Benchmarking SAP R/3 Archiving Scenarios*

Bernhard Zeller Alfons Kemper
Universität Passau
94030 Passau, Germany
<*last name*>@db.fmi.uni-passau.de

Abstract

According to a survey of the University of Berkeley [6], about 5 Exabytes of new information has been created in 2002. This information explosion affects also the database volumes of enterprise resource planning (ERP) systems like SAP R/3, the market leader for ERP systems. Just like the overall information explosion, the database volumes of ERP systems are growing at a tremendous rate and some of them have reached a size of several Terabytes. OLTP (Online Transaction Processing) databases of this size are hard to maintain and tend to perform poorly. One way to limit the size of a database is data staging, i.e., to make use of an SAP technique called archiving. That is, data which are not needed for every-day operations are demoted from the database (disks) to tertiary storage (tapes). In cooperation with our research group, SAP is adapting their archiving techniques to accelerate the archiving process by integrating new technologies like XML and advanced database features. However, so far no benchmark existed to evaluate different archiving scenarios and to measure the impact of a change in the archiving technique. We therefore designed and implemented a generic benchmark which is applicable to many different system layouts and allows the users to evaluate various archiving scenarios.

1. Introduction

When a company operates globally the database volumes of enterprise resource planning (ERP) systems like SAP R/3 are growing at a tremendous rate. Some of them have already reached a size of several Terabytes. OLTP (Online Transaction Processing) databases of this size are hard to maintain and tend to perform poorly. However, not all data stored in the tables of an SAP R/3 system is actually needed for every-day-business. Some of the data is outdated and rarely used because the corresponding business objects are no longer of interest (e.g., old orders). Therefore, SAP uses *archiving* which demotes rarely used data from the production database to tertiary storage (tapes) [9]. By moving rarely used data to tape archives the size of the tables in the production database is reduced to speed up mission-critical transactions and to alleviate the work of the database administrators.

So far, the SAP archiving process is not adequately supported by the commercial database systems. As one supporting technique, we showed in [5] how the bulk deletion of archived tuples from the production database system can be carried out efficiently. In [10] we described an XML generating archiving operator which can easily be integrated into most database systems. The operator can make use of the advanced XML processing features most database vendors provide today (see Section 4 for more details). However, so far no benchmark existed to evaluate these different archiving techniques and to compare the results for tuning and sizing purposes. We therefore implemented a generic benchmark which is applicable to various archiving scenarios and therefore allows the users to evaluate many different archiving scenarios. We integrated the benchmark in *SSQJ*, an SAP performance evaluation tool. In order to derive performance evaluations of practical relevance to the end users, the entire archiving process—i.e., the entire application system—is benchmarked. In [3] this aspect was already taken into account for decision support queries and a standard database benchmark was used to analyze the performance of database management systems as back-ends of an SAP R/3 system. Due to it's generality our benchmark is applicable to any archiving scenario which is supported by SAP R/3 and because of the integration into SSQJ the results are easily comparable.

The remainder of this paper is organized as follows: Section 2 gives a brief overview of the possible archiving scenarios. Our benchmark is described in Section 3. Section 4 shows how XML can be exploited for archiving purposes. Section 5 concludes the work.

2. Overview of Possible Archiving Scenarios

Archiving demotes rarely used data from the production database to tertiary storage (e.g., tapes) [9]. This is done by executing the following *archiving steps*: First, rarely used business objects are identified. Then the data of these business objects is copied to tertiary storage. Afterwards the

* This work was supported by an SAP research cooperation contract within the Terabyte-Project.

data of these business objects is deleted from the production database[1].

Archiving scenarios differ mainly in the following 3 dimensions: **When** are the different archiving steps executed—i.e., in parallel or serial? Which file **size** is used to store the archived data? What kind of **hardware media** is used to store the archived data? The settings in each dimension can be altered independently, resulting in many combinations.

When the archiving steps are executed in parallel, each record is copied to the tertiary storage and then deleted from the database. When the archiving steps are executed serially, first all records marked for archiving are copied to the tertiary storage. Afterwards they are deleted from the database in one batch. This way the expensive delete operations can be carried out during time periods with low database loads.

The archived data is stored in special *archive files*. In each archive file many objects are packed to save disk space but within one archive file only business objects of one kind are stored, e.g., financial data.

