SAPinsider

This article appeared in the Jul▪Aug▪Sep 2006 issue of *SAP Insider* and appears here with permission from the publisher, Wellesley Information Services (WIS), **www.WISpubs.com**.

WIS

Data Archiving — The Fastest Access to Your Business Data?

Regular Feature

Performance & Data Management Corner

Dr. Bernhard Brinkmöller, SAP AG

Helmut Stefani, SAP AG

Data archiving extracts business-complete data from the database and writes it to the file system, helping to limit data volume and greatly reduce a system's overall TCO. But many companies are still apprehensive about data archiving projects due to two common misconceptions, both of which reveal some understandable concerns about the speed of data access:

1. There's a common fear that once data has been archived, it's as good as gone since it takes so long to access. Companies believe that, as a rule, accessing business data in the online database is much faster than accessing the same data from the archive.
2. IT teams think they can improve the performance of online accesses by simply relocating archive indexes, such as archive information structures, from the database to the file system, believing that any reduction of the database size will automatically improve performance.

In this column, we'll take a closer look at these archiving myths to see why data archiving not only enables ready access to archived data, it promotes *better performance than the actual database*. We will demonstrate why accessing a single document from the archive — a sales order, for example (see sidebar) — is usually faster than accessing it from a database, and we'll debunk the idea that relocating archive indexes to the file system boosts performance. We'll also see how new partitioning of infostructures functionality in SAP NetWeaver 2004s supports more cost-efficient data storage.

With a closer look into SAP's data archiving practices, you'll have a better understanding of why SAP promotes an object-oriented archiving approach and how you can use archiving to enhance access to data and documents, reducing your infrastructure costs and resource usage at the same time.

How Object-Oriented Storage Improves the Performance of Single Document Accesses

To better understand the advantages of data archiving, you first need to consider the two methods of data

Be Mindful of the Enterprise Data Life Cycle:

Why Data's Age Determines How It Is Accessed and How It Should Be Archived

As we examine these archiving myths, it's important to keep the typical life cycle of enterprise data in mind. Let's take a closer look at how users interact with data throughout its "career." Once data is generated, you can think of its life cycle as having three phases (see **Figure 1** on the next page):

Phase 1: Frequent User Accesses

In the first phase of the data life cycle, shortly after the data has been created, the probability that frequent accesses will occur on this data is highest. New or recently created data is included in current business processes, which makes it subject to frequent user accesses — sales order information, for example, is frequently accessed to create invoices or deliveries.

There are typically two types of user accesses: **single document accesses** and **evaluations**. These two types differ considerably in terms of how data is

Sidebar continued on next page

storage: **relational** storage and **object-oriented** storage.

In a *relational* database management system (RDBMS), data is stored in related tables. This means that data pertaining to a specific business object, such as a sales order or a billing document, usually spreads across several database tables. The sales order, for example, can include up to 150 different tables. Fortunately not all of these tables are filled or read for any given sales order instance.

The knowledge about the related tables is stored within the business object definition and, from an archiving perspective, also in the archiving object definition. Whenever a document is being processed, displayed, or changed, the database needs to access the majority of the associated tables in order to extract the entire set of data that makes up the document. Depending on the complexity of the document — that is, the number of database tables that need to be accessed — this process can be quite time-consuming.

SAP data archiving changes the way the data is stored from relational to *object-oriented* storage (see **Figure 2**). With object-oriented storage, all the data that belongs to a particular document is stored in a "flat," object-like format. As a result, there are some notable differences when you access the data. Archived single documents can generally be displayed faster than documents from the database because the data is available in a single location and does not need to be collected from several tables. This primary benefit of data archiving may be diminished by performance drawbacks if a slow storage system is used.

On the other hand, there are a few disadvantages when you evaluate and report on archived data, since data archiving is not designed for this type of aggregated data access. A potential decrease in accessibility is accepted, since analyzing business-complete processes means that these evaluations are rarely the focus of daily work.

This is where SAP NetWeaver Business Intelligence (SAP NetWeaver BI) comes in as a specially developed

Sidebar continued from previous page

Be Mindful of the Enterprise Data Life Cycle

accessed and displayed. When carrying out a single document access — for example, when you want to change the quantity of items on a sales order — the performing program usually reads most of the relevant tables that contain entries belonging to the document. Then, the program displays only the one document, showing detailed information.

