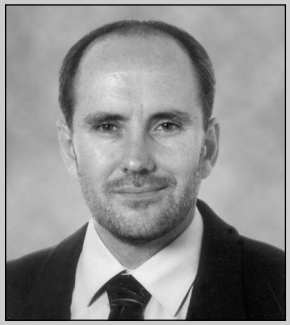
Mastering the Administrative and Development Tasks Required to Put Archived Data in Easy Reach of Every User

Dr. Rolf Gersbacher and Helmut Stefani

Dr. Rolf Gersbacher
Data Archiving Development and Coordination
SAP AG



Helmut Stefani
Data Archiving Development and Coordination
SAP AG

(complete bios appear on page 97)

mySAP Technology Data Archiving relocates obsolete application data — that is, data that belongs to closed business transactions, and is therefore no longer needed for online operations — from the database to archive files. The whole point of an archive exercise is to preserve the data so that it may be referenced at a later date for business, legal, or technical purposes.[1] And yet, despite the fact that many users and administrators know that using SAP's data archiving solution to relocate old application data from the database to archive files is an absolute must for keeping a tight rein on the long-term database volume, they still feel uneasy about "their" data being stored in a place that they perceive as difficult to access. Although understandable, their fear is groundless. There are a variety of tools and methods available for retrieving archived data.

One way that users can access archived data is by searching the archive for individual documents from within an online document's display transaction. We showed you how this is done for the FI_DOCUMNT archiving object in our previous article.[2] Users can also access archived data using the read programs that transaction SARA provides for many archiving objects. But, as you will see, this approach, which displays just the contents of one data object, is often not sufficient. Users typically need more information. With regard to

[1] Business reasons refer mainly to the controlling or auditing departments, whereas legal reasons are usually imposed by tax authorities. Technical reasons mostly refer to information retention requirements, such as ensuring long-term readability of data by regularly migrating it from old to new media.

[2] "Data Archiving Essentials — What Every Administrator Needs to Know," *SAP Professional Journal*, March/April 2001.

a sales order, for instance, they would need to see adjoining data objects, such as the bill or the shipping document, so that they can reconstruct a historical business context. Displaying the broader business context (relative to both online and offline objects) in which a data object was once embedded *is* possible with the Document Relationship Browser (DRB).

In this article, we examine the DRB tool, first from the perspective of the end user, then from the perspective of you administrators and developers who need to set up your SAP environments to support this tool.[3] We will also look at the Archive Information System (AS). This is the tool that enables users to access and display individual archived business documents. In short, AS is used for retrieving a document, and DRB is used to evaluate its fit within the context of a business process that may comprise a large number of individual documents. Note that in addition to using AS and DRB in this complementary fashion, these tools can also be used separately.

In order for users to avail themselves of these tools, administrators and developers need to carry out several tasks.

Administrators need to:

- Define the archive information structures, which are the basis for data retrieval with AS.
- Fill and empty the archive information structures.
- Monitor the status of the archive information structures (i.e., which archive files are considered during retrieval and which are not).

Developers need to:

- Write new read programs and/or customize existing ones for accessing archived data, if required.
- Integrate customized programs into the AS framework.
- Develop user-specific business views, if required.
- Integrate customer-specific tables in AS (see "Appendix B — Integrating Customer-Specific Tables in the Archive Information System").

This article shows you how to accomplish these tasks. It also offers valuable behind-the-scenes information and concrete instructions for customizing existing retrieval programs and developing new retrieval solutions. Hopefully, the details, tips, and tricks we provide here will enable you to remain calm, cool, and collected when users ask: "I need to access my archived sales docs. Is that possible?"

You Can Assure Users That Archived Data Is Easily Accessible!

In **Figure 1**, we offer you a look at how an archived SAP R/3 sales order is displayed via a read program from within transaction SARA. The upper part of the display contains the header data of the sales order; the lower part contains item-specific data.

As you can see, there is no information displayed here for adjoining data objects, such as the bill or the shipping document. And, let's face it, if the user's search has been launched (as most are) to reconstruct a historical business context in order to address a customer query, complaint, or service request, what is displayed here falls short of what is needed. A sales order, for example, does not exist in a vacuum. It is part of a more complex business process that is usually made up of several data objects.

Figure 2 depicts the flow of information in a sales order process:

[3] Access to archived SAP data can only be performed via these SAP proprietary tools. Non-SAP tools are not supported.

Figure 1 Displaying a Sales Order from Within Transaction SARA

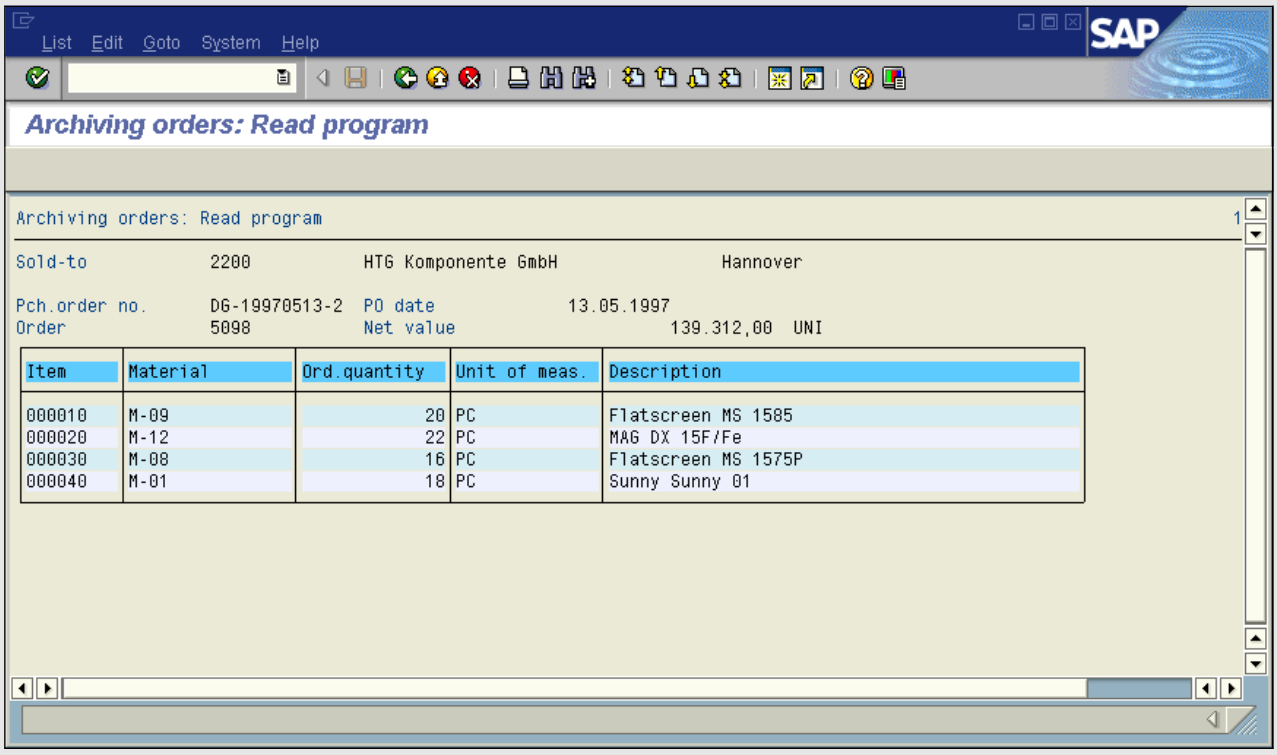
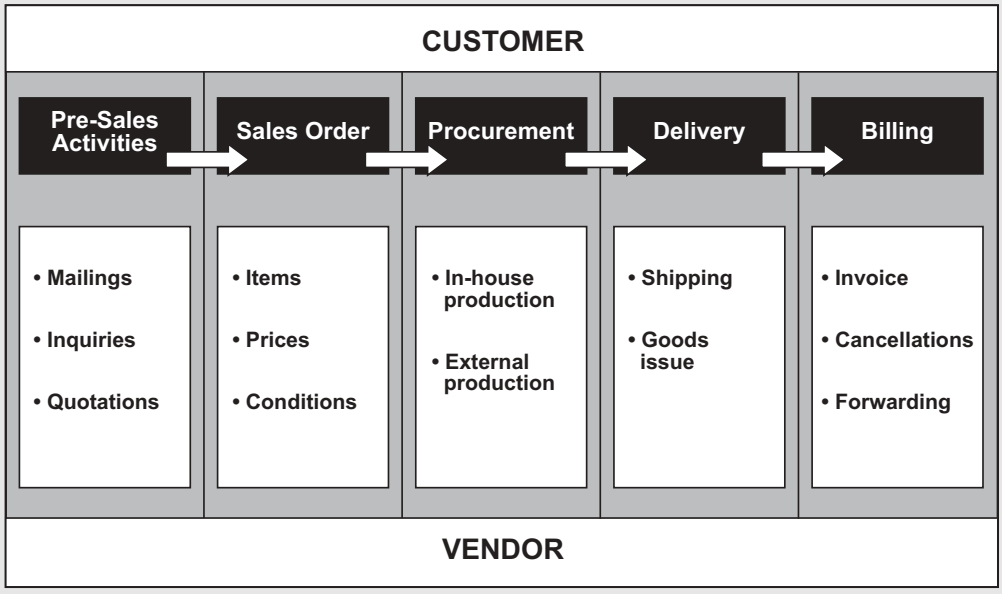


Figure 2 The “Sales Order Process” Business Scenario



Archiving objects involved: Activities, Sales Documents, Deliveries, Billing Documents, Purchase Documents, Material Documents, Financial Accounting Documents

1. The process starts with the pre-sales activities. For example, based on a customer's inquiry, a quotation is created and sent to the customer. The archiving objects involved in this step are SD_VBKA (sales activities) and SD_VBAK (sales documents) of the order type QT (quotation).

2. When the customer accepts the quotation, an order with reference to this quotation is then created in the system. The order contains customer-related information, information about the deliverables, and information on the pricing and delivery conditions. The relevant archiving object is SD_VBAK (sales documents) of the relevant order type — e.g., OR (standard order).

3. During procurement, the SAP R/3 system determines where the goods are to be obtained from. They can either be delivered by an external vendor, taken from the company's warehouse, or produced in-house. If the latter is the case, the system will probably also create a production order. The following archiving objects are associated with the procurement phase: MM_EKKO (purchasing documents), MM_MATBEL (material documents), and, if applicable, PP_ORDER (production orders).

4. As soon as the procurement is completed, a delivery is created in the system with reference to the sales order. All the delivery-specific data, such as materials and quantities, is obtained from the sales order. A delivery document comprises all the activities required to process a delivery completely, including picking and packing the goods as well as planning and supervising shipping. The archiving objects involved are RV_LIKP (deliveries) and SD_VTTK (transport).

5. After the goods have been shipped, a billing document is created in the system.[4] The necessary data is taken from the relevant sales order and delivery documents. In financial accounting, the billing document is used as the main data for payment processing and monitoring. The archiving objects involved here are SD_VBRK (billing documents) and FI_DOCUMNT (financial accounting documents).

Even from this simple scenario, it's clear that several data objects contribute to the same business process: pre-sales activities, sales orders, purchase documents, material documents, deliveries, and so on. Some of the data objects may have been archived. Some may still be in the online database. How do you establish and display the complete process, regardless of whether its data objects are still online or have been moved offline?

The answer to this question is the Document Relationship Browser (DRB).

The Document Relationship Browser (DRB)

The Document Relationship Browser (DRB) can display a *complete* sales order process for a user.

The evaluation of the process can be started either using transaction ALO1 or from within the Archive Information System's Archive Explorer tool (provided that the entry node has been archived). There are several entry points possible, depending on whether the end user's working area is SD, MM, or FI. Based on the entry point chosen, the tool establishes the context of the process. In the screen shot shown in **Figure 3**, the entry point is a sales activity.

In **Figure 4**, the sales activity can then be specified more closely according to business criteria, such as document number, contact type, or sales organization.

[4] Other documents that can be created during sales order processing are outbound invoices, which are printed and sent to the customer, or mail documents from the correspondence with the customer. In most cases, this type of document is stored in an external (document) storage system.

