SAPinsider This article appeared in the Oct▪Nov▪Dec 2005 issue of *SAP Insider* and appears here with permission from the publisher, Wellesley Information Services (WIS), **www.WISpubs.com**.

Data Archiving Improves Performance — Myth or Reality?

Regular Feature

Performance and Data Management Corner

Dr. Bernhard Brinkmöller, SAP AG   Georg Fischer, SAP AG

One of the strengths of SAP applications is scalability, with a three-tier client/server environment that takes much of the load away from the database. Still, for many companies, the database remains the weakest part of their IT system. If you have stretched your database's capabilities to their limits, and need ways to manage the ever-expanding amount of data generated by your enterprise applications, you have two options: expand your systems, or eliminate old data by deleting or archiving it. Many companies have realized that the key to managing rampant data growth is data archiving — it's a less risky option than erratically deleting old data, and a better long-term solution than simply expanding your systems.

Data archiving reduces storage needs by taking data no longer accessed in everyday business processes, removing it from the database, and writing it to archive files; later the archive files can be moved to less expensive media like tape or CD, if required. If initiated during go-live and performed regularly, data archiving will have dramatic effects on controlling the growth of a company's data and consequently lowering the total cost of ownership (TCO) of its systems.

But in addition to these space or cost-related arguments, companies often look to their archiving projects for another benefit: improving system performance. In fact, companies often make the mistake of entering an archiving project assuming that these goals — reducing storage costs, freeing up data space, and improving performance — can all be achieved by the same means (that is, archiving the largest files). More and more archiving projects are moving forward with the objective of increasing system performance, but with the practice of analyzing for space — a mindset that is incorrect. Some projects successfully reduce the amount of data in the system, only to find that performance remains unaffected. This article will help explain why, and where to target your archiving project resources with performance in mind.

Revising Assumptions About Data Archiving and Performance

Understandably, many customers come to the conclusion that an archiving project that reduces the amount of data in the system will automatically improve performance, and that the relationship between saved disk space and performance is more or less proportional. In other words, they initiate the project believing that if the data volume is reduced by a certain amount, the performance will automatically improve by the same degree.

With a superficial look at your system, these assumptions may appear to be supported. For example, in an initial analysis based on data in a customer system, we monitored the growth of the table MSEG, which holds line items of material documents, and measured how long it took to display a material document in transaction MB51. Transaction MB51 displays records of material movement — for example, if you're a telephone manufacturer, a record is written (and stored in table MSEG) whenever you move a telephone from point A to point B, and you can call that record through transaction MB51.

The results of our analysis (shown in **Figure 1** on the next page) indicate a clear relationship between an increase in table size and a slowdown in response times — as more telephones are moved, more data is created, and system response time slows down. On the other hand, you will not see any degradation of performance for your everyday logistics execution processes, including recording of new material documents, with the increasing size of this table. So the relationship is more complex than it first appears. When you consider the type of data that really deserves your attention — by analyzing the layout of your indexes and the nature of system queries, as well as the other factors that can affect performance — you'll have a fuller picture of archiving's true affect on the performance of your systems.