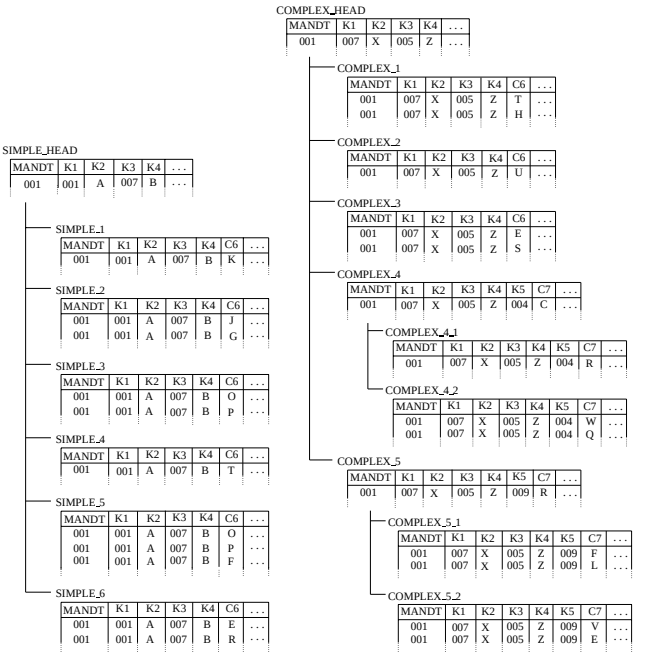
In cooperation with our University research group, there are developments under way at SAP to enable the users to store the archived data in other file formats like XML; but until now only a proprietary file format can be used in production SAP systems.

3. The Archiving Benchmark

To provide a generic and widely applicable benchmark we 'simulated' two different business objects by providing the necessary database tables and archiving programs. We further implemented a powerful data generation tool, which enables the users to generate business objects with different layouts and sizes. The following sections describe these components in detail and demonstrate their usage within the archiving benchmark.

3.1. Object Layouts and the Data Generation Tool

We implemented two artificial business objects called *SIMPLE* and *COMPLEX*. The structure of these business objects is very similar to the structure of SAP R/3 Financial Accounting component (FI) business objects, which are common business objects in SAP R/3 application domains. The data of both kinds of objects is stored in special tables that come along with the benchmark. The layout of the benchmark tables is based on the layout of SAP R/3 standard tables: the first column in the table specifies the SAP client number followed by additional key columns and the columns storing the data. The data columns are mainly storing character data of different lengths, but some columns store date, time and number values, respectively. The key columns are of type *CHAR(18)*. Also the layout of the objects is very similar to the layout of FI objects: For each archiving object there exists one *leading* table and a set of *dependent* tables. In the case of COMPLEX objects two of the dependent tables constitute leading tables for their own sub-hierarchy. The dependent tables are connected with the leading table using foreign key constraints (see Figure 1). For each archiving object there exists one record in the leading table and a set of records in the dependent tables. The leading tables have 30 columns while the dependent tables have 60 columns. The tables are filled using a data generation tool, which comes along with the benchmark. The data generation tool allows the users to adapt the database to their needs by providing a set of generation parameters. These parameters are stored in a configuration table which is read by the generation tool (see Table 1). The following sections describe the parameters in more detail.



a) SIMPLE Objects b) COMPLEX Objects

Figure 1. Tables and Their Dependencies

Object_Type The Object_Type is a unique identifier that identifies a set of generation parameters. All objects of a certain type are generated using the same generation parameters. The value of the object type parameter is stored in a key field of the benchmark tables and therefore allows to store objects of different type in the benchmark tables at the same time. This enables the users to evaluate different business object layouts at the same time.

Number_of_Objects This parameter specifies how many objects of a certain object type have to be generated.

Load_Factor SAP R/3 is a comprehensive and highly generic business application system that was designed for companies of various organizational structures and different

1 Even though the data is deleted from the production database it is still accessible via the SAP R/3 system.

Object_Type	Number_of_Objects	Load_Factor	Different_Values	Min_Rows	Max_Rows	Char_Fill_Factor
Object1	100	1	0	5	10	1
Object2	1000	0.5	20	0	20	0.7

Table 1. Configuration Table Used by the Data Generation Tool

lines of business (e.g., production, retailing, finance, etc.). Due to this generality many default values may occur in the production databases. To simulate this effect, the Load_Factor parameter of our data generation tool specifies, how many data columns are actually filled with data and how many remain untouched.

Different_Values The number of distinct values contained in table columns of ERP systems like SAP R/3 is often very small even when the types of the columns allow a greater variety of values. For instance, some columns of type *CHAR(1)* are used as Boolean value, e.g., they contain only the characters *X* (to denote that the corresponding business object has a certain property) or space (to denote that the business object lacks this property). To generate databases with a similar behavior the parameter Different_Values allows to specify how many different values should be used during database creation when generating the data for a column.

