Performance Tuning for SAP R/3

Copyright 1997 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Alfons Kemper *Donald Kossmann* *Bernhard Zeller*
University of Passau, Germany
`http://www.db.fmi.uni-passau.de`

1 Introduction

SAP R/3 is the most successful product for enterprise resource planning (ERP). It is used by most Fortune 500 companies and comprises modules for human resource management, accounting, logistics, etc. Like many other application systems, SAP R/3 is based on a commercial relational database system which is used to store all R/3 related data; e.g., a company's sales information. When installing R/3, it is possible to choose among several commercial systems; e.g., Adabas D, IBM UDB, Informix Adaptive Server, Microsoft SQL Server, or Oracle 8.

Obviously, very good performance is crucial for users of SAP R/3. In many companies, transactions of thousands of users must be processed concurrently by SAP R/3 and the underlying database system. In addition, the size of the database can become very large. Today, the largest SAP R/3 databases have about 1.5 terabytes; in a few years, these databases are likely to grow several times that size as a result of new R/3 releases and developments; e.g., component architecture or SAP's business information warehouse. Unfortunately, tuning the performance of an SAP R/3 system is very difficult because both the R/3 application system and the underlying database system must be tuned. Furthermore, the performance of an R/3 system can suffer because parts of R/3 were designed and implemented in the early eighties at a time when commercial database systems and their interfaces were immature. Finally, SAP R/3 is an open system that allows customers and third-party vendors to integrate specialized modules which are not part of the *R/3 standard*. Such modules can severely hurt the performance of the whole system, if they are poorly implemented.

The purpose of this paper is to give an overview of various performance aspects of SAP R/3. This paper describes tuning options and experiences for transaction processing (OLTP) and query processing

(OLAP) and shows how an R/3 system can be monitored and benchmarked. The paper is structured as follows: Section 2 gives an overview of the architecture of R/3. Sections 3 and 4 address performance issues for OLTP and OLAP workloads. Section 5 describes monitoring of SAP R/3 and benchmarks developed for R/3. Section 6 contains a summary.

2 Overview of SAP R/3

SAP R/3 is the market leader for integrated business administration systems. It integrates all business processes of a company and provides modules for finance, human resources, material management, etc. SAP R/3 is based on a (second party) relational database system which serves as an integration platform for all components of SAP R/3. The database system manages the SAP database which stores all business data of a company (e.g., customer and supplier information, orders, ...), all of SAP R/3-internal control data, an SAP R/3 data dictionary, and the code of all application programs. Virtually no data are stored outside this SAP database, thereby avoiding the use of a file system.

Describing the whole system in detail is beyond the scope of this paper. In the following, we will focus on the properties of SAP R/3 which are most relevant for performance tuning. In particular, we focus on the SAP R/3 interface to the relational database system back-end.

2.1 Architecture of SAP R/3

SAP R/3 [WHSH96, BEG96, Sch99, DHKK97] is based on a three-tier client/server-architecture with the following layers (see Figure 1):

1. The presentation layer. It provides a graphical user interface (GUI) usually running on PCs that are connected with the application servers via a local (LAN) or a wide-area network (WAN).
2. The application layer. It comprises the business administration “know-how” of the system. It processes pre-defined and user-defined application programs such as OLTP and the implementation of decision support queries. Application servers are usually connected via a local area network (LAN) with the database server.
3. The database layer. It is implemented on top of a (second party) commercial database product that stores all data of the system, as described above.

In a small company that uses SAP R/3, the application servers and the database system could be installed on the same middle-range machine and users would enter business transactions or issue decision support queries using their PCs. Such a configuration, however, is not practical for large companies with a very high volume of data and transactions. In such companies, all application servers and the database system would be installed on separate dedicated machines. To this end, SAP R/3 has been ported to a large variety of hardware and operating system platforms, and it is also operational on a number of commercial RDBMSs.

