where-used" List*



one, this list may be so large that it's hard to find your data storage location. For example, KUNNR shows up in 743 objects in Release 4.0B. An infrequently used field may show up in fewer objects, and you can find your data source in the list.

Note also that the list includes structures; sometimes, lots of structures. When you drill into a field name in the list, you'll see the table display that will identify it as a structure or a transparent table. If it's a structure, you'll have to keep looking.

Technique #8: "Z_DATA_ELEMENT"

Since the "Where-used" list for a data element generally contains so many (non-data carrying) structures, the list is not terribly useful for our purpose. Take a look at **Figure 9**, for example. It has 77 hits.

You can create the Data Dictionary view Z_DATA_ELEMENT, which will show only the data tables that use your data element.

Figure 10 Components of a View for Searching Data Elements

Z_DATA_ELEMENT	Base tables
Table	
DD03L	
DD02L	
DD02T	
DD04T	

Z_DATA_ELEMENT	Joins			
Table	Field	=	Table	Field
DD03L	TABNAME	=	DD02L	TABNAME
DD03L	TABNAME	=	DD02T	TABNAME
DD03L	ROLLNAME	=	DD04T	ROLLNAME

Z_DATA_ELEMENT	Fields	
Base table	Base field	View field
DD02L	TABNAME	TABNAME
DD03L	FIELDNAME	FIELDNAME
DD02L	TABCLASS	TABCLASS
DD03L	POSITION	FLD_POSNR
DD03L	KEYFLAG	KEYFLAG
DD03L	MANDATORY	MANDATORY
DD03L	ROLLNAME	DATA_ELEM
DD03L	CHECKTABLE	CHECKTABLE
DD03L	NOTNULL	NOTNULL
DD02T	DDTEXT	TBL_DESCR
DD04T	DDTEXT	D_E_DESCR

Z_DATA_ELEMENT	Selection conditions				
NOT	Table	Field name	Op	Comparison value	AND/OR
	DD04T	DDLANGUAGE	EQ	'E'	AND
	DD02T	DDLANGUAGE	EQ	'E'	AND
	DD03L	AS4LOCAL	EQ	'A'	AND
	DD02L	AS4LOCAL	EQ	'A'	AND
	DD02L	TABCLASS	EQ	'TRANSP'	OR
	DD02L	TABCLASS	EQ	'CLUSTER'	OR
	DD02L	TABCLASS	EQ	'POOL'	

Create the view with **Tools → ABAP Workbench → Dictionary → Object name=Z_DATA_ELEMENT → Views → Create**, then enter the values shown in **Figure 10**.

Use this tool with **Tools → ABAP Workbench → Dictionary → Object name=Z_DATA_ELEMENT → Views → Display → Utilities → [Display data or Table contents] → DATA_ELEM=[the name of the data element you are searching for] → [Execute or F8]**.

This view will give you a list of only the tables that use your data element.

Technique #9: Views

Transaction data is normally stored in several hierarchically connected tables, and R/3 may have one or more views defined for the transaction you're dealing with. You can search through those views to find other associated tables. Any one of those might store your data. For example, assume you want the Order Quantity data from the Sales order: Single-Line Overview screen. It's listed as RV45A-KWMENG, but you find that RV45A is a structure. You already know that **Order** is VBAK-VBELN, so you look for views in the Where-used list for VBAK. View VIVEDA connects VBAK, VBUK, and VBAP, and you find KWMENG in the VBAP table.

Technique #10: Logical Databases

Tables that are related in a business process are frequently connected in logical databases. The VBAK "Where-used" list for logical databases, for example, includes AAV, which also shows the connection to VBAP.

Technique #11: Other Screen Fields

One or more of the other fields in the same transaction may be directly connected to their database tables. If you can discover the **Header** table of the transaction, it may include the field you want. For example, assume you want the **Sold to** data from the Sales Order initial screen. It's listed as RV45S-KUNNR, but you find that RV45S is a structure. Its data element is KUNAG, but Release 4.0B uses that data element in 276 tables, so it would be tough to find it there. On the same screen, the **Order** or **Standard Order** is listed as VBAK-VBELN. When you look up the VBAK table, you find that it also includes KUNNR.

Technique #12: Application Hierarchy

In 4.0B or above, go to **Tools → ABAP Workbench → Overview → Data Browser**. Press **F4** in the *empty* **Table name** field, then click on **SAP Applications → Application Hierarchy**. Drill down through the menu tree to find the tables associated with your application. Your field may well be in one of those tables.

Technique #13: Graphics

In 4.0B or above, go to **Tools → ABAP Workbench → Dictionary**, then type in the name of a table you believe is related to your field. Follow the path **Display → [Extras or Utilities] → Graphics**. You'll see a graphical representation of the selected table and its links to other tables. You can use this to see the entire family of tables related to a business object, and you may be able to find where your field is located.

The Last Resort: Techniques 14-18

These are the techniques of last resort, which you should turn to only if you've exhausted all other possibilities.

Technique #14: Change Records

R/3 allows for changes that are made to many documents to be recorded in change records, and frequently the change information includes the name of the table and field affected. If change recording is turned on, you can change the field you're looking for in such a document, and then read the change record to find the table and field name. Depending on the R/3 release, here are some examples of menu paths to change records:

- In Plant Maintenance or Service Management

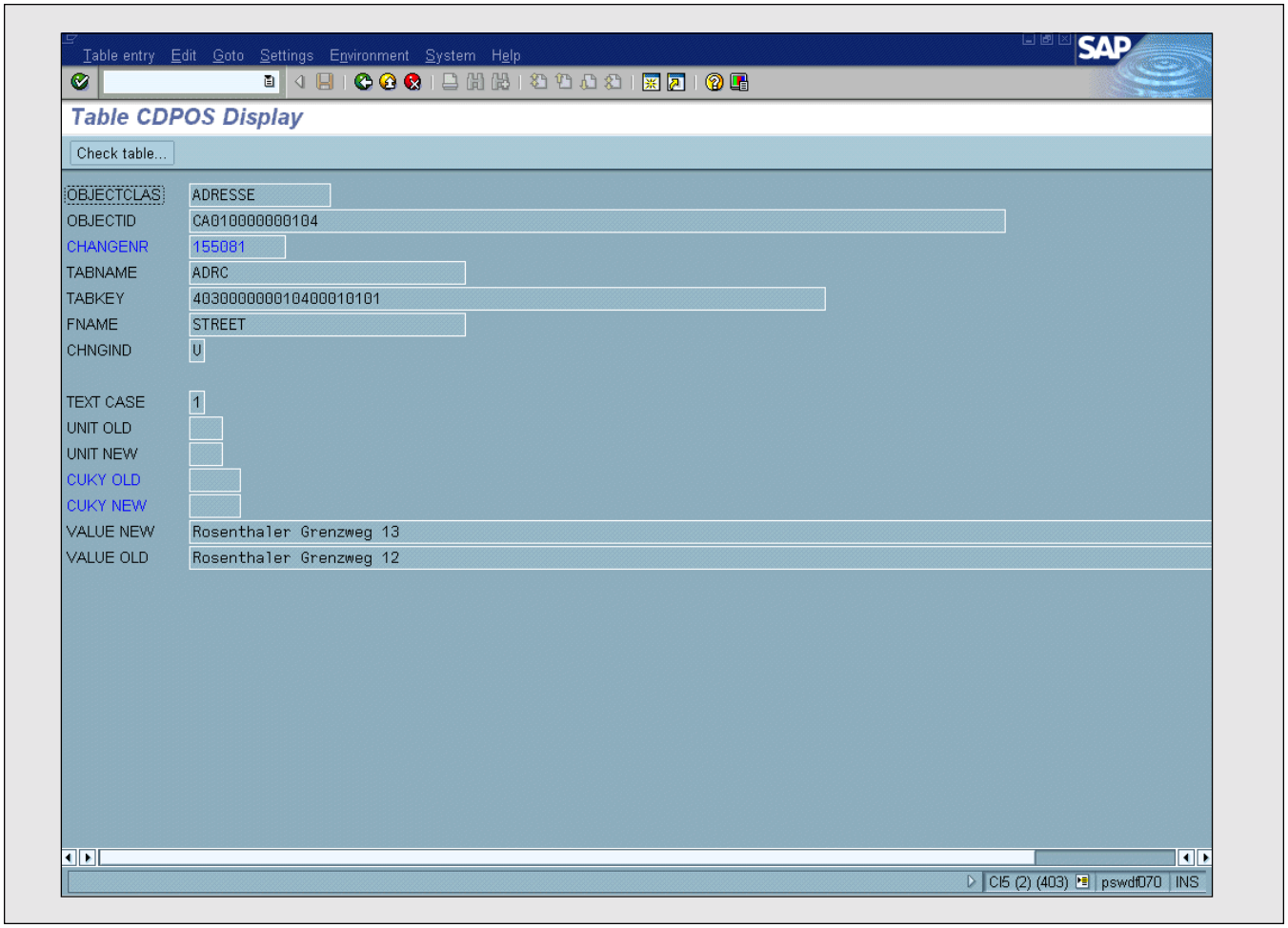
  Technical Objects: **Extras → [Display Changes or Environment] → Display Changes**