The Different_Values parameter has a great impact on the object compression rate during the archiving process: The lesser different values occur, the better the compression works. This again influences the archiving file size.

Min_Rows and Max_Rows The parameters Min_Rows and Max_Rows specify how many records of a business object a dependent table at least contains (Min_Rows) and what the upper bound for the number of records per business object in the dependent tables is (Max_Rows).

Char_Fill_Factor With the Char_Fill_Factor parameter objects of different sizes can be generated. The Char_Fill_Factor parameter denotes to which degree a data column of type character is actually filled with data. E.g., when the type of a data column is *CHAR(18)* and the Char_Fill_Factor is 0.5, then the column is filled with 9 characters.

The above described parameters can be used to generate objects of different sizes and different layouts. E.g., SIMPLE objects can be generated with only a few entries per object in the dependent tables (using the Min_Rows and Max_Rows parameters) and the entries themselves can be kept small (using the Char_Fill_Factor and Load_Factor parameters). These objects correspond, for instance, to orders of individuals: small orders with only a few line-items in it. On the other hand large COMPLEX objects can be generated simulating orders in a B2B scenario with many line-items and a lot of additional information (e.g., discounts or shipping information) per line-item.

There are many more scenarios possible. To actually archive these objects additional components are necessary, which come along with the benchmark and are described in the next section.

3.2. The Archiving Programs

To be able to actually archive SIMPLE and COMPLEX objects we integrated the necessary archiving programs into SSQJ, an SAP performance evaluation tool. The archiving programs were developed according to the guidelines for developing archiving programs published by SAP [7]. The programs allow users to specify what should be done (archive objects or read objects from the archive) and what volume of data—i.e., how many objects—should be involved.

During the archiving process each object obtains a unique ID which allows the SAP R/3 system (and therefore the users) to reference a certain archived object in the archive—e.g., for validation purposes or to read it from the archive. Our archive programs generate these IDs from the key fields of the leading object tables. This approach allows us to influence the reading of objects from an archive in a manner, that objects can be read sequentially—i.e., in the order the objects were archived—or randomly.

The archiving benchmark can easily be adapted to new archiving scenarios because our archiving programs are developed according to the guidelines published by SAP. This means that a change in the hardware used for archiving or a change in the size of the archive files doesn't affect our archiving programs or our data generation tool.

Even when a completely new archiving method is implemented within the SAP R/3 system, which is not compatible with the known archiving methods, the benchmark can be used by just adapting the archiving programs. Again, the data generation tool can be used without any changes because the way an object is archived doesn't influence the definition of an archiving object.

4. Exploiting XML for Archiving Purposes

Archived data is rarely used data from the SAP R/3 system's point of view. However, for the company itself this data is still very useful for, e.g., decision support or data mining. Sometimes, this data is also indispensable for legal reasons. Therefore the archived data should be accessible for many years and by many different applications. However, so far the archived data is stored in special proprietary archive file formats. Therefore, the data is only accessible through the SAP R/3 system it was archived with. Moreover, after every change in the archive file format (e.g., due to an upgrade), the archived data has to be transformed to the new format. This transformation is very expensive for SAP (because they have to provide the transformation tools) and for the customers (they have to transform their archives). To solve these problems, we investigated in cooperation with SAP how XML could be integrated as an