2.2 Data Model and Schema of SAP R/3

SAP R/3 is a comprehensive and highly generic business application system that was designed for companies of various organizational structures and different lines of business (e.g., production, retailing, ...). This genericity and comprehensiveness resulted in a very large company data model with over 13.000 database tables. To manage the meta data (e.g., types and interrelationships) of these tables, SAP R/3 maintains its own data dictionary which is (like all other data) stored in SAP’s relational database and which can be used by SAP application programs.

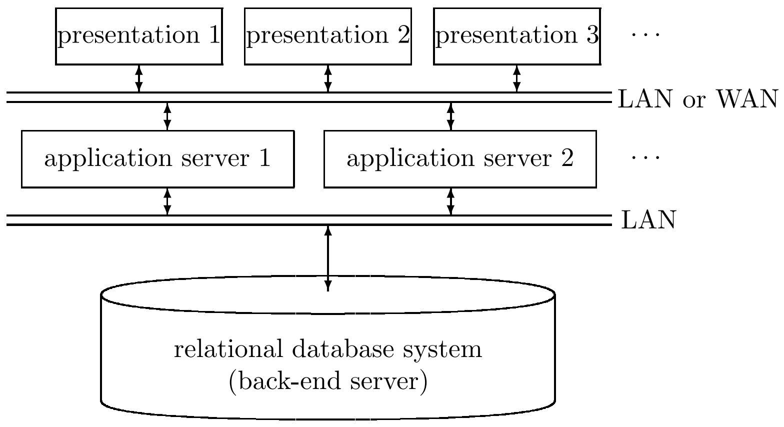

Figure 1: Three-Tier Client/Server-Architecture of SAP R/3

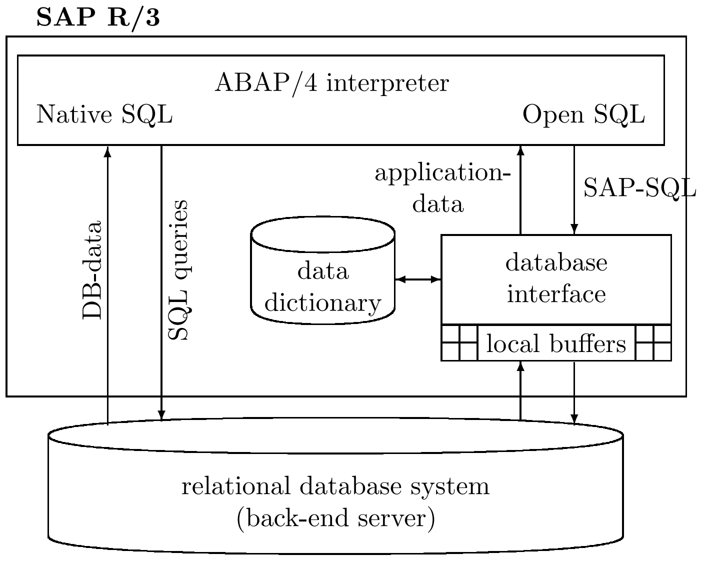

Figure 2: Database Interface of ABAP/4

2.3 ABAP/4

Applications of the SAP R/3 system are coded in the programming language ABAP/4 (**A**dvanced **B**usiness **A**pplication **P**rogramming Language) [MW97]. Except for a small kernel, actually the entire R/3 system is coded in ABAP/4. ABAP/4 is a so-called Fourth Generation Language (4GL) whose origins can be found in report/application generator languages. For this reason, ABAP/4 programs are often called reports. In the course of the R/3 evolution, ABAP/4 was augmented with procedural constructs in order to facilitate the coding of more complex business application programs. For example, ABAP/4 contains language constructs to program so-called "Dynpros" which are dialog programs with a graphical user interface including the logic for validating and processing user entries. Recently, SAP has extended its ABAP programming language with object-oriented features, sometimes referred to as ABAP Objects.

ABAP/4 is an interpreted language, which makes it very easy to integrate new ABAP/4 application programs into the system. Like ordinary data, all ABAP/4 application programs are managed by the R/3 data dictionary and the program code is stored in the SAP database. Also, SAP R/3 includes a software development workbench for developing new application code.