For evaluations, however, things are quite different. Say that you want to create a work list that includes all sales orders of customers from a specific region. To search by region, you would likely use the zip code as a search criterion. To deliver your work list, the evaluation program sequentially reads all of the documents that are relevant to the selection criteria and then displays them in a condensed format — as a list, for example. In most cases the display does not include the entire set of document fields as with single document accesses, but only a selection of these fields. With evaluations, we do not only refer to classic business evaluations, such as the Compact Document Journal (report RFBELJ00), but to all sequential accesses designed to access and process data.

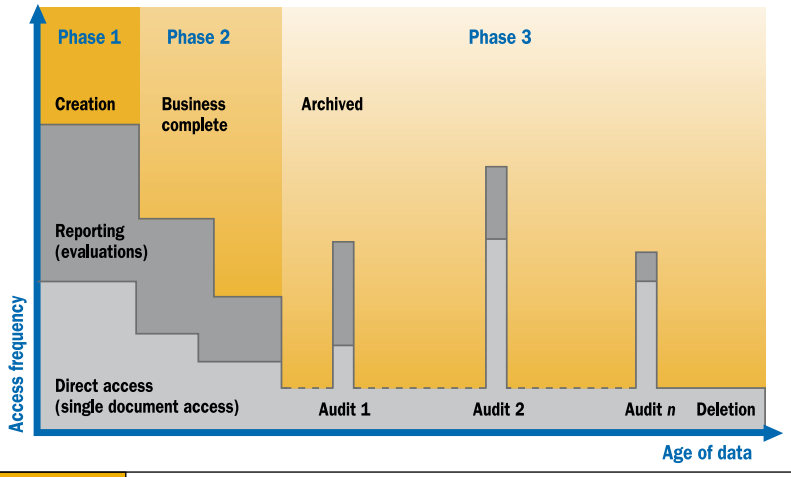

Figure 1 Data's Age Determines Its Access Frequency and Access Type

Whether a document is mainly accessed using the single document or the evaluation option strongly depends on the document type and whether the document is relevant for a particular evaluation. The evaluations occurring in this frequent access phase are very different from those in later phases of the data life cycle. Usually, every evaluation requires a special index so that a considerable number of secondary indexes — indexes that are created in addition to the primary index — are in the database. But

Sidebar continued on next page

solution for the drill-down reporting of historical data. We recommend using SAP NetWeaver BI for all aggregated reports and using the archive in the original system to drill down into single documents.

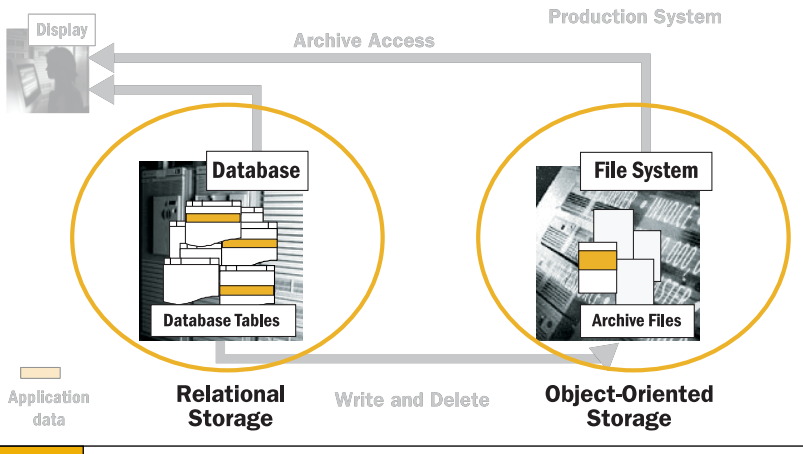

Figure 2 Comparing Relational and Object-Oriented Storage of Data

Indexes for Optimized Archive Access

Object-oriented storage can facilitate fast access on single documents only if the accessing program "knows" exactly where data resides in the archive. This is achieved using the same resource that is used for online data: *indexes*. An archive index is a table with attributes for searching for an archived document, and is a pointer to the document itself.