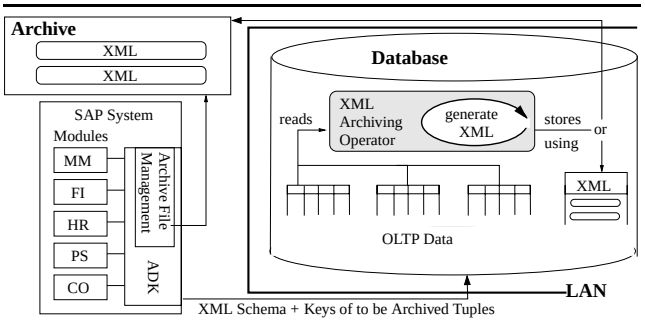


Figure 2. The XML Archiving Operator

archive document format into their systems[2]. XML [1] is a self-contained format and therefore the archive data stored in XML documents can be read by other applications and the data is readable for years even when an application upgrade occurs. The archiving benchmark is also applicable to such an *XML archiving scenario*: only the archiving programs have to be adapted. Then it is possible to evaluate the impact of certain parameters—e.g., the ratio between 'real' data and XML meta-data—on the XML generation. In fact, we already used this benchmark to assess and improve the performance of the XML Archiving Operator we proposed in [10]. The proposed XML Archiving Operator allows to archive business objects and to store the data of each object in a separate XML document. The XML Archiving Operator archives business objects at the database level instead of archiving the data of a business object at the application level (see Figure 2). To do so, an XML schema and the keys of the data to be archived are passed to the XML Archiving Operator. The XML Archiving Operator reads the data referenced by the keys from the production database, generates XML documents (one for each archived business object) corresponding to the annotated XML schema and then deletes the data from the production database. To actually generate the XML from the data stored in the database, the XML generation techniques of the XML Archiving Operator can be used [11]. It is also possible to use the XML features of the underlying database or more general relational-to-XML-transformation approaches like the ones described in [2, 8, 4].

5. Conclusion

In this work we presented an archiving benchmark which allows to evaluate different SAP R/3 archiving scenarios. In particular, the benchmark allows to assess advanced database features like DBMS-internal XML document creation or bulk deletion with respect to the archiving process. Archiving in this context means, to demote seldomly used business objects from the production database to tertiary storage , where the objects are still accessible from the SAP R/3 system. Archiving limits the size of a database and improves therefore the overall performance of the database system and alleviates the work of the database administrators. This is necessary, because the database volumes of todays enterprise resource planning (ERP) systems like SAP R/3 are growing at a tremendous rate and some of them have already reached a size of several Terabytes. OLTP (Online Transaction Processing) databases of this size are hard to maintain and tend to perform poorly. However, so far no benchmark existed to evaluate different archiving scenarios. We therefore implemented a generic benchmark which is applicable to many different system layouts and allows the users to evaluate different archiving scenarios. We implemented a powerful data generation tool and provided the necessary SAP R/3 programs to archive business objects. To actually evaluate the archiving process, we integrated the benchmark into SSQJ, an SAP internal performance evaluation tool.

2 This is ongoing work. Until now, this functionality has not been released.

References

[1] eXtensible Markup Language. `www.w3.org/XML`.

[2] M. J. Carey, D. Florescu, Z. G. Ives, Y. Lu, J. Shanmugasundaram, E. J. Shekita, and S. N. Subramanian. XPERANTO: Publishing Object-Relational Data as XML. In *Proceedings of the Third International Workshop on the Web and Databases*, pages 105–110, Dallas, USA, May 2000.

[3] J. Doppelhammer, T. Höppler, A. Kemper, and D. Kossmann. Database performance in the real world: TPC-D and SAP R/3. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, pages 123–134, Tucson, AZ, USA, May 1997.

[4] M. Fernández, Y. Kadiyska, D. Suciu, A. Morishima, and W.-C. Tan. SilkRoute: A framework for publishing relational data in XML. *ACM Trans. Database Syst.*, 27(4):438–493, 2002.

[5] A. Gärtner, A. Kemper, D. Kossmann, and B. Zeller. Efficient bulk deletes in relational databases. In *Proc. IEEE Conf. on Data Engineering*, pages 183–192, Heidelberg, Germany, 2001.

[6] P. Lyman and H. Varian. How Much Information?, 2003. Retrieved from `www.sims.berkeley.edu/research/projects/how-much-info-2003/` on 11/03/2003.

[7] SAP AG. Archiving Application Data. `help.sap.com/saphelp_46c/helpdata/en/6d/56a06a463411d189000000e8323d3a/frameset.htm`.

[8] J. Shanmugasundaram, E. J. Shekita, R. Barr, M. J. Carey, B. G. Lindsay, H. Pirahesh, and B. Reinwald. Efficiently publishing relational data as XML documents. *The VLDB Journal*, 10(2-3):133–154, 2001.

[9] H. Stefani, editor. *Archiving Your SAP Data -A comprehensive guide to plan and execute archiving projects*. SAP Press, 2003.

[10] B. Zeller, A. Herbst, and A. Kemper. XML-Archivierung betriebswirtschaftlicher Datenbank-Objekte. In *Proc. GI Conf. on Database Systems for Business, Technology and Web (BTW)*, pages 127–146, 2003.

[11] K. Zimmermann. Efficiently Archiving Data Using the XML-Archiving-Operator. Master's thesis, Universität Passau, Fakultät für Mathematik und Informatik, D-94030 Passau, 2003.