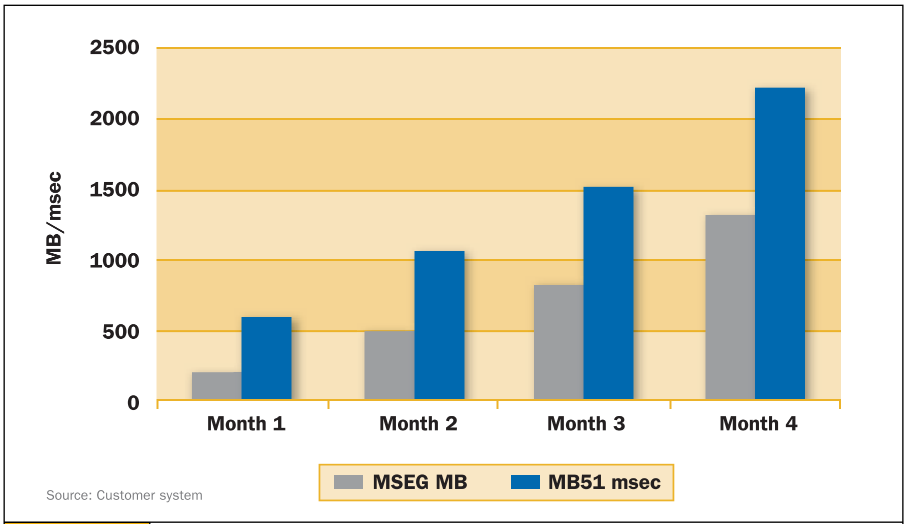


Figure 1 Relationship Between Response Time for Transaction MB51 and the Size of Table MSEG

A Closer Look at System Performance

The term *performance* denotes the capability of a data processing system to meet predefined standards with respect to system efficiency. More precisely, performance characterizes the *response time behavior* of the system during user dialog or background processing.[1] A large number of individual factors — such as average central processing unit (CPU) time, average wait time, average roll wait time, database request time, the I/O rate, and many more — are key performance indicators (KPIs) contributing to response time behavior.

Of the major performance KPIs, mainly the database request time and the I/O rate will be affected by an increase in data volume. The I/O rate is the number of data blocks read from the hard disk in a given time period. Data blocks are sequences of continuous data characters or bytes transmitted as a unit and stored in memory for possible further use. This applies to tables, where the data is stored, as well as to indexes, which are used to access the data in the tables.

The I/O rate is determined by the amount of data that needs to be processed and the quality of the database buffer. The quality is considered good if a minimum of 95 percent of all blocks accessed by queries are found in memory (random access memory, RAM). Inversely, this means that a maximum of five percent of all blocks accessed by queries are allowed not to be in memory prior to the execution of the query. Assuming that three to four index blocks are needed for accessing one table block, about 100 percent of accessed index blocks and 80 percent of tables should be in memory in order to reach the average of 95 percent hit rate in memory.

[1] Response time is the time between initiating a request to the system and the system's response, either showing the requested data or acknowledging the completion of a requested task.

Are You Focused on the Wrong Data?

To understand the role of data archiving for performance, we concentrate on the I/O rate of the database (see sidebar "A Closer Look at System Performance"), and we make two assumptions:

1. Only data that is no longer needed for business purposes is eligible for archiving. SAP's archiving programs ensure that only business-complete data is archived. However, you must examine your reporting habits to determine the retention period for business-complete data in the database, based on your company's needs.
2. Only data that is needed for a certain task is read from the database. Therefore, old data should never be transferred to the application.

Based on these assumptions, and supposing an appropriate index layout, we know that data blocks containing exclusively old data will not be touched in everyday operation. Therefore, these blocks cannot negatively impact the I/O rate. Only if the data block contains a mixture of old and new data is there a chance of the old data being transferred into the memory, where it will take away memory space without ever being requested by the application. A mixture of new and old data can occur in both table and index blocks (see sidebar "Where to Find Mixed Data").

"Old" data should never be transferred to the application, so it cannot influence any performance KPI that is related to the application server and frontend.

From here, we will concentrate on the mixture of old and new data *in indexes*. Because indexes are created in every system as soon as the first bits of data are inserted, index layout is key to performance. The type of index used for accessing data influences the I/O rate; this is one of the most important aspects to consider when determining which data must be archived first to obtain the largest performance benefit.

Where to Find Mixed Data

To achieve the highest performance improvement from your archiving projects, you must first look to data blocks that contain a mixture of new and old data. Archiving blocks of old data, which are never requested by the application, will do nothing more than create additional space in the database. So where can you find these old-and-new-data mixtures?