Figure 3 Entry Point for Process Evaluation: Sales Activity

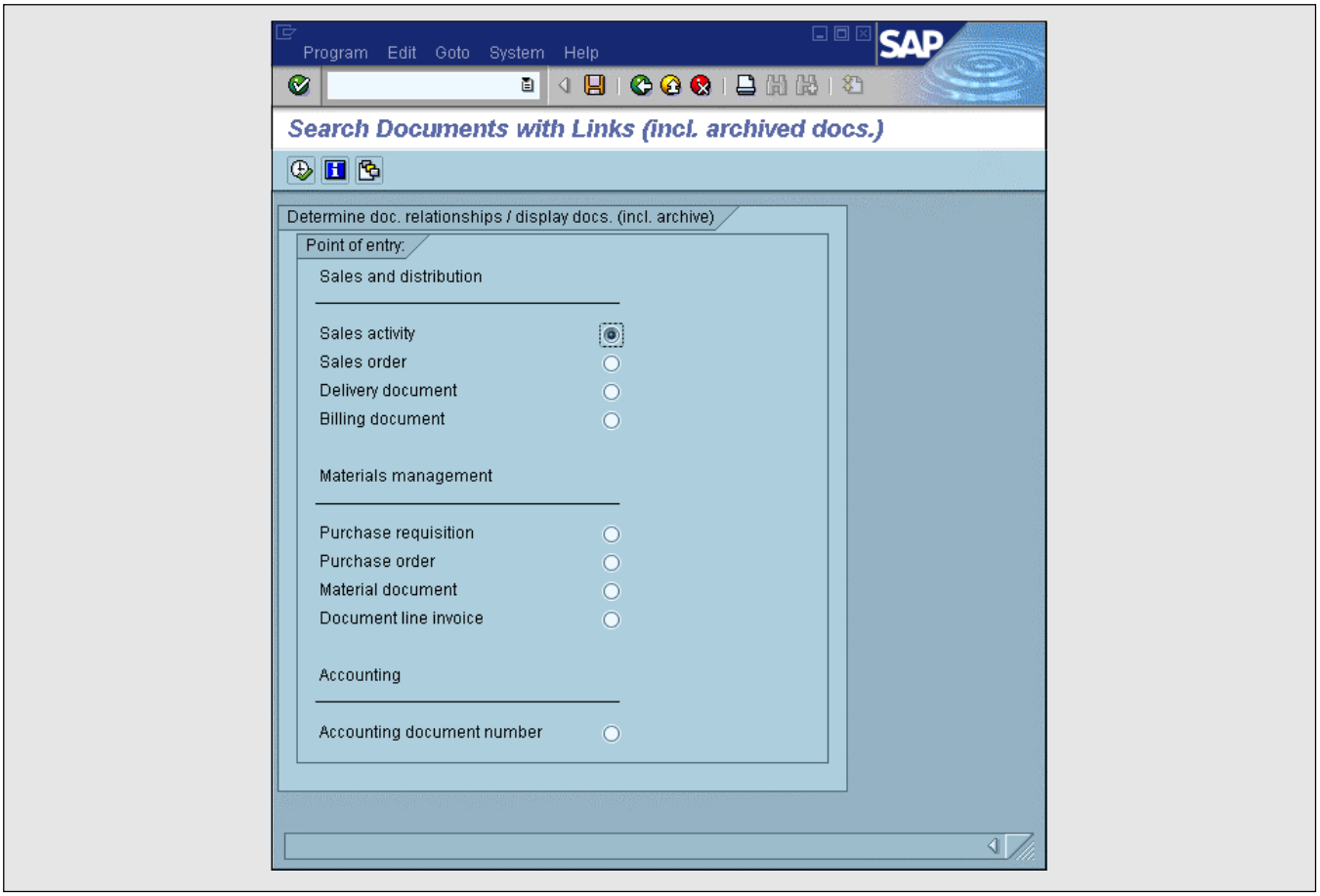
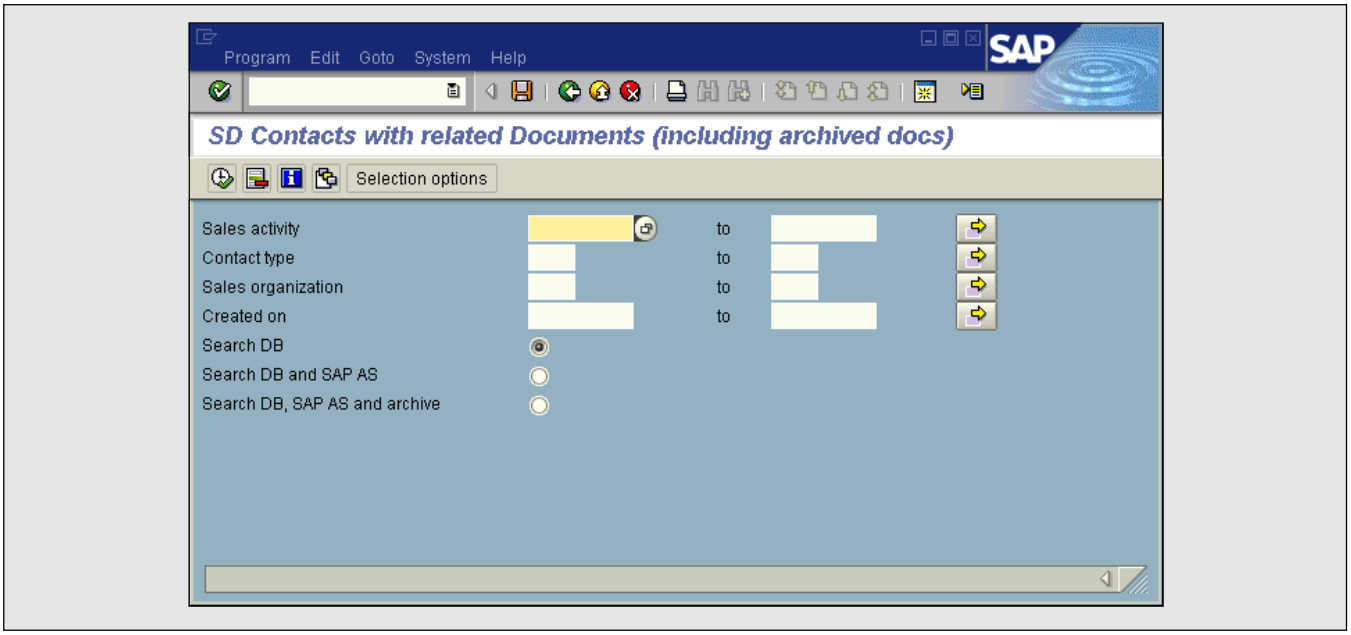


Figure 4 Providing Search Criteria for the Process Evaluation



The Document Relationship Browser (DRB) in a Nutshell

When stored in archive files, data objects are always grouped according to the archiving object to which they belong (see also "Appendix A — The Structure of an Archive File"). The business process in which they were once embedded usually gets lost. This is the reason we developed the SAP Document Relationship Browser (DRB). A process or tool was needed to enable a process-oriented evaluation of data objects, regardless of their location in the database or in the archive.

DRB was devised to enable users to display, in its entirety, a business process to which an individual document belongs. This includes any other items that the document is connected to, such as texts, change documents, etc., or generally speaking, all preceding or subsequent documents. The starting point for the business process display is a document that a user selects — for example, a particular delivery for which the user would like to examine the sales document or bill. And remember, DRB is able to process both archived documents and documents that are still in the database.

The term **business process** denotes a set of activities carried out in an enterprise in order to obtain a customer benefit. A business process usually consumes enterprise resources, such as processing time or funds. The SAP R/3 system captures every single step of a business process in that it creates documents and stores them persistently in the database. Several departments of the enterprise are usually involved in a process. The system ensures that there are no gaps in the process, which makes it easy to tell who carried out which action and when, and to provide the kind of comprehensive documentation trail that is required for legal, reporting, business, or technical purposes.

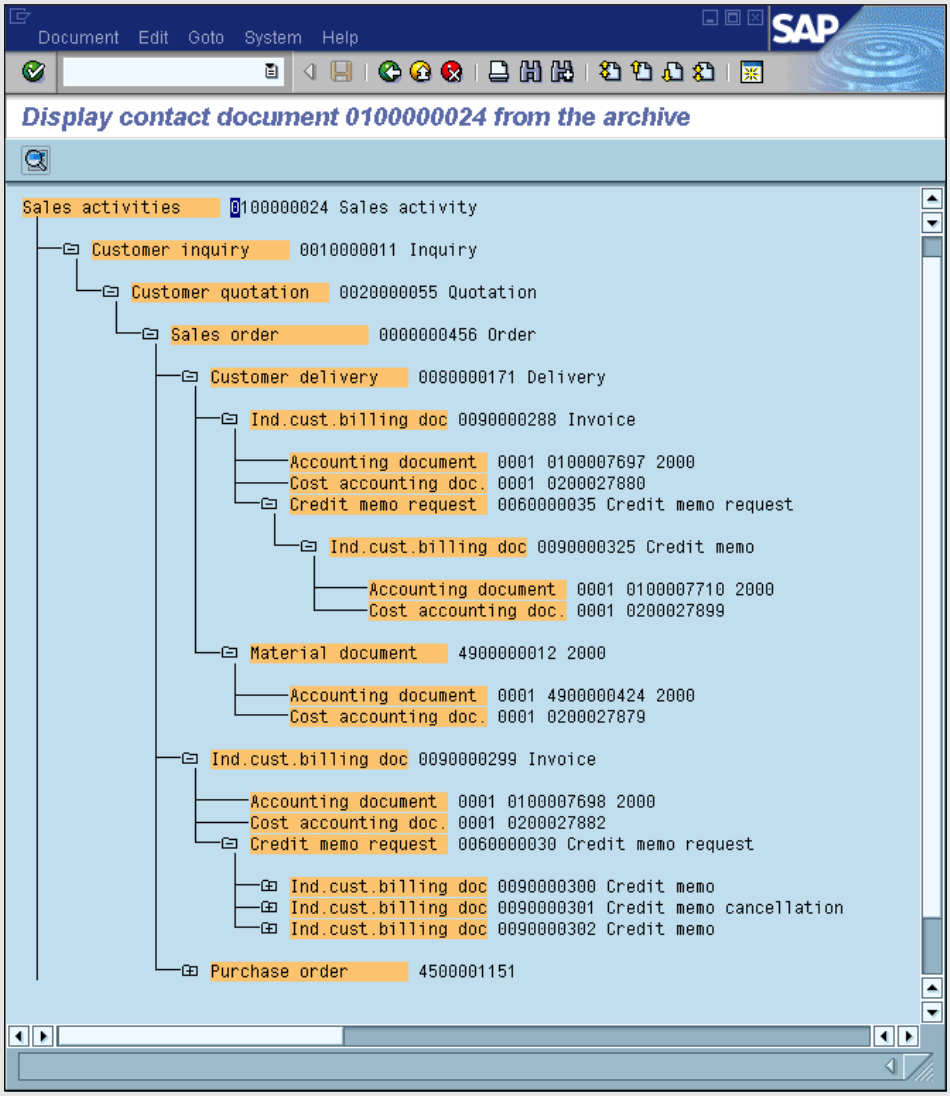
Specific product information, for example, must be made available to the company's customer service department for the entirety of the product's lifecycle. This includes construction-relevant CAD drawings, bills of material, in-built components, production aids, and work schedules. From this list, you can see that the end product is defined by various data objects. In some cases, it may be necessary to reconstruct the whole process in which the product was involved. In many cases, responsibility for the product does not end with its use, but can also include recycling. Some data objects with product information may still be in the database, whereas other data objects may have already been archived.

In the past, the SAP R/3 system offered a few programs that enable a combined evaluation of data objects from the database and from the archive. But, until now, no tool was available that could display a business process in its entirety — that is, all data objects, regardless of whether they are still in the database or whether they have been archived. This gap has now been closed with the development of the Document Relationship Browser.

Here, the user can also specify the search strategy according to which data locations the program is going to evaluate:

- **Search DB** — Only the database will be searched.
- **Search DB and SAP AS** — In addition to the database, the AS information structures will be searched according to the selection criteria specified. (Recall that "AS" is the abbreviation for Archive Information System.)
- **Search DB, SAP AS and archive** — In addition to the first two locations, all existing archive files will be sequentially searched. This kind of search may take quite a while.