In SAP systems, archive index tables are stored in the database and are administered by the Archive Information System (AS). The amount of index data stored in these tables has to be subtracted from the amount of data freed from the database through

Sidebar continued from previous page

the databases are optimally designed to meet the challenges of this first phase. The ratio between the amount of index data and the actual business data is generally about 40:60.

Phase 2: Trend Analysis and Specific Queries

But what if the data has aged considerably and is no longer needed in current business processes? In this second phase of the data life cycle, evaluations such as creating sales order work lists are no longer necessary. The indexes in the database are now only used to quickly establish whether or not the data is still subject to such evaluations. The accesses performed here are mainly single document accesses.

For example, say that you want to run an evaluation to find out how many products or materials of a certain type were ordered in a particular month. To carry out such an evaluation efficiently, the database needs an index of items from the pertinent sales orders. Only data from business-complete documents is subject to this evaluation — all data still in use in the system is not eligible here.

Old data must of course still be accessed for carrying out such a trend analysis, but this is usually performed on condensed or aggregated data that is stored in a separate business information warehouse, such as SAP NetWeaver Business Intelligence.

Phase 3: Internal or External Audits

Even after a long time, you may need to access old data in the original system. This most commonly happens during internal or external audits — for compliance reporting, for example. Although they represent the largest part of accesses, single document accesses are not the only accesses that are performed in this phase. Quite frequently, evaluations on the original data are required because the aggregated data in the business warehouse does not meet the requirements of an audit.

This means not only a temporary increase in the number of accesses on old data, but also a larger number of evaluations. These, however, are not as varied as the evaluations from the first phase.

How the Data Life Cycle Influences Your Archiving Strategy

To summarize, as data gets older, both the frequency of data accesses and the type of these accesses change:

- **Access frequency** — The older the data gets, the less frequently it is accessed.
- **Access type** — When old data is accessed, it will most often be on a single document access basis.

A good archiving strategy will always take these facts into consideration. In light of this changing access behavior, you can determine if a relational storage approach — the approach often used if the archiving is tightly linked to the database — is still the most appropriate strategy for data archiving, especially given the single document access behavior that is typical for older data.

archiving, making data archiving somewhat less effective. However, since archived data requires only a reduced set of access options, an archive index generally occupies less space in the database than the sum of indexes in the relational tables of the document. Instead of a 40% share in the total data volume (as with normal indexes), you can assume that the value for archive indexes is somewhere in the single digits.

But since data resides considerably longer in the archive than in the database (in the context of the data's entire life cycle), the AS tables can become very large over time — these tables are often among the largest tables in the system. Therefore it might seem like a good idea to relocate these tables, which are only needed online in rare cases, out of the database. The benefits appear obvious: This would help replace expensive storage space in the database with more economic storage space in the file system, and would benefit performance just like archiving data from the original tables does.

So why doesn't SAP follow this approach and move archive indexes out of the database?

Enhanced Performance by Keeping Archiving Indexes in the Database

As we demonstrated in a previous *Performance & Data Management Corner* column,[1] old data in the database does not impair performance if:

- It is not mixed with new data within the same data block
- It is not accidentally read by wrong "where" clauses

To meet these criteria, you do not need to relocate data from the database. It is sufficient to separate the old and new data — and this is already done for you when the data in the AS tables remains in the database.

Consequently, relocating archive indexes from the database cannot enhance the performance of accesses to the remaining online data. Instead, relocating the index data would *worsen* the performance of archive accesses. To fight this performance deterioration, you would have to develop your own optimizer, a sophisticated database mechanism that finds the optimal access path for various searches, and store the index data redundantly.

Support for Complex Structures and High-Performance Searches: Real-World Example

The problem becomes more transparent if, in addition to direct, single-document access on the archive through a document number, you also need to do some evaluations for internal and external audits. Let's look at an example from an SAP customer.

During a presentation at a recent DSAG conference, a customer showed an AS table with nine fields that she had created for archived financial documents that needed to be reviewed for compliance purposes. Nine seems to be a small number compared to the total number of fields in the original tables, but it already provides a great variety of search options. To carry out a high-performance search in AS, several secondary indexes must be created to support the search for these fields.