- **Table blocks** — During the insertion of new data, if table blocks that are not completely empty are being reused, a mixture of old and new data will occur. This table fragmentation can have a significant influence on the overall performance impact of data archiving. Note, however, that if neither data archiving nor deletion has ever occurred in a system, there is no way that table blocks can be fragmented, so performance considerations do not apply.
- **Index blocks** — Here, the position in the index and therefore the data block into which a new entry is inserted depends on the order of the indexed fields. A mixture of new and old data will occur if the index is not chronologically sorted.

Which Data Should You Archive First to Achieve the Highest Performance Improvement?

Index data can be sorted in two ways: chronologically or non-chronologically. In a *chronologically sorted index*, data is organized in reference to a time factor. This does not necessarily mean a specific date or time. An index that is organized according to document numbers, for example, is a chronologically sorted index, provided the numbers for new documents ascend over time. In these indexes, old and new data do not reside in close proximity to each other. Therefore, it is unlikely that old and new data reside in one data block.

Under our earlier assumption that only new (non-archivable) data is used in queries, we can be sure that only a small part of all index blocks, containing the most current data, will ever be read into the database memory. All the archivable data will be in data blocks that are accessed only in rare cases. Archiving such data will free space on the disks of the database, but it will not influence performance.

Non-chronologically sorted indexes are organized in reference to a factor other than time — for example, IDs that are generated more or less at random, such as certain types of Global Unique Identifiers (GUIDs), or IDs using some concatenated indexes, which use semantics in addition to time.

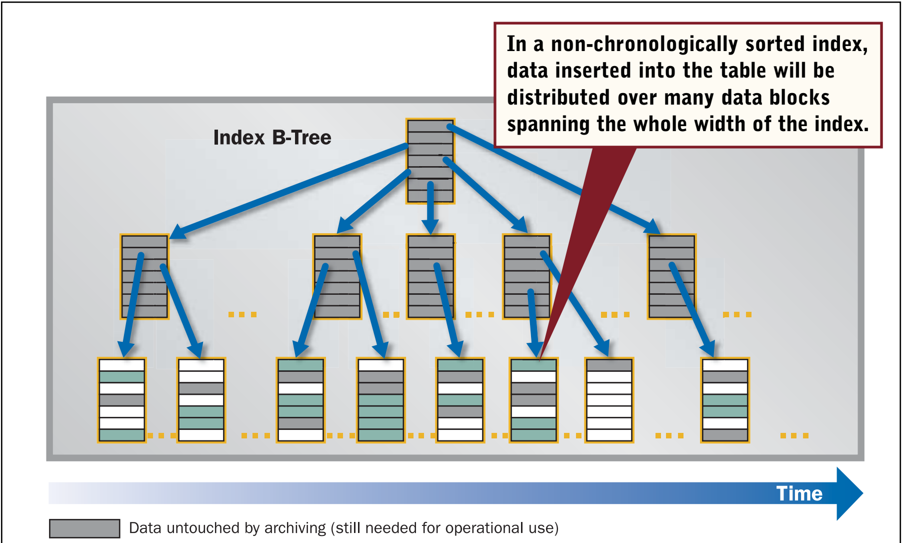
In a non-chronologically sorted index, data that is inserted into the table at a given time will be distributed over many data blocks spanning the whole width of the index. The most recently inserted records that are needed for everyday business are distributed equally over the full index width. As more and more data is added to the system, the index width increases, and the efficiency — or buffer quality — for query accesses goes down, leading to a decrease in performance. In this case, data archiving will help to stop the growth and — if followed by an index reorganization — reduce the size of the index, which in turn will stop performance degradation or even improve performance (see **Figure 2** on the next page).

Simply stated, when looking to gain the most performance improvement in your data archiving project, look first to the statements that produce high I/O rates for your database. If those statements are supported by a non-chronological index, then data archiving is the only means to reduce the I/O rate and therefore improve performance.