As sketched in Figure 2, ABAP/4 provides commands that allow to access the database via two different interfaces: *Native SQL* and *Open SQL*. The Native SQL interface can be addressed by so-called EXEC SQL commands. It allows the user to access the SAP database directly without using the SAP-internal data dictionary. The advantage is, that the database system-specific properties and services (e.g., non-standard SQL statements like optimizer hints or specialized operators like *cube*) can be fully exploited and additional overhead by SAP R/3 is avoided. However, using the Native SQL interface incurs some severe drawbacks: (1) The EXEC SQL commands may be database system-specific which renders non-portable ABAP/4 programs. (2) By circumventing the SAP-internal data dictionary, EXEC SQL commands cannot access encapsulated relations (containing encoded/clustered data that can only be interpreted using the transformations maintained in the data dictionary). (3) Native SQL reports are potentially unsafe because Native SQL directly reads database relations, and the implementor of a Native SQL report might overlook intrinsic business process interpretations which are otherwise carried out implicitly by SAP R/3's application programs; that is, bypassing SAP R/3's data dictionary requires expert knowledge about the rules and dependencies of the system.

Safe and portable ABAP/4 reports can be written by relying exclusively on ABAP/4's Open SQL commands. Consequently, with very few exceptions the entire R/3 system shipped by SAP is programmed using Open SQL.

Transaction Processing in SAP

SAP uses the term *Logical Unit of Work* (LUW) for transactions. Basically, an SAP LUW has the same ACID properties as database system transactions: an SAP LUW can span several dialog steps and an SAP LUW is either executed completely or not at all (i.e., atomicity). However, SAP LUWs are not mapped 1:1 to database transactions; rather a more complex SAP LUW may involve several database transactions. To synchronize LUWs SAP implemented its own locking scheme which is managed by a (centralized) *enqueue server* which runs in one of the application servers. Basically, SAP also implemented its own TP monitor consisting of a *message handler* and *request queue* in every application server. In comparison: PeopleSoft uses third-party TP monitors such as Tuxedo.

Figure 3 illustrates the processing of user dialog steps by the application server. When a user logs in, a message handler finds the application server with the smallest load (load balancing). This application server handles all of the requests of that user session. A user session consists of several transactions (LUWs), and every transaction consists of several dialog steps. Every application server has one dispatcher process and several work processes. The dispatcher queues requests until a work process is available to carry out the dialog step. For this, the relevant data is "rolled into" the private buffer and the ABAP program is interpreted. Thus, every user session is handled by a single application server, but dialog steps of the same LUW may be handled by different work processes. There is, however, an exception for transactions that involve large objects: They are assigned to exclusive work processes to avoid the cost of rolling data in and out.

Figure 4 summarizes the two-phase processing of an SAP transaction (LUW): Log records for updates—rather than issuing actual database updates—are generated during the processing of the dialog steps. These log records are translated into update requests which are propagated to the RDBMS in the posting phase. SAP locks are requested during the online phase and released after the posting phase is complete (2-phase locking). But note that these are SAP specific locks; the corresponding data is not locked in the database which makes it, in theory, possible to access and update the data via other, e.g., non-SAP, database application programs accessing directly the underlying RDBMS tables. Like dialog steps, posting steps are possibly handled by different work processes (potentially in parallel).

There are several reasons why SAP installed its own transaction processing scheme on top of the database transaction management: Typical RDBMSs do not allow transactions that cross process boundaries. This is necessary in SAP because the dialog steps of an LUW can be handled by different

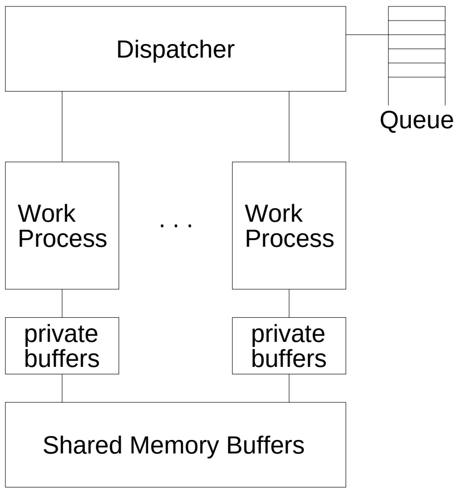

Figure 3: Architecture of the Application Server

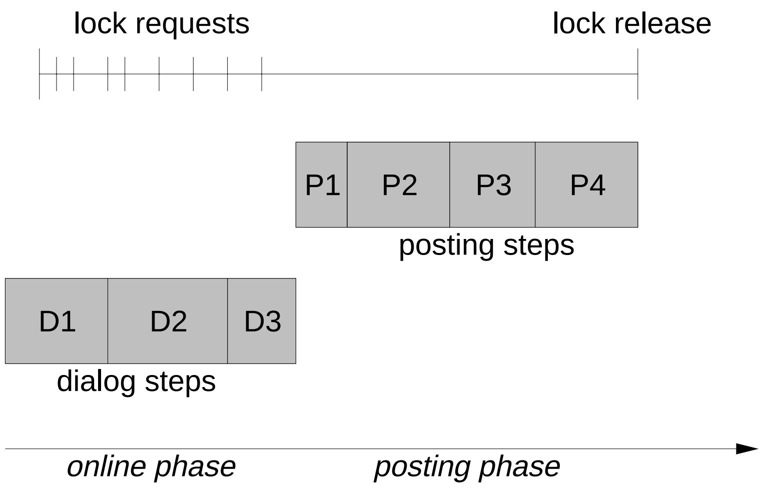

Figure 4: 2-Phase Processing of SAP Transactions

work processes. Aborts of user dialogs are quite frequent (e.g., out of stock) and are carried out before posting—as a result, no roll-back at the RDBMS, the bottleneck, is required for such user-initiated aborts. SAP carries out locking in the granularity of "business objects" which are defined in the SAP dictionary. A business object typically comprises many tuples of different relations.

3 Transaction Processing in SAP R/3

3.1 Application Servers and Work Processes

Obviously, one crucial decision is to determine the hardware used for the application server machines and the database backend server. This decision impacts greatly how many concurrent users can be supported. The following is a brief overview of possible hardware configurations [BEG96, KKM98]:

1. *tiny* (1 user): the whole system is installed on a laptop; the golf club of one of SAP's founders, for example, is organized in this way. Such installations are particularly useful for teaching purposes.
2. *small* (10 users): application and database server run on one mid-range machine
3. *medium* (100 users): a few mid-range machines are used to run the application servers and a separate machine is used for the database server; the machines are connected by an Ethernet
4. *large* (1000 users): several machines are used for the application servers and a high-end (multi-processor) machine is used for the database server; the machines are connected by a very high-speed LAN.

Another decision concerns the number of work processes established at each application server. Clearly, the CPU of an application server machine will be under-utilized if too few work processes are established; on the other hand, many CPU cycles will be wasted due to context switches if too many work processes are established. Furthermore, the number of work processes impacts the use of main memory (Section 3.2). A good rule of thumb is to have one work process for five to ten users.

3.2 Main-memory Management

An SAP R/3 application server divides the available main memory into several segments. First of all, the memory is divided into *shared memory* which can be accessed by all working processes and into *local memory* for each individual working process. The shared memory is further subdivided into the *R/3 extended memory*, the *R/3 paging area*, and the *R/3 buffer/cache*. The local memory of a working process is subdivided into local *roll memory*, the *R/3 heap*, and memory for local data (e.g., application code). The size of each of these segments can (and must) be set by an R/3 system administrator. It should be noted that SAP R/3 main memory management is of course implemented on top of the virtual memory management of the operating system.

In order to demonstrate the importance of a right configuration, we will show how the roll memory, the R/3 extended memory, and the R/3 heap memory are used during transaction processing. A transaction is composed of several