Figure 5 Displaying the Process Context for a Sales Activity

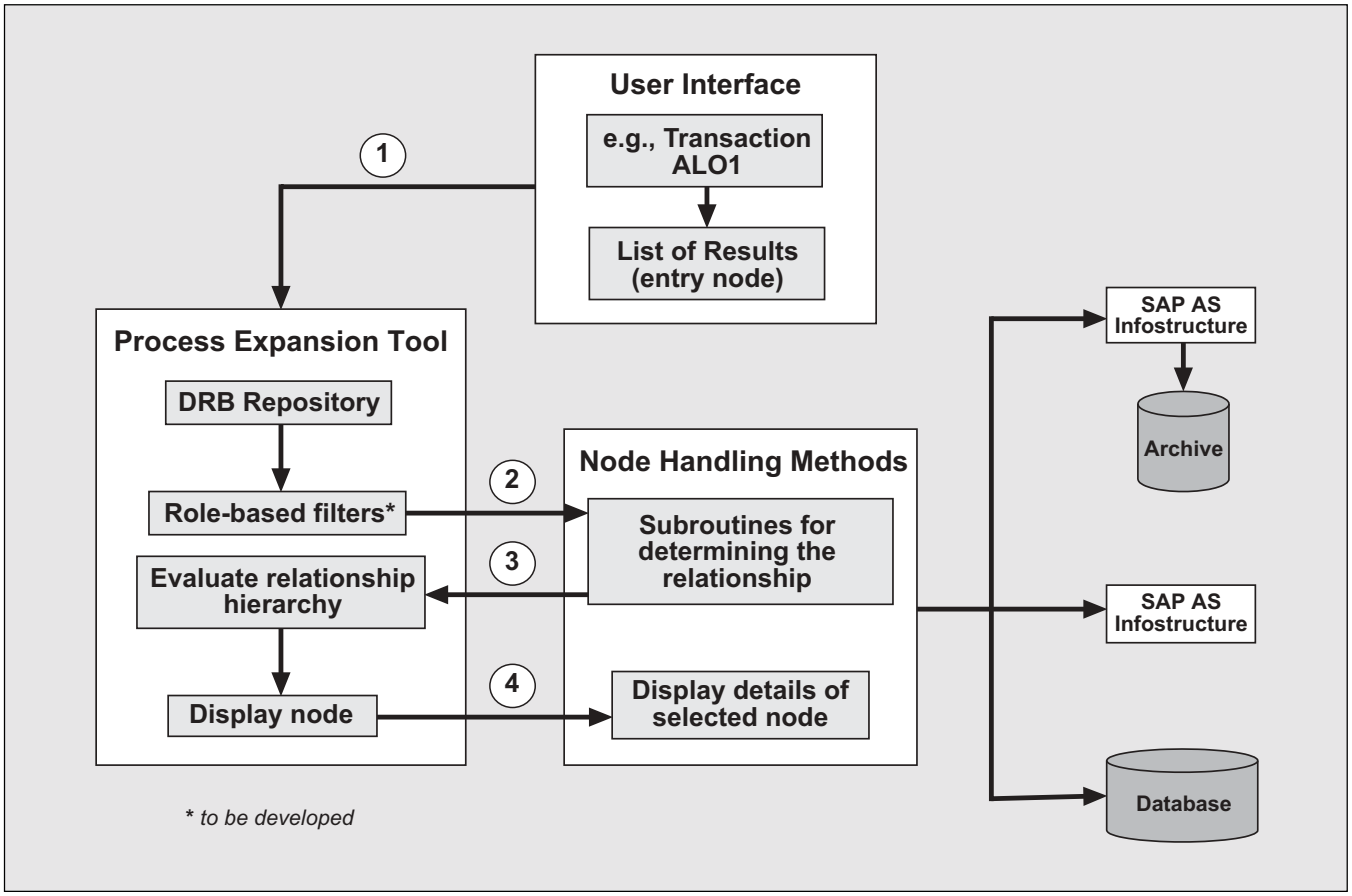


Executing this screen results in a hit list from which the user can select the desired activity. On the basis of the activity chosen, the program determines all the data objects that are directly or indirectly linked to the activity.

Initially, the Process Expansion tool, which is automatically invoked, identifies the related documents, and displays them in a hierarchical tree structure. Then, in a second step — after the tree has been expanded — the Process Expansion tool identifies the second hierarchy level and adds it to the tree structure.

In the screen shot in **Figure 5**, you may recognize the business process that we sketched. The process starts with a sales activity (here, it is the contact with document number “0100000024”) that results in a sales order. The next steps include creating material documents, manufacturing and shipping the product, and, finally, creating the relevant invoices.

Figure 6 Process Flow in the Document Relationship Browser



Double-clicking an individual document leads the user straight to the relevant display transaction.[5] If the data object was already archived, it will be displayed from the archive. When evaluating the context of the process, the program takes into account not only data objects that still reside in the database, but also data objects that were archived.

How Does DRB Display All Relevant Documents?

To understand how DRB is able to display all the documents associated with a particular business process, we refer you to **Figure 6**.

1. To start the process, a user can use either transaction ALO1[6] or the AS Archive Explorer tool (provided that the entry node has been archived). If the user opts to use AS to display the related items of a sales document, for example, from the AS Archive Explorer, he or she would choose

[5] How the process hierarchy is actually displayed always depends on the entry node — that is, the first object that the user selects.

[6] Transaction ALO1 is not to be considered *the* standard DRB transaction, but rather the first comprehensive implementation of an entry point to DRB that provides access to some of the most commonly used business documents, such as sales documents, purchase orders, accounting documents, etc. Depending on the specific needs for data evaluation within an application, other entry points may be implemented (this requires certain programming expertise and extensive knowledge of the business data objects involved).

"DRB: SD-Order" when selecting the desired display function, and then "Related items" in the display screen of the sales document. If the user opts to use transaction ALO1, he or she would double-click an entry node, and the selection result would be transferred to the Process Expansion tool (step 1 in Figure 6).

2. After evaluating the relevant Customizing entries, the Process Expansion tool sequentially calls all the available methods (subroutines for determining the document relationship).[7]

3. The subroutines identify the relationships that exist between the entry object and its connected objects (process context), and return this information to the Process Expansion tool.

   Generally, the program can take into account three different data sources when evaluating the process context:

   - Data objects that reside in the database
   - Archived data objects about which information has been stored in one or several archive information structures (mini-documents)
   - Archived data objects that are evaluated directly from the archive using an archive information structure

   The actual knowledge about the object relationships (such as the preceding and subsequent documents, or in which tables or archiving information structures the relationship data is kept) is stored in the subroutines.

4. Initially, the Process Expansion tool identifies the related documents, and displays them in a hierarchical tree structure. Then, in a second step — after you have expanded the tree — the Process Expansion tool identifies the second hierarchy level and adds it to the tree structure (see, for example, Figure 5).

The Archive Information System (AS)

The Archive Information System (AS) is a generic[8] tool that enables users to access and display archived business documents. Users can access the Archive Information System from the initial screen of Archive Administration (transaction SARA) by choosing "Information system." It is completely integrated within the data archiving environment, and has been part of the standard SAP R/3 distribution since Release 4.5B.[9] AS is primarily used for displaying single documents. It is the administrator's task to define and fill the archive information structures, thus enabling the user to use the tool.[10]

To give you a flavor of how the user actually uses AS, let's take a look at some screen shots taken during the retrieval of a sales document.[11] After entering the names of the archiving object and a suitable archive information structure in the Archive Explorer

[7] Future releases of DRB will provide role-based filters that restrict the number of methods on a per-user basis (since it is not desirable for all users to be able to view all related documents).

[8] This means that AS is independent of archiving objects. It can be used for all available archiving objects, including customer-specific objects, and does not require any modifications to existing programs. Prior to AS, extensive programming was required to implement functionality similar to what AS offers today.

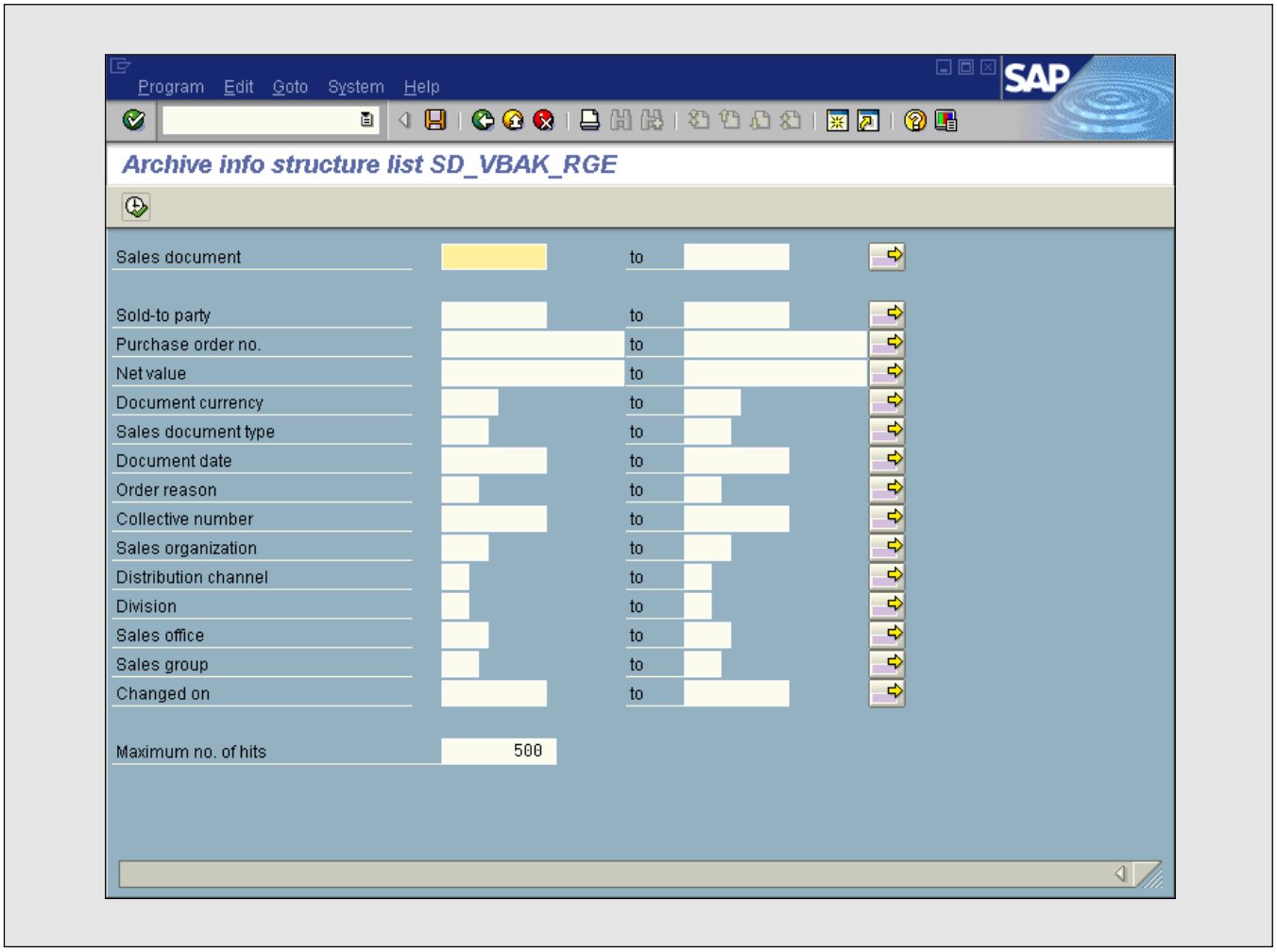
[9] If you are running an older release (any release from 4.5A down to 3.0D), you can install AS by following SAP note 99388.

[10] In contrast to AS, which uses only one data source (archive information structures), DRB can cope with several data sources:

- Online data
- Archived data
- Archive information structures

[11] Please note that this is just a sample archive information structure that we created by selecting fields from a standard field catalog. Usually the administrator sets up new information structures if those provided by SAP are inadequate.

Figure 7 The Archived Document Selection Screen



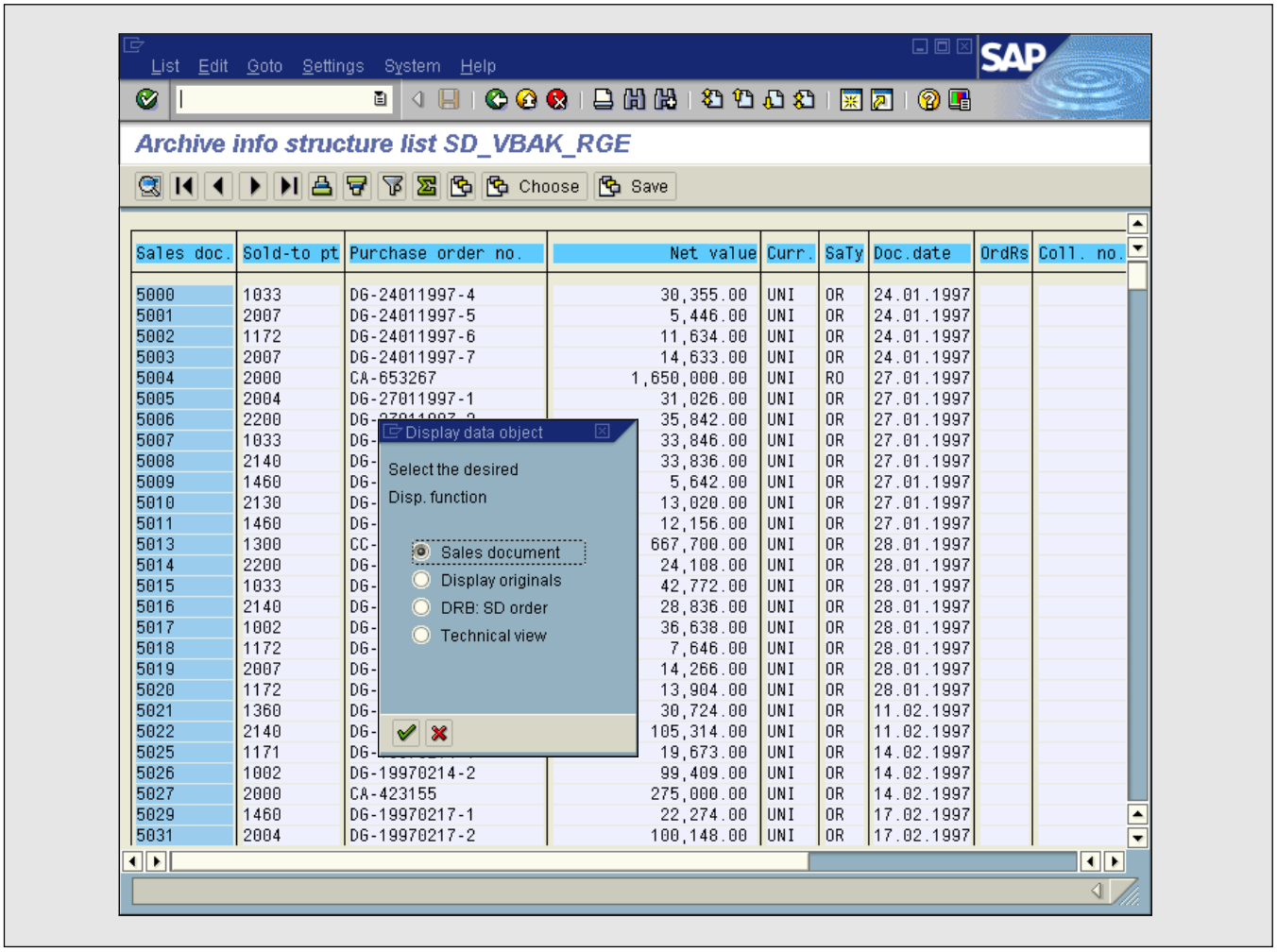
initial screen, the user accesses the screen in **Figure 7**. Here, he or she can select the archived sales document(s) to be displayed.