This is not a problem as long as the table resides in the database, since a high-performance search requires the use of an optimizer. This method makes sure that the interface remains very lean, even if a complex archive search is carried out on this rather broad infostructure, or database index table.

To obtain the same functionality with an infostructure that is stored outside the database, you would have to create a huge number of disjoint structures. The index maintenance would also get more complicated, and you would have to develop your own optimizer for the archive access.

In summary, the benefits of keeping infostructures in the database include:

- **You can create secondary indexes** — Secondary indexes facilitate document searches by means of adding selection fields. But if you keep your index tables outside the database, you do not have this option.
- **You'll get optimizer support** — To benefit from the full functionality and speed of an optimizer during a search, the index tables must reside in the database.

Cost-Efficient Storage of Archive Indexes in the Database

We've argued that if you want to improve the performance of archive accesses, you are well advised to leave the archive indexes in the database. But you still might claim that the price for this performance increase is much too high compared to the inexpensive storage of archive indexes in the file system. After all, the main purpose of data archiving is reducing administration and storage costs.

In addition to the optimizer, the database also includes mechanisms designed to guarantee transactional security and to enable data backups. For archive indexes, however, these backups would be overdimensioned. That is, if archive indexes are treated just like any other data in the database, they will lead to additional costs — for example, through multiple mirroring or by extending backup times — costs that far outweigh the real value of the archive indexes.

However, it is not always necessary to completely remove data from the database to achieve cost reductions. If your database works with table spaces, you can create special table spaces for

[1] Please see the article "Data Archiving Improves Performance — Myth or Reality?" in the October-December 2005 issue of *SAP Insider* (www.SAPinsider.com).

archive indexes. We recommend creating these table spaces on disks that are not subject to the same expensive mirroring methods used for online data and that do not take part in backups.

Using this approach, you can obtain the same cost reductions as if the data were moved to the file system, but without the drawbacks of not having optimizer-based access. Depending on the options offered by the database, additional optimization potential can be tapped. For example, using the *Row Compression* option offered by the IBM DB2/UDB database system may prove beneficial for certain infostructures.

New Ways SAP Enables Reduced-Cost Index Storage

For the reasons explained above, SAP has deliberately decided against providing options for relocating infostructures to the file system as a remedy against growing index tables. Instead, SAP has decided to continuously improve the handling of infostructures in the database. The **partitioning of infostructures** can be regarded as a first step in this direction.

This partitioning feature, which was introduced with SAP NetWeaver 2004s, enables you to distribute the data of an infostructure across several tables, reducing the number of problems that can occur when handling very large tables. You can also assign your own names to the individual tables, which facilitates the assignment to dedicated table spaces. The tables you specify are generated in the development or customizing system and are then transported to the production system.

In the past, infostructure tables were only created in the production system. Thus, system administrators had far fewer options to influence the properties of these tables.

With this new development, SAP has set the standard for enabling cost-efficient storage of infostructures in the database. At SAP, we are convinced that by optimizing the storage of index tables in the database, customers can obtain much larger improvements in performance and cost reduction than they could by relocating these infostructures to the file system.

Conclusion

Despite popular belief, archived data is not lost, nor is it difficult to access. If you optimize your archive storage toward the object-oriented accesses that are typical for old data and refrain from moving your archive infostructures to the file system, you can actually improve access times and overall performance — and, in turn, TCO.

For more information on SAP's data archiving strategy with respect to performance, please visit **http://service.sap.com/~sapidb/011000358700000507038 2005E**. ■

Dr. Bernhard Brinkmöller studied physics at the University of Münster, Germany, and went on to graduate from the University of Bonn, Germany. He then worked for the University of Minnesota, US, and the University of Karlsruhe, Germany, with research placements at the Los Alamos National Lab and the Paul Scherrer Institute, Switzerland. He joined SAP AG in 1994, where he worked as a performance specialist until he took over the position of Development Manager for Data Archiving in 2000.

Helmut Stefani studied applied linguistics in Mainz/Germersheim, Germany, at the Johannes Gutenberg University, where he graduated as a technical translator for English and Spanish. He has been a member of the Performance, Data Management, and Scalability team at SAP since 1997, where he works in the documentation, product management, and rollout of data archiving topics. Helmut has jointly authored several publications, including a book, on data management and data archiving.