  Order: **Extras → Order documents → [Changes or Extras] → Display Changes**

  Notification: **Extras → Notification documents → Changes**

  Contracts & Planning: **Environment → Changes**

Figure 11 Change Table Illustration in Release 4.0B



- **In Sales and Distribution**

  Order: **Environment → Changes**

  Delivery: **Environment → Changes**

If more than one change is shown, drill into the change you're interested in (generally the most recent). Tables TCDOB and TCDOBT list all the available object classes for change document creation. You can get clues there about whether your application supports change documents.

Technique #15: Change Documents

If you can't find the change record in your transaction, have a look in the change document tables CDHDR and CDPOS (now there's a mix of English and German naming conventions: "Change Document Header" and "Change Document Position"!) as follows. After you have changed and saved the field you're searching for, open CDHDR in the ABAP Workbench dictionary, then follow **Utilities → Table contents** to find the change record. In the selection screen enter today's date, time range, and your user name. It should pull the one matching record. **Ctrl+C** (copy) the value of CHANGENR, and use it to pull the appropriate record from CDPOS (**Figure 11**). This record contains the names of the table and field affected, as well as the new and old values, so you can ensure that you're looking at the right record.

Figure 12 Data Path from Service Order to Measurement Document

Tables	Joins	Conditions
AUFK or AFIH	AUFNR	
AFKO	AUFNR AUFPL	
AFFH	AUFPL OBJID	OBJTY='FH'
CRVE_A	OBJID EQUNR	
EQUI	EQUNR OBJNR	
IMPTT	MPOBJ POINT	
IMRG	POINT	

Technique #16: SQL Trace

If all else fails, it's time for some serious archeology. In one session, go to your transaction but don't run it yet. In another session, go into **Tools → ABAP Workbench → Test → SQL Trace → Trace On → Ok**. Switch to your transaction session and run your transaction. Switch back to the trace session and click **Trace off → List trace → [Execute or Ok]**. You'll have a list of all the SQL calls to the tables with their parameters. It may be easier to download it into a word processor document for searching. Use **System → List → Save → Local file → unconverted → Enter → File name=[c:\mypath\filename.doc] → Transfer**. Open "filename.doc" in Word, search for your field name, and notice the tables it is associated with. It may involve detective work to determine which table actually stores the data. This approach requires an understanding of SQL commands and principles.

Technique #17: Runtime Analysis

Alternatively, use runtime analysis with **System → Utilities → Runtime Analysis → Execute → [Transaction=your_transcode or Program=your_program_name] → F8**. After you post the transaction or complete the program, the system will return you to the Runtime Analysis screen. Choose **Analyze**, then **Tables** or **Table Hit List** (**F6**) to see all the transparent and pool tables that were accessed during the transaction. **Hit List** (**F5**) will show you all the program calls and specify the ABAP program that was running. Sometimes the Where-used list from the "Structure" display with **Programs** checked will list a program in which the field is associated with the database. Compare the programs in that list to those you found in the Runtime Analysis Hit List to narrow your search.

Technique #18: Data Contents

Finally, extract some data from the set of tables associated with your transaction, and copy those records into a text file. Use the "Search" function in your word processor to find data flows by content where the source and destination fields may have different names and data elements. R/3 often uses internal numbers that you'd never think of, such as Object ID and Object Type, to link records. Their values, however, can be quite distinctive, and you may be able to find the path of data flow by searching for the values (ignoring the field names) when you have pulled data from several transactions.

For example, to find Measurement documents associated with a service order, we needed to trace the connections shown in **Figure 12**. There was nothing obvious about which fields linked the data.

We created a service order and several Measurement documents in R/3. Then we pulled out the records starting with the service order numbers, and chased through the various joins to pull all the associated records. While displaying each table in SE11 we used **Utilities → Table contents** to extract the data.

We exported the records as text documents using **System → List → Save → Local file** and merged them all into a single Word document. We then used Word's "Find" function to follow the data through the tables by following the value of the contents. OBJID, OBJNR, and MPOBJ contain internal numbers that "mean" nothing to the transaction, yet they linked the records.

Helpful Hints

✔ Many transactions have header information and detail or item lines in separate linked tables. Remember that R/3 is a German product. "Kopf" is German for "head," and "position" can be interpreted in German as "detail" or "item." You'll often find header/item table pairs with "K" and "P" in their names, such as VBAK and VBAP. When you are looking for linked tables containing the data for a transaction, this tip may help you identify one if you have the other.

✔ There are small differences between R/3 Releases 3.1I, 4.0B, 4.5B, and 4.6B for the menu paths and directions used in this article. For example, Release 3.1I menus refer to the ABAP/4 (Workbench, Editor, etc.) whereas higher releases refer to ABAP. The Release 4.0B **ABAP Workbench → ABAP Editor** menu path shows up in Release 4.6B with the extra step **ABAP Workbench → Development → ABAP Editor**. The figures in this article were taken from a 4.0B system unless there was an important difference in appearance on another release. Many installations today are still using Release 3.1, and most are running 4.0 or below, so those images are important. The corresponding 4.5 and 4.6 screens are virtually identical to Release 4.0B except for Figure 2, so there's no need to show the other figures in those releases as well.

✔ Release 3.1I and below can't execute technique 5 because they don't have the watchpoint functionality in their debuggers, or techniques 12 and 13 because they don't have the application hierarchy or graphics functions.

Conclusion

SAP's three-tiered architecture is the backbone of an R/3 infrastructure. It's what makes an R/3 system flexible, reliable, and open, all at the same time. It's also what makes locating underlying data, at times, so inherently difficult. Hopefully the techniques presented here will help. Opt for the ones that you find easiest and most convenient to use. Leave the others as techniques of last resort. Some fields may still resist these techniques; there are no doubt other ways yet to chase them down. If you know of other techniques (or corrections to these techniques), please e-mail me at dennis.barrett@sap.com.

Acknowledgements

Thank you to the many SAP and partner consultants who gave me suggestions and feedback, and to those who reviewed this article.

Dennis Barrett is an applications consultant with SAP America who focuses on Service Management/Customer Service and the Service Provider solution. He has been consulting in computers for over 15 years, always blending applications and programming. He's a certified ABAP programmer and often facilitates communications between business process owners and programmers. Dennis is the author of "SAP R/3 ABAP/4 Command Reference," published by Que/MacMillan, and wrote "Customer Service ABAP Tables for Programmers," published this spring in the PM/CS e-Newsletter, which is sent to all SAP consultants in Plant Maintenance and Customer Service. Dennis maintains a personal Web site at ***www.mindspring.com~dennis.barrett****, which includes copies of the PM/CS articles as well as white papers and other items of interest to ABAP programmers and CS consultants. You can reach him at dennis.barrett@sap.com.*