On executing the selection screen, the user is taken to the appropriate list of sales documents (if more than one has been selected, the list can be quite large). Double-clicking on a list entry (which corresponds to an individual sales document) displays a pop-up menu, where the user can select the view in which the document is to be displayed (**Figure 8**). The options include:

- **Sales document** — This option displays the document from the archive.
- **Display originals** — This option displays the original scanned document from the storage system (if available).
- **DRB: SD order** — This option branches to the DRB tool, where the user can evaluate the process the sales document belongs to.
- **Technical view** — This option provides information regarding the tables where the archived sales document originally resided before it was archived.

The terms "archive information structure" and "field catalog" are crucial to understanding AS. Let's begin by examining these key terms.

Figure 8 List of Data Objects for an Archive Information Structure



An *archive information structure* is an index, or an array of pointers, containing all the information needed to identify individual archived data objects. This information is contained in business-related fields, such as the document number, and in technical fields that determine the position of the data object within the archive file.[12] An archive information structure can also contain information regarding the contents of the data object, such as a short description, the fiscal year, or the name of the person who created the data object. By having a look at this "mini-document," a user can easily recognize the nature of the archived data object — that is, what kind of data it contains. In many cases, this information gives the user everything he or she needs, so the user can forgo actually having to access the archive.

Physically, as is also the case with other information systems in SAP R/3, such as the Logistics Information System or the Sales Information System, an archive information structure is nothing other than a transparent table that is stored in the database. In AS, each information structure is assigned to a single archiving object. An archiving object can have several archive information structures assigned to it. When you create a new archive information structure, the system automatically generates, along with the transparent database

[12] These are the fields ARCHIVEKEY and ARCHIVEOFS.

Figure 9 Example of an Information Structure for CO Orders

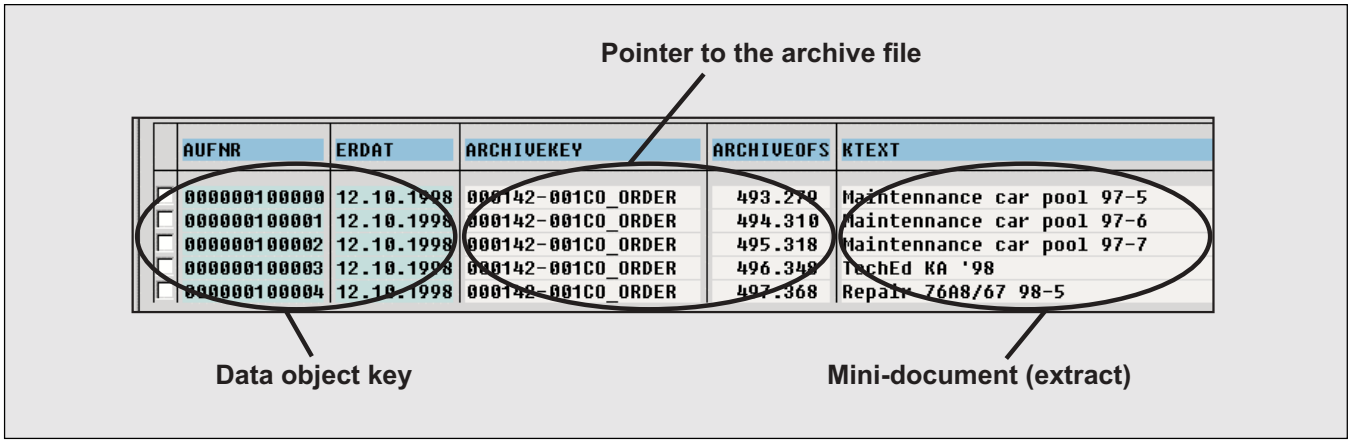


table and the information structure itself, a suitable reporting program.[13]

Figure 9 shows how an archive information structure is organized. It is made up of three sections:

- The **data object key** contains the fields that identify the data object from a business point of view. Note that as a default, the client also belongs to the key. However, it does not appear in displaying the information structure.

- The **pointer to the archive file** provides the archive key (ID created and managed by the Archive Development Kit, or ADK,[14] to identify individual archive files),[15] and the location of the data object in the archive file (or, to be more precise, its exact starting position or *offset*). This is needed for direct access to the data object.

- The **mini-document** section contains additional information on the data object, such as a short description. This information is obtained from the data object.

All this information is stored in the transparent database table corresponding to the archive information structure. When the user starts the reporting program, it first evaluates the information in this table and displays the results in a list. At this point, the archive has not yet been accessed. Only if the user wants to obtain more details about the data object (by double-clicking the desired list entry) does the program access an archive.

An archive information structure should contain all the fields that a user may wish to use as search criteria, and whose content is to be displayed when retrieving the archived data. Typically, it is your task as an administrator to set up the archive information structures so that the user can use them. You can change the organization of an information structure by adding more fields, such as the billing date, net value, sales unit, etc., or by removing fields that are not needed.

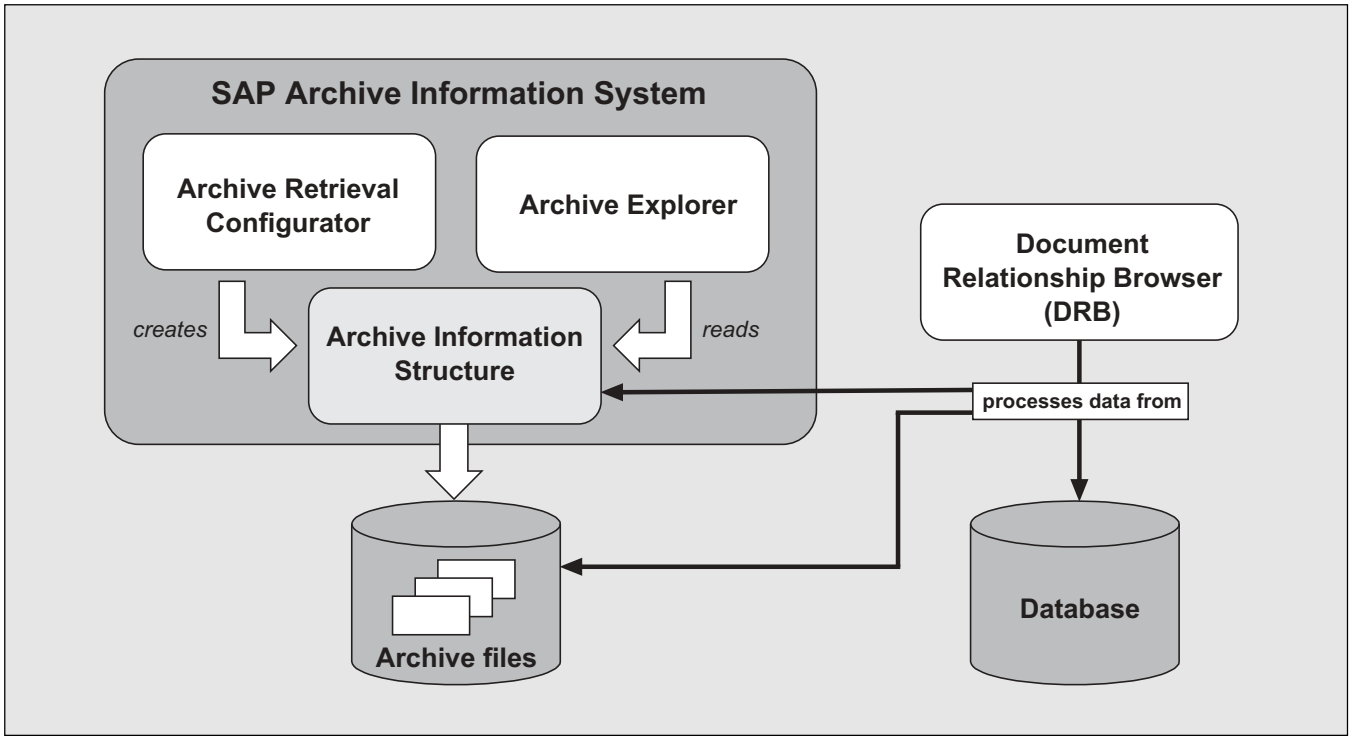
The fields in an archive information structure are taken from a standard *field catalog* provided by SAP. Such a field catalog is a collection of all possible data fields that are specific to a particular business document, such as a sales document. For instance, a sales document can contain the field's material number, order quantity, net value, and so on.

[13] In the first instance, only the information structure is generated. Then, when the user activates it, the database table and the reporting program are created.

[14] For more information on ADK, refer to our previous article in the March/April 2001 issue of this publication.

[15] The archive key contains the following information: number of the archiving session, number of the archive file for that session, and the name of the archiving object used.

Figure 10 Components of the Archive Information System



> ✓ ***Tip***
>
> *When setting up an archive information structure, do not include more fields than necessary.*[16] *Information structures containing a large number of fields grow rapidly when they are filled with data. Also, do not create more information structures than you actually need. Data archiving typically deals with mass data, so when setting up an archive information structure you should be aware that it will eventually be used to store such data. Therefore, having several large information structures set up and filled with data may have a serious impact on the database size, since information structures are kept in the database. In a worst-case scenario, by storing large information structures, you could end up consuming more space in the database than you previously gained by archiving.*

[16] For example, the standard field catalog for sales documents contains 69 different fields, most of which you hardly need. So there is no point in inserting such a large number of fields in an information structure.

The selected fields can originate from one or several source tables. The field catalogs supplied with AS in the standard system contain the fields from the underlying database tables that are normally important for the business context used. Because of this, there is usually no need to create customer-specific field catalogs. Also, creating new field catalogs requires extensive knowledge of the data model underlying the business objects used. Field catalogs are initially provided by SAP. They represent a superset of fields that can be used for information structures.

The AS Components

The Archive Information System consists of two major components:

- Archive Retrieval Configurator
- Archive Explorer

Figure 10 shows how these two components

operate on the archive files when an archive information structure is created or evaluated. The diagram also shows the three data sources that DRB can potentially access.

Archive Retrieval Configurator

This is the central part of AS. It can be accessed using the "Customizing" button on the initial screen ("Archive Information System: Central management") of AS. In the first instance, it enables administrators to set up new archive information structures and associate them with the archiving objects for which users need to retrieve archived data. Using the Archive Retrieval Configurator, you can also change the organization of existing information structures, create new field catalogs,[17] and activate or deactivate information structures. Deactivating an information structure means that this information structure will not be filled automatically during the next delete phase. You can, however, fill it manually at a later stage.

The fields ARCHIVEKEY and ARCHIVEOFS are implicitly generated upon the creation of an archive information structure. When activating an archive information structure, the relevant reporting program is also created automatically, along with the programs for filling and emptying information structures.

Archive Explorer

The main task of the Archive Explorer is to enable the retrieval of archived data objects. It does this by reading the archive information structures that have been created and filled with data for a specific archiving object. Generally, the data can be displayed in two different ways:

- In generic or technical views (similar to transaction SE16)
- In application-specific views (the data is displayed in the same way that data from the online database is viewed)

While the technical view is always available, the availability of application-specific views depends on the archiving object. SAP has provided application-specific views for the most commonly used archiving objects. More archiving objects will be covered in future releases.

Runtime Environment

Due to the tight integration of AS in data archiving, the AS and ADK runtime environments are inextricably linked. ADK is always aware of any action that is performed in AS.

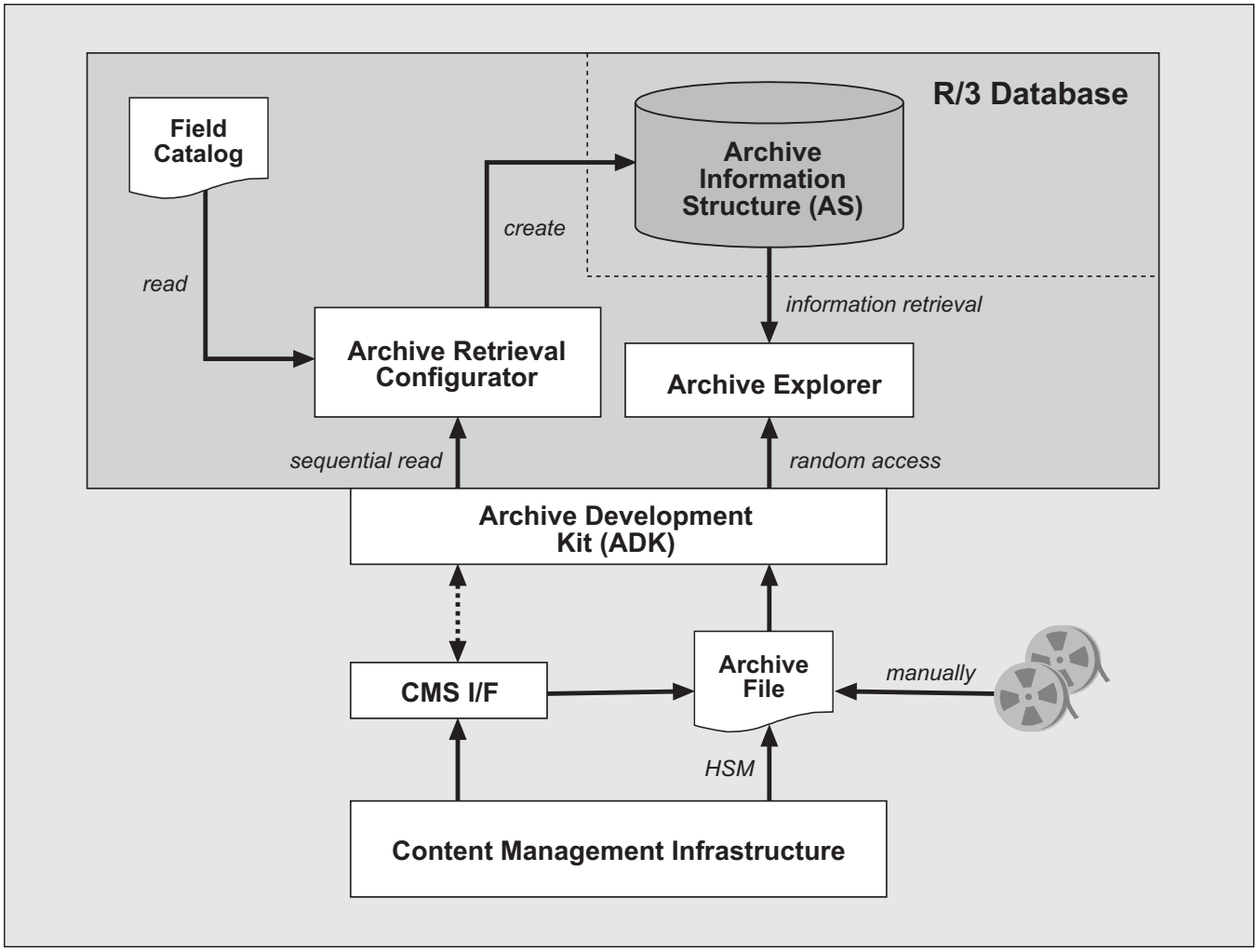
The primary task of the AS runtime environment is to support the interaction between the components involved, and to make sure that AS-specific tasks, including the following, can be carried out:

- Automatic filling of active information structures during a data archiving delete run[18]
- Mapping the fields in an archive file to the fields in an information structure
- Activating and deactivating information structures
- Storing and updating the status information

[17] Although AS offers this possibility, it is best to use only the standard field catalogs supplied by SAP. While these field catalogs must not be changed, they can be used as a template for new field catalogs. Please note that creating your own field catalogs requires extensive knowledge of the data model underlying the business object. You also should be familiar with the structure of the archiving object used.

[18] The filling of archive information structures can also be triggered explicitly by the user.

Figure 11 Integration of AS in the Data Archiving Environment



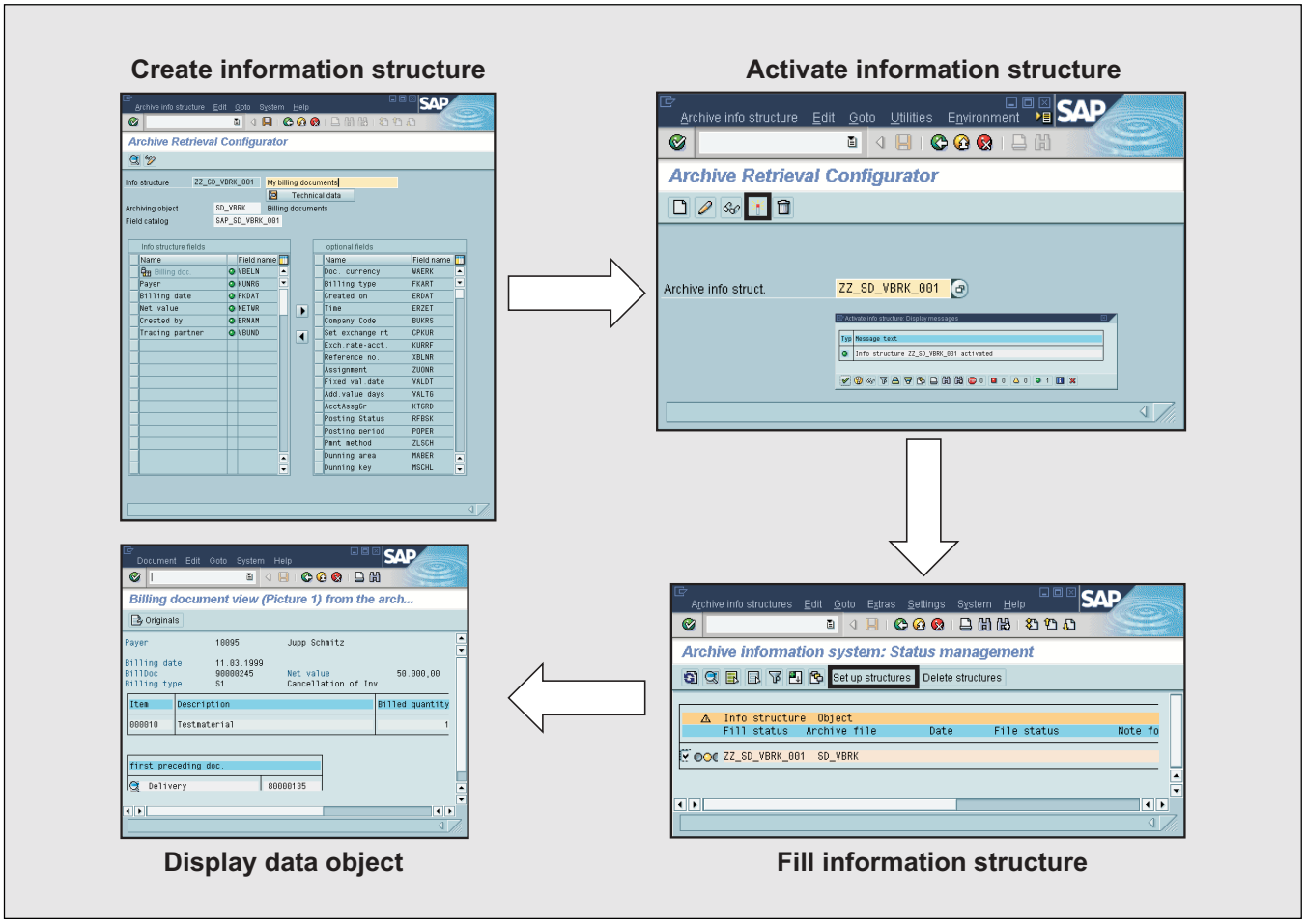
In **Figure 11**, you can see how the AS components are embedded in the data archiving environment, and how the components interact with each other. The shaded box at the top contains the main components of the AS solution.

At the core of the entire AS concept is the *archive information structure*. It constitutes the basis for all the activities with regard to the evaluation of archived data using AS. Information structures are set up by the *Archive Retrieval Configurator* using predefined fields from application-specific field catalogs. The *Archive Explorer* has two main tasks: selecting the data from the information structure according to the selection criteria entered by the user, and displaying it in either a technical or business view.

Using the Archive Information System

Let's look now at the general process of evaluating archived information using the Archive Information

Figure 12 Overview of the Archive Information System Processes



System (AS).[19] **Figure 12** shows an overview of the archive information system processes.

Creating the Archive Information Structure

First, the user should check whether an archive information structure that meets the requirements already exists. If not, he or she can create a new information structure.[20] When creating a new archive information structure, the user needs to specify the name of the archiving object for which the information is intended, and also the name of a suitable field catalog. The user then adds the required data fields to the information structure by choosing them from a list of optional fields in the field catalog.

Activating the Archive Information Structure

Once the information structure has been created, it needs to be activated. Only active information structures are taken into account during filling. After activation, no more changes can be made to an information structure. However, information structures can be deactivated at any time, thus ensuring that they will be ignored in the next delete phase.

[19] What we provide here is but a brief overview of AS. For more information on using AS, refer to the application help (**Help** → **Application Help**), or download the relevant documentation in PDF format from the SAPSERV3 server (see SAP note 99388).

[20] You only need to do this if you feel that one of the standard information structures provided with AS does not suit your needs — for example, because it does not contain the fields you require. Alternatively, you can choose a standard information structure.

> ✓ ***Tip***
>
> *If you make changes to an (inactive) information structure — for example, by inserting new fields — be aware that you might also have to regenerate the underlying database table. Before doing this, you must first delete the data in it. Finally, you refill the information structure itself. Deleting and regenerating a database table involves several database activities, such as delete and insert, which are quite expensive in terms of performance. It is therefore not advisable to carry out this procedure on a regular basis, but rather in an emergency. Instead, we recommend that when creating new information structures, you ascertain thoroughly the fields that you will eventually need, and try not to change the information structure later. In this way, you avoid having to regenerate the database table.*

Filling the Information Structure

In order to report an information structure, it must be filled with data from archive files. This can happen either automatically during a delete run, or manually, initiated by the user.

> ✓ ***Tip***
>
> *AS also supports the selective filling of information structures. This means that you can select individual archive files from an archiving session to be processed. So if you know that older data objects are only viewed very rarely, you can delete the corresponding entries by processing the appropriate archive sessions. This is particularly important if the information structure is especially large.*

An information structure can only be filled from archive files that have been processed by the data archiving delete program. During deletion, ADK passes all data records found in the archive file to the AS interface. According to the definition of the information structure, AS filters the relevant information from the data records and places it into the corresponding fields of the information structure — or, in more precise terms, into a transparent database table together with an access key that points to the data objects in the archive file. This table is the basis for future reporting runs.

Displaying the Archived Data Objects

Displaying archived information is accomplished by means of the Archive Explorer tool. The user needs to specify the archiving object and the information structure from which he or she wants to report. In the selection screen, he or she can restrict the search to specific parameters,[21] such as the document number or the date of creation, if necessary. The result of the search is presented in a list. Detailed information on an individual data object can then be obtained by double-clicking the corresponding list entry and selecting the desired view in which the data is to be displayed. This can be either a technically specific view or an application-specific view. The scope of the available views differs considerably and is dependent on the archiving object used.

> ✓ ***Tip***
>
> *A special way of retrieving archived data with AS is by using the "Ad-hoc Reporting" function. This enables you to report directly from the archive, thus bypassing the database table. Although this kind of reporting still requires an information structure, there is no need to fill the information structure beforehand. This is particularly useful for test purposes — for example, for reporting archived data that has not yet been deleted from the database. This process eases the pressure on the database, as the contents of an information structure do not have to be stored in a database table. However, the process can be time consuming, so make sure the sizes of the data objects you are going to report from are reasonably small when using this function.*

[21] This includes all fields contained in the archive information structure except the fields MANDT, ARCHIVEKEY, and ARCHIVEOFS.

> ✓ ***Tip***
>
> *AS does not automatically update information structures that contain archived data that was later reloaded. In this case, AS will mark the display status of these information structures as incorrect. You will then need to carry out a manual update of the information structures. Proceed as follows: In the "Status per Archive" view of AS Status Management, mark all archiving sessions whose files were reloaded (these sessions are marked by a red traffic light), and empty them. After completing this step, mark the archiving sessions that were created by the reload, and fill the information structures once again.*

> ✓ ***Tip***
>
> *If you need to evaluate application data, always try to do this before the data is archived. Although some applications provide combined evaluation programs for processing online data and archived data, in most cases only a restricted set of data display features can be used. Also, since the read program needs to retrieve the data from the archive, the process may take longer, depending on the amount of data processed. Particularly, if you are using the Data Retention Tool (DART) or the SAP Business Information Warehouse (SAP BW), we advise you to load these systems while the data is still in the online database.*

Developing User-Specific Retrieval Solutions

Since data archiving selects and stores the data according to application-specific criteria, the read programs required to view the archived data are also application-specific. This is the reason there is no generic read program that can be used to access data from various applications, such as SD, FI, or HR. So it is entirely within the responsibility of each application to provide a read program for a particular application and to determine how the data is actually displayed. Some applications, such as FI, provide combined read programs that are able to access online data and archived data alike, and display it within the same list. However, only a few applications have such a feature at their disposal. Usually, the applications offer two types of read programs: one for online data, and another for archived data (if any).

The information and tips that we have collected in the following sections may be helpful to you if you need to write your own retrieval solution, such as a read program or a specific business view for AS. We have tried to describe the most important relationships, be they technical or business, that exist between the components involved to increase your understanding and to bring you up to speed very quickly when implementing a retrieval solution.

To access archived data from an application, the system needs to know the exact file name and path name of the archive file. This information is stored in the database. However, the system only stores a logical representation of the file name and path name rather than their physical descriptions. When accessing the archive file for reading, the program uses the logical path name to determine the actual, physical path under which the archive file is stored by evaluating the corresponding Customizing entries (you can make these entries using either transaction FILE or transaction SF01).

> ✓ ***Tip***
>
> *A logical file name can be created either as client-specific (using transaction SF01) or as cross-client (using transaction FILE). Remember that a client-specific file name always overrides a cross-client file name, so make sure you delete any unnecessary or obsolete client-specific definitions in all clients affected.*
>
> *Also note that if there are two identical file names that were created in both transaction FILE and transaction SF01, the physical destination associated with them may still be different. As the client-specific definition always has priority over the cross-client definition, a read program will try to find the archive files under the path specified in transaction SF01.*

> ✓ ***Tip***
>
> *What happens if the archive files are migrated to a different storage medium (due to hardware reorganization or an upgrade)? In such a case, ADK will not be able to access these files anymore because it is not aware of the change. To make these files available again for archiving programs, you need to update the physical path by assigning the logical path to the physical path using transaction FILE.*

Along with the file and path name, the database also keeps a record of whether the archive file has been transferred to an external Content Management System, and, if so, also retains a pointer to the storage location. Furthermore, the database also stores the following administration data for each archive file:

- Date of creation
- Processing status (complete, incomplete, canceled, etc.)
- Number of data objects contained

The data itself is accessed via a standardized interface that provides suitable methods for the evaluation programs.

ADK stores data in a way that enables read access at any time. Usually, two forms of access and display are used:

- Sequential reading
- Direct access

Let's take a closer look at these two methods of data access.

Accessing Archived Data — Sequential Reading

This method is the simplest form of access to archived data. It is mainly used to analyze archived datasets and is available for most archiving objects.

By using a sequential read program, you can display in list form all selected data objects, for example, for a specific posting period or for a document number range. There is a direct, linear relationship between the time behavior of the read program and the size of the archive file to be read.

The read process comprises the following steps:

- **Reading the data:** The read program first positions the file pointer at the start of the archive file and reads the metadata. After the metadata has been successfully read, the program continues reading the data object by moving the file pointer bit by bit (see also "Appendix A — The Structure of an Archive File").

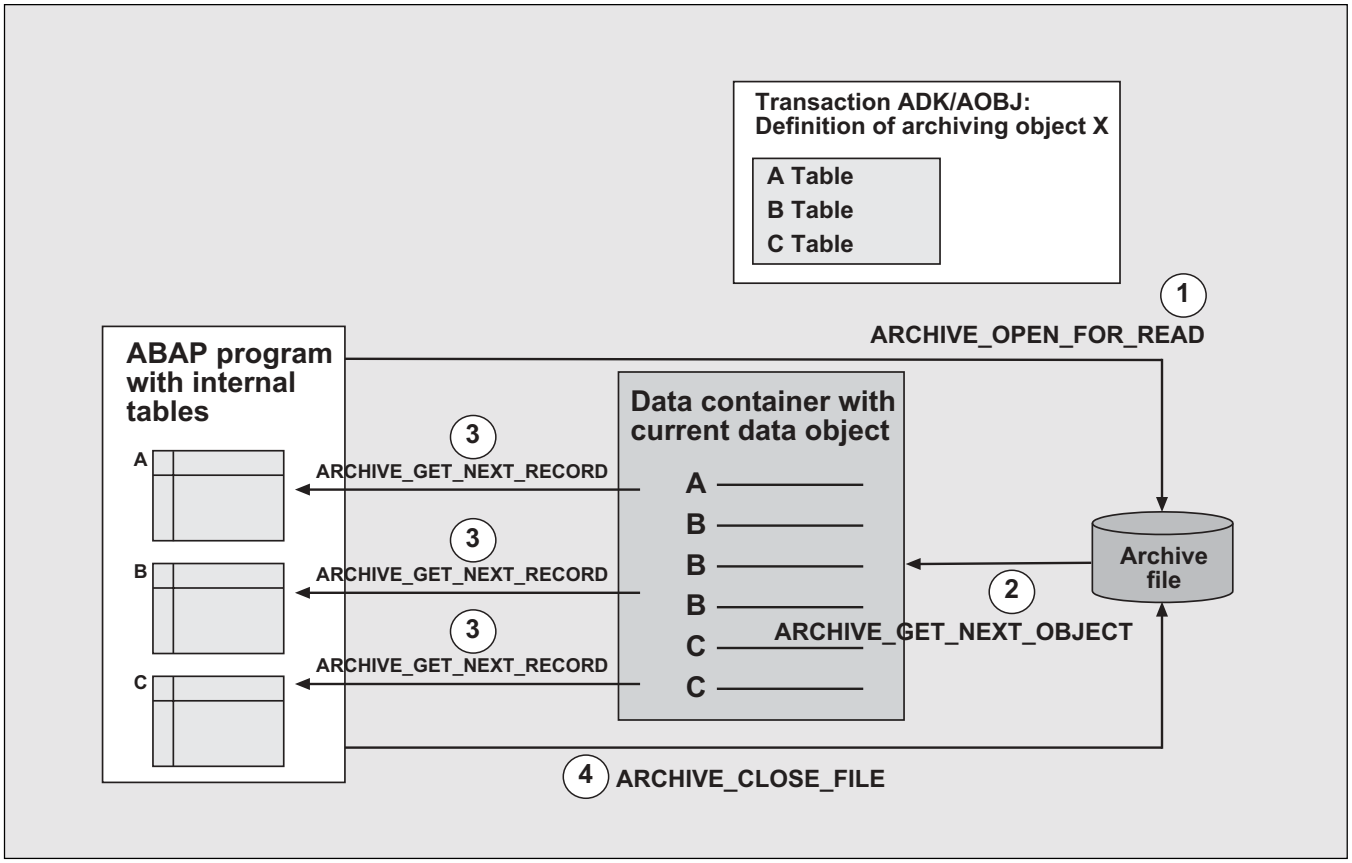
- **Transferring the data:** The program then passes the read bitstream over to the archive interface (ADK), which decompresses it and stores it in structured work areas (data containers). These data containers are then passed over to the evaluation program (written in ABAP) that reads the data and prepares it for display or evaluation.

The read process is complete when the pointer reaches the logical EOF characterized by the hexadecimal string "FFFF."

The display program decompresses the data and displays it in line with the selection criteria used. The type and number of selection criteria depends on the application. How the data is displayed depends on the display function that was used for the archiving object. The display is not the same for all archiving objects.

An authorization strategy ensures that archived data that is later read by a data archiving read program is protected in much the same way that access to application data is protected. Various authorization tests take place in both the sequential read program and when the user enters the Archive Administration transaction (SARA). The central authorization object for archiving activities, S_ARCHIVE, is also used for sequential access.

Figure 13 The Flow Logic of Sequential Read Access



The performance of a sequential read program mainly depends on how the data to be displayed is selected. If you do not make any entries in the selection screen (which means that all data objects are to be displayed), the read program must read all the data in the selected archiving sessions before the data objects can be displayed. This leads to a serious diminution in the performance of the read program. For faster access to archived data objects, we recommend that you use the direct access function.

Structure of a Sequential Read Program

Figure 13 depicts the flow logic of sequential data access. It is assumed that the physical table design of the corresponding archiving object comprises the tables A, B, and C. The process is divided into the following steps:

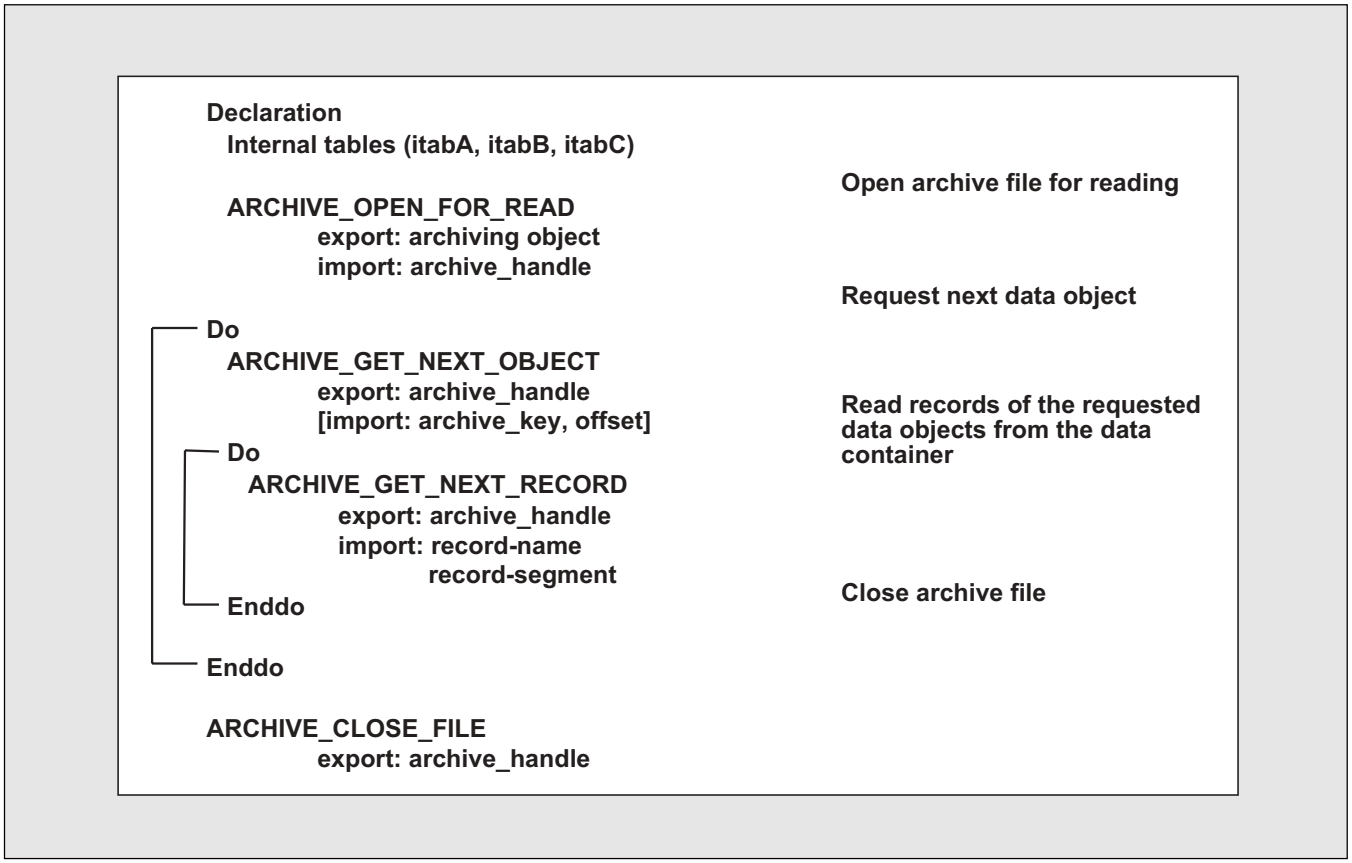
1. First, the program opens the corresponding archive file in read mode by calling the ADK function module ARCHIVE_OPEN_FOR_READ.

2. In the next step, the program loops over the data objects, and sequentially requests the data objects until the loop ends. The data objects are then placed into a data container that also contains the name of the corresponding database table from where the data originates.

   The data container itself is implemented as an internal table with a standard structure, which is the same for all archiving objects.

3. The program then sequentially reads the records from the container, and places them — according to the original tables they were taken from — into the internal memory of the relevant read

Figure 14 The Logic of a Sequential Read Program



programs. When all records have been processed, the read process for this data object is complete.

4. When all data objects have been evaluated, the archive file is closed.

For full details on the individual steps, including the function modules involved, refer to the next section.

But what does this flow logic look like in terms of ABAP coding? The sample in **Figure 14** shows how a read program is generally implemented using standard ABAP routines. First, the internal tables[22] (itabA, itabB, itabC) for storing the records to be processed by the read program are declared. Then, the relevant ADK function modules are called.

[22] These are actually the ABAP memory areas, which have the same structure as the underlying database tables.

Next, we are going to take a closer look at the most important function modules used by sequential read programs, and how they interact with each other. You can, if required, use these function modules to develop customized read programs.

Open Archive File

The ARCHIVE_OPEN_FOR_READ function module opens one or several archive files for reading. The system then displays a list of existing archive files from which you can make your selection. The names of the selected archive files are taken as import parameters, and stored in the table parameter "selected_files." The function module returns a so-called *archive handle* that points to the selected archive files. All selected archive files can be opened using the same handle.

ARCHIVE_OPEN_FOR_READ is also responsible for interpreting the metadata that was stored with the archive file when it was created.

The API function modules of ADK are multi-instance-enabled. Each individual instance is addressed by a single handle.[23] In a single program run, several archive files (even if they belong to different archiving objects) can be written or read simultaneously. ADK uses the handle to distinguish between the various files.

Function modules accessing an archive in read mode treat all opened files that belong to this handle as a single file. Every additional call of this function module creates a new handle. This enables the simultaneous processing of several archive files from different archiving objects. The following is an extract from the relevant program code:

```
CALL FUNCTION 'ARCHIVE_OPEN_FOR_READ'
  EXPORTING
     OBJECT               = ARCHIVING_OBJECT
  IMPORTING
     ARCHIVE_HANDLE       = HANDLE
  TABLES
     SELECTED_FILES       = SELECTED_FILES
  EXCEPTIONS
     FILE_ALREADY_OPEN    = 1
     FILE_IO_ERROR        = 2
     INTERNAL_ERROR       = 3
     NO_FILES_AVAILABLE   = 4
     OBJECT_NOT_FOUND     = 5
     OPEN_ERROR           = 6
     NOT_AUTHORIZED       = 7
     OTHERS               = 8.
```

Request Data Object

The ARCHIVE_GET_NEXT_OBJECT function module requests the next data object for a handle and places it in the data container. The program identifies the archive file it needs to search by reading the handle that was created by the ARCHIVE_OPEN_FOR_READ function module. ARCHIVE_GET_NEXT_OBJECT continues reading until all data objects of this archive file have been processed. In the case that the read includes several different archive files, the function module still continues reading until all archive files that are addressed by this handle are processed.

Read Data from a Data Object

By calling the ARCHIVE_GET_NEXT_RECORD function module, the records from the previously called data object are read from the data container. This is performed in an inner loop. The first call of ARCHIVE_GET_NEXT_OBJECT automatically returns the first record, and then continues sequentially until all records from the data object have been processed.

Close Archive File

The ARCHIVE_CLOSE_FILE function module closes all archive files that were previously opened for the handle. After this, all resources that were allocated for the handle are released. The handle becomes invalid and must no longer be used. Any attempt to further use this handle raises the exception WRONG_ACCESS_TO_ARCHIVE.

Accessing Archived Data — Direct Access

Direct access (also known as single document or random access) to archived data objects, such as a sales order or an invoice, can only be carried out with the aid of an index table.[24] This is needed to assign and store attributes for individual data objects. The data object to be displayed is first restricted to search

[23] The handle used by ADK is similar to a file handle that file management systems usually use.

[24] In this context, this table should rather be referred to as a pointer table, since it has nothing to do with an index of a database table. An archive information structure is an example of a pointer table.

criteria within the index table. If this is successful, the archive file containing the relevant data object is located using the index (archive information structure), and opened. The read program is now able to access and display the data object. Direct access of this kind requires extensive programming, and is therefore only available for a few archiving objects, such as FI_DOCUMNT.

To locate the data object, the read program needs the exact starting position of the data object within the archive file. The read program retrieves this information from an index table, such as an information structure, that is stored in the database. Additionally, it always reads the metadata from each archive file. The read process is complete when the start of the next data object (marked by the control string "%START") is reached (see also "Appendix A — The Structure of an Archive File"). The bitstream is then transferred to the archive interface, which decompresses it and passes the data container over to the read program.

The Archive Information System offers a considerably more comprehensive, and easier-to-use function for swift, direct access to data objects. This function was discussed in more detail earlier.

In addition to reading and displaying archived data objects, most read programs are also able to include in the display independent data that is not part of an archiving object. This is the case, for example, for scanned original documents, CAD drawings, work items, IDocs, and e-mails that were assigned to a data object. These documents are usually necessary to reconstruct the original process context in which the data objects came about. This includes links to independent data objects, such as the preceding or subsequent document within a process chain. The scope of the function available in this context depends on the archiving object.

Read Program: Direct Access

Direct access can only be carried out for existing information structures that are already filled. This means that all relevant information must be available, so that the program can use it for searching. Direct access also requires the name and path of the archive file, as well as the exact location of the data object within the archive file. This only works, however, if the corresponding archive information structures have been filled. First, the relevant information needs to be determined from the information structure. This task can be carried out by the function module AS_API_READ, which has the following interface:

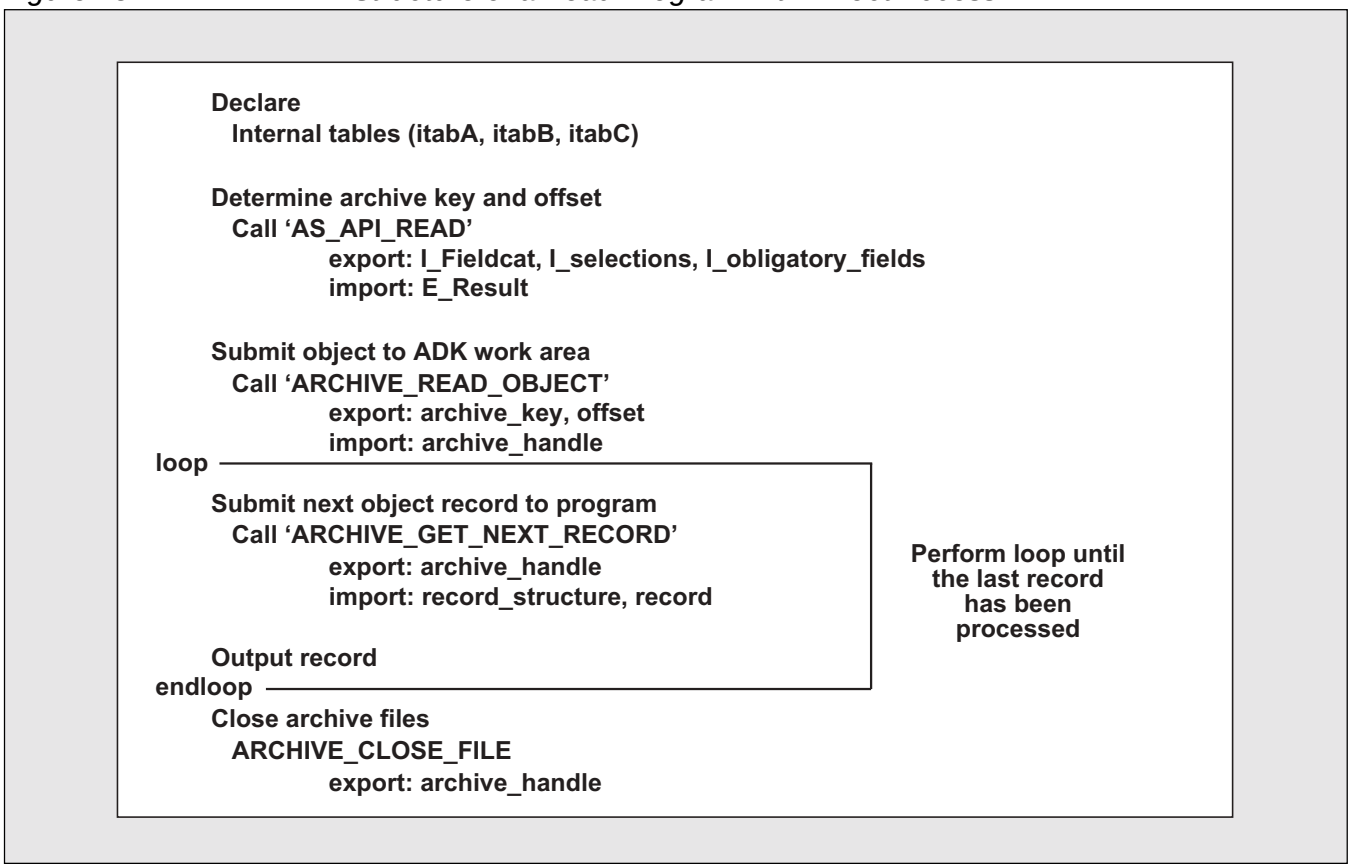
```
CALL FUNCTION 'AS_API_READ'
     EXPORTING
          I_FIELDCAT          =
          I_SELECTIONS        =
          I_OBLIGATORY_FIELDS =
     IMPORTING
          E_RESULT            =
     EXCEPTIONS
          PARAMETERS_INVALID  = 1
          NO_INFOSTRUC_FOUND  = 2
          OTHERS              = 3
```

The input parameters used for this function call have the following meanings:

- I_FIELDCAT: Name of the field catalog
- I_SELECTIONS: Restricts the data selection
- I_OBLIGATORY_FIELDS: Field names that are to appear in the information structure
- E_RESULT: Hit rate based on the above input parameters. The hit rate implicitly incorporates the name of the archive file and the location of the data object within the archive file.

This information is processed when the function module ARCHIVE_READ_OBJECT is called. The function module reads a particular data object, belonging to an archiving object, from an archive file, and returns a handle for further processing. If the archive file was stored using the ArchiveLink interface, this function module can also be used to read data objects contained in the archive file. The function module is capable of reading the data object, regardless of whether the archive file is stored in the file system or via ArchiveLink.

Figure 15 Structure of a Read Program with Direct Access



> ✓ ***Tip***
>
> *In a standard information structure, only some important fields, such as the document number or the item number, are flagged as primary key fields. If, in your selection screen, you choose fields other than those belonging to the primary key, the system will not find a suitable database index. This results in a full table scan across the entire information structure. To optimize the access time, you should either add these fields to the primary key, or create a secondary index (using transaction SE11) that contains the fields in question.*

The function module AS_API_READ returns a hit list to the calling program (according to the input parameters I_SELECTION) through the internal table E_RESULT. This table contains the logical archive key, which identifies the archive file, and also the offset, which determines the exact position of the data object within the archive file. After the data object has been read by ARCHIVE_READ_OBJECT, you can use ARCHIVE_GET_NEXT_RECORD to access the data by evaluating the handle.

Figure 15 contains a commented extract from the relevant coding.

Developing Business Views

If there is no standard business view available for a particular archiving object, you can implement a customized business view. The key element for this task is the table AIND_STR5.

The general procedure for displaying a data object using AS is as follows: The system evaluates the entries (names of the business views) made in this table, and prompts the user to select an option. The system then processes the business view that the user has selected.

In order to implement a customized business view, including its integration in AS, you must make appropriate entries in the table AIND_STR5. You need to specify the name of the archiving object and the name of the function module used. You can do this in transaction SE16. The function module must possess a standard interface with the same logic as the sample interface shown in **Listing 1**.

The field I_ARCHIVEKEY contains the name of the archiving object, whereas the field I_OFFSET specifies the location of the data object within the archive file. By default, the field content is established from the archive information structure. When AS is called, the fields are automatically supplied with the relevant values.

Using this information, the previously mentioned function module can now read the contents of the data object. The data must then be prepared for output and displayed onscreen. This task is carried out by the Archive Explorer, which provides at least a technical view for displaying the data. In addition, it is also able to display the data in business-specific views, provided these have been implemented for the archiving object for which the search is carried out.

Listing 2 contains a commented extract from the coding for displaying sales orders. In addition to the

Listing 1: Sample Function Module Interface

```
FUNCTION SD_VBAK_01_DISPLAY_DATA.
*"----------------------------------------------------------------------
*"*"Global Interface:
*"       IMPORTING
*"             REFERENCE(I_ARCHIVEKEY) LIKE  ADMI_FILES-ARCHIV_KEY
*"             REFERENCE(I_OFFSET) LIKE  ARCH_IDX-OFFSET
*"       EXCEPTIONS
*"              OBJECT_NOT_FOUND
*"----------------------------------------------------------------------
```

Listing 2: Extract of Sample Coding for Displaying Sales Orders

```
1  FUNCTION SD_VBAK_01_DISPLAY_DATA.
2  *"----------------------------------------------------------------------
3  *"*"Global Interface:
4  *"       IMPORTING
5  *"             REFERENCE(I_ARCHIVEKEY) LIKE  ADMI_FILES-ARCHIV_KEY
6  *"                                OPTIONAL
7  *"             REFERENCE(I_OFFSET) LIKE  ARCH_IDX-OFFSET
```

(continued on next page)

(continued from previous page)

```
*"       EXCEPTIONS
*"             OBJECT_NOT_FOUND
*"----------------------------------------------------------------------
*.Refresh in internal tables
  CLEAR XVBFA.  REFRESH XVBFA.
  CLEAR XVGBEA. REFRESH XVGBEA.

*.get object
  CALL FUNCTION 'ARCHIVE_READ_OBJECT'
       EXPORTING
            OBJECT         = 'SD_VBAK'
            ARCHIVKEY      = I_ARCHIVEKEY
            OFFSET         = I_OFFSET
            OBJECT_ID      = ' '
       IMPORTING
            ARCHIVE_HANDLE = HANDLE
       EXCEPTIONS
            OTHERS         = 1.

  IF SY-SUBRC <> 0.

    MESSAGE ID SY-MSGID TYPE 'S' NUMBER SY-MSGNO WITH
                SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
    EXIT.
  ELSE.

    DO.
*.get records
      CALL FUNCTION 'ARCHIVE_GET_NEXT_RECORD'
           EXPORTING
                ARCHIVE_HANDLE   = HANDLE
           IMPORTING
                RECORD           = DATA
                RECORD_STRUCTURE = STRUCTURE
           EXCEPTIONS
                END_OF_OBJECT    = 1.

      IF SY-SUBRC <> 0.
        EXIT.
      ENDIF.

*.move records
      CASE STRUCTURE.
        WHEN 'VBAK'. XVBAK = data.
```



```
52         when 'VBAP'. xvbap = data. append XVBAP.
53         when 'VBKD'. xvbkd = data. append XVBkd.
54         WHEN 'VBPA'. XVBPA = DATA. APPEND XVBPA.
55         WHEN 'VBFA'. XVBFA = DATA. APPEND XVBFA.
56         WHEN 'SADR'. XSADR = DATA. APPEND XSADR.
57       ENDCASE.
58
59     ENDDO.
60
61     CALL FUNCTION 'ARCHIVE_CLOSE_FILE'
62          EXPORTING
63               ARCHIVE_HANDLE = HANDLE.
64   ENDIF.
65
66 *. Authority-check
67   AUTHORITY-CHECK OBJECT 'V_VBAK_AAT'
68       ID 'AUART' FIELD XVBAK-AUART
69       ID 'ACTVT' FIELD ACTVT_03.
70
71   IF SY-SUBRC <> 0.
72     MESSAGE I501.
73     EXIT.
74   ENDIF.
75
76
77 * authorization for sales organization distribution channel division
78   AUTHORITY-CHECK OBJECT 'V_VBAK_VKO'
79        ID 'VKORG' FIELD XVBAK-VKORG
80        ID 'VTWEG' FIELD XVBAK-VTWEG
81        ID 'SPART' FIELD XVBAK-SPART
82        ID 'ACTVT' FIELD ACTVT_03.
83
IF SY-SUBRC <> 0.
85     MESSAGE I501.
86     EXIT.
87   ENDIF.
88
89 *.control switching in the business view
90   VIEW = 'SD_VBAK_BWL'.
91
92 *.read textpool
93   PROGRAM = 'S3VBAKAU'.
94   READ TEXTPOOL PROGRAM INTO TEXTP LANGUAGE SY-LANGU.
95
```

(continued on next page)

(continued from previous page)

```
96 *.display data
97
98   SCREEN_CNT = SCREEN_CNT + 1.
99   SET TITLEBAR '000' WITH SCREEN_CNT.
100        CONDENSE SY-TITLE.
101
102        CALL SCREEN 100.
103        SCREEN_CNT = SCREEN_CNT - 1.
104
105      ENDFUNCTION.
```

embedded comments, you can find more comments below the sample coding. The name of this function module is stored in table AIND_STR5.

In the interface, the function module is handed the archive key and the offset. ADK automatically fills these parameters with values after the user has selected a sales order in AS (lines 5-7). Based on this information, the sales order can then be read from the archive file. This is done by calling the function module ARCHIVE_READ_OBJECT, which contains the archive key and the offset as input parameters.

This function module returns a handle that is used to request the individual data records via the archive interface. This is accomplished in a DO/ENDDO loop (lines 34-59) by calling the function module ARCHIVE_GET_NEXT_RECORD. The data records returned by this function module are stored in the internal tables XVBAP, XVBKD, XVBPA, XVBFA, and XSADR.

When all the records have been read, the DO loop is terminated and the archive file is closed by calling the function module ARCHIVE_CLOSE_FILE (the handle becomes invalid).

Finally, the data is displayed to the user by calling screen 100 (line 102). The actual formatting of the data occurs in this screen in the PBO (Process Before Output) phase.

Conclusion

Archived data can originate from a large number of different sources. As such, there are lots of differences between the individual data objects, which means there can be no one-size-fits-all tool or method to facilitate all retrieval requirements. Because of this, there are multiple tools available — such as the Document Relationship Browser (DRB) and the Archive Information System (AS) — each of which is a specialist in its own particular area. Depending on the task, the user can choose the appropriate tool. For standard retrieval tasks, no further programming is required. Users can get started right away.

In certain cases, administrators and developers will need to carry out some minor integration or programming work. The reason is that there is no such thing as an off-the-shelf solution that covers all aspects, particularly if user-defined data objects are involved.

We hope that we have been able to demonstrate that the process isn't difficult. We've tried to design things so that it is easy to add value to the existing retrieval tools and adapt them to the specific needs of your users.

What's In Store?

What improvements in data archiving functionality can you expect for upcoming releases of mySAP.com solutions? The following outlines some of the recent projects that SAP development is currently engaged in:

✓ In **mySAP CRM**,[25] data archiving will be considerably enhanced and adapted to the specific needs in a CRM environment. For example, a new status concept will be introduced that basically works as follows: A CRM data object, such as a contact or an order, can have either the status "archivable" or the status "deletable," as established by several application-specific checks. The ADK write program selects all data objects having the status "archivable" and writes them to an archive file while setting their status to "deletable." Then, in contrast to standard data archiving in SAP R/3 (where the ADK delete program is responsible for deleting the archived data objects from the database), all data objects having the status "deletable" are deleted by a dedicated CRM delete program running at regular intervals.

✓ The planned enhancements to the **Archive Information System** include more application-specific views, more (and updated) field catalogs, integration in application functions, and enhanced integration of archiving classes. Pressure on the database could also be eased if information structures could be stored in a medium outside the database where they can be easily accessed, for example, by using a specific hash algorithm. SAP AS development is currently examining different approaches to this promising concept and is confident that it will implement a comprehensive solution in the near future.

✓ In **Document Relationship Browser** development, one of the major tasks to be tackled soon is implementing this useful evaluation tool within a mySAP CRM environment. DRB development is also committed to integrating additional business processes and expanding existing ones in order to extend continuously the range of DRB.

---

[25] Customer Relationship Management is the application area covering every aspect of the interaction between a company and its customers. This includes the more traditional sales and distribution functionalities, as well as newer concepts like Business-to-Business Procurement, Internet-enabled information provisioning for potential customers, e-Commerce, etc.

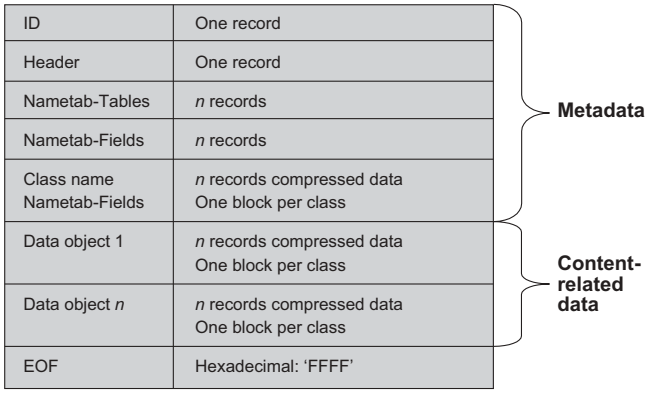
Dr. Rolf Gersbacher joined SAP in 1994, where he was engaged in the implementation of business process reengineering projects using workflow technology. Since 1998 he has been a member of SAP's Performance, Benchmark and Data Archiving (PBA) department, where he is responsible for conducting extensive analysis and has co-written projects in the area of data archiving. Together with Helmut Stefani, Rolf developed the best practices guide "Managing SAP Data Archiving Projects," which has become the vade mecum *for data archiving issues. He can be reached at rolf.gersbacher@sap.com.*

Helmut Stefani joined SAP in 1997 as a member of the Data Archiving Coordination team (now a part of the Performance, Benchmark and Data Archiving, or PBA, department), which coordinates the development of archiving solutions across applications. He holds responsibility for information development in this area, and is co-author of the best practices guide "Managing SAP Data Archiving Projects." Prior to joining SAP, Helmut was a software localization specialist at Computervision (now PTC), a major producer of CAD/CAM software. He can be reached at helmut.stefani@sap.com.

Appendix A — The Structure of an Archive File

To better understand how the various tools for accessing archived data operate, it is useful to know something about the underlying structure of all archive files. The format of the data file is unique to all archiving objects and is SAP proprietary. For data security reasons, any archive file that was created by data archiving can be interpreted only via the Archive Development Kit (ADK). No other tool or program is able to read such an archive file. The interface, however, has been published so that anyone who wants to develop their own read programs can use it.

All archive files are built according to the same structure, which generally consists of two parts (see the diagram below): metadata and content-related data.



The information contained in the metadata is used to describe the actual data objects. When the archive file is created, additional information is written to this section to describe the data model of the underlying archiving object.

At the start of the metadata section, in the *ID* segment, the program stores information on the system environment that was current when the archive file was created. This includes the hardware, the code page, and the number format. The read program needs this information to rebuild the data object when it is viewed in a different system environment, for instance, after a hardware upgrade.

The *Header* segment contains additional information on the SAP R/3 system environment, such as the R/3 release and the system name. The segments *Nametab-Tables* and *Nametab-Fields* are about the physical design of the database tables. This includes the definition of the SAP tables from the ABAP Dictionary, and the definitions of the underlying fields. Information on additional dependent data objects, such as SAPscript texts or change documents that were also written to the archive, are stored in the segments *Classname* and *Nametab-Fields*.

The data that was actually archived is stored in the next data section beginning with *Data Object 1*. This contains the complete set of data that was transferred from the data object to the archive file, including SAPscript texts and change documents. When the size of the archive file has reached its limit, or when the maximum number of data objects is reached — both of which are defined in Customizing — the write program marks the logical EOF in the archive file by inserting the hexadecimal value "FFFF."

Appendix B — Integrating Customer-Specific Tables in the Archive Information System

Along with the standard SAP tables, which contain the application data, many customers also use tables of their own, usually referred to as "Z tables" or "customer-specific tables." Customer-specific tables must comply with SAP standards; for example, they must be created in the customer namespace rather than the SAP namespace. Objects that are created in the customer namespace are named with a character string that usually begins with "Z," hence the name "Z table."

But why should you ever need to create customer-specific tables? This need only arises if you need to expand standard business objects, such as sales documents, by adding additional information that is not contained in the business object as delivered by SAP. When the business object for which you have created a customer-specific table is archived, the archiving program also considers the contents of this table (provided you have made the table available to the corresponding archiving object). So archiving customer-specific tables is not a problem. However, when you evaluate the archived business object using the Archive Information System (AS), the information will not be considered unless you have taken certain steps, which are discussed below.

The solution to this problem is best illustrated by means of an example. Let's assume that you have created the customer-specific table Z_VBAK_INFO. It contains additional information for sales orders that is specific to your company. When you create a new sales order, SD will automatically enter the relevant data in this table. Technically speaking, this is accomplished by a temporary modification of the standard SAP program for sales orders. The primary key of the table Z_VBAK_INFO is composed of the primary key of the leading table (SD_VBAK), its own key field, plus any additional data fields (which are not key fields). For example, this may appear as follows:

1. VBELN
2. Z_KEYFIELD
3. Data fields

To be able to evaluate the information contained in the table Z_VBAK_INFO using AS, you need to create an information structure that incorporates both the fields from the standard SAP tables VBAK (header data) and VBAP (items) as well as the fields from the table Z_VBAK_INFO.[1] You can only add new fields to an information structure by choosing them from an existing field catalog. If they are not contained in the field catalog, you must first add them to the field catalog. Since modifying an information

[1] A general prerequisite to this is that the customer-specific table is embedded in a hierarchical relationship with the relevant standard SAP tables, in that its key represents a subset, or extension (albeit an improper one) of the standard SAP table.

structure is fairly straightforward and well documented in the AS online help, we would just like to concentrate on the first part — that is, preparing the field catalog.

Depending on how the primary key of the customer-specific table is used, two different scenarios can be distinguished:

- **Scenario 1:** The primary key is an extension of a standard SAP table (which is already known to the field catalog).

- **Scenario 2:** The primary key does not appear in any standard SAP table (which is already known to the field catalog).

For **Scenario 1**, we suggest the following general procedure:

1. Copy the corresponding standard field catalog to a new field catalog in the customer namespace.

2. In the field selection screen, select the source fields belonging to the relevant table (in our example, VBAK-VBELN).

3. Enter the customer-specific table (Z_VBAK_INFO) using the function *Other source fields*.

4. Add the relevant fields to your field catalog. When creating your information structure, these fields will then also be available to you, along with the fields from the standard tables.

Scenario 2 is potentially too complex to be dealt with in detail here, so we can only provide you with a rough outline of the procedure:

1. First, you need to create a new field catalog (for the relevant archiving object) that contains the customer-specific table only.

2. You then create a new information structure based on this field catalog.

A drawback to this solution is, of course, that you cannot run a combined evaluation of SAP standard tables and the customer-specific table[2] using AS.

[2] If this scenario is specific to you, we advise you to seek professional advice from SAP Consulting.