Returning to Our Material Documents Example

Let's recall the example of transaction MB51 discussed earlier to understand why archiving data, which is accessed using a non-chronologically sorted index, and sound index layout are so important for system performance. Imagine that your company transports a truckload of telephones to a local retailer, and you want to know whether certain red phones were on the truck. Since the movement of telephones is recorded in table MSEG, MB51 will simply query MSEG using the document number associated with the truckload to make the determination. If, however, you wanted to see how many red phones were transported this year, the query becomes a bit more complicated. Information about the red phones is located in MSEG, but the information of the transport day is written in MKPF — the header table of the material document. Now, the database optimizer has to decide where to start looking. Because the number of different material types is more selective than the number of years for which material documents are stored, it makes sense to first locate all documents that show the movement of red phones, and then restrict the result to phones moved this year.

To do this, the database would first read the index on the material number for selecting data from MSEG, which is not chronologically sorted. After it located all MSEG records for red phones (including those of previous years) it would use the material document number to join the header data from table MKPF


In a non-chronologically sorted index, data inserted into the table will be distributed over many data blocks spanning the whole width of the index.
Index B-Tree
Time

▨ Data untouched by archiving (still needed for operational use)
□ Free blocks remaining after data is archived and deleted
🟩 New data inserted into free blocks, which opened up after archiving and deletion

Figure 2 Data Distribution in Non-Chronologically Sorted Indexes After Archiving

to restrict the result to movement from the desired time period.

In this example, data blocks from both the full width of the index and from the full width of the MSEG table are read. However, only with the join to table MKPF can the old blocks be filtered out and not be transferred to the application. Because relevant, time-specific information is located in table MKPF, you cannot create a chronologically sorted index for MSEG. So, in this case, there is no chance to improve the performance by optimizing the index design.

In our example, data archiving would keep the data volume in table MSEG small, ensuring an acceptable performance level when users access material documents via the material number (searching for phone records). Archiving the material documents will have no effect on performance, however, if you access the database via the document number that is ascending over time.

Lessons to Take Into Your Next Archiving Project

The overall influence of data archiving on performance strongly depends on the number of accesses performed on certain data, and on the ratio of chronologically sorted to non-chronologically sorted indexes. This ratio varies considerably between tables and between systems.

The largest impact on performance can be achieved by archiving data from tables that show a high I/O rate for accesses that do not use chronologically sorted indexes. If the overall buffer quality is at or below the desired 95 percent rate (see sidebar on page 100), and if the index layout has already been checked and optimized, data archiving should especially be considered. If your main focus is on performance, you should start archiving the tables with the accesses showing the highest I/O rate (not necessarily the largest tables in the system).

Summary

Data archiving can be very helpful in improving system performance — particularly in instances where you may not have considered archiving as a performance-improving method. You should always bear in mind, however, that data archiving is *not* a cure-all for maintaining optimal system performance. Data archiving does not improve system performance, for example, when performance lags are caused by an unrelated issue, such as not enough CPU power or memory on the application server.

Performance optimization is attainable, but only with a proper understanding of how data archiving works. Whenever performance is a key objective in an archiving project, we encourage you to examine both the layout of your indexes and the nature of system queries. Even in a well-tuned system with a good index layout, data archiving can enable you to tap further performance potentials that would otherwise lie fallow.

To learn more about the unique relationship between data archiving and system performance, please access the article "Performance Aspects of Data Archiving" at the SAP Service Marketplace (**http://service.sap.com/data-archiving** → *Media Library* → *Literature & Brochures*).

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 was employed as a performance specialist until he took over the position of Development Manager for Data Archiving in 2000.

Georg Fischer has been responsible for data archiving product management at SAP AG since 1998. Since 2003 he has been product manager for Performance, Data Management & Scalability at SAP. After studying information technology at the Darmstadt University of Technology, Germany, he worked at Digital Equipment Corporation (DEC) in document management, optical archiving, and data archiving. Before he switched to product management, Mr. Fischer worked for companies in Europe and the US as the project manager responsible for implementing archiving in the SAP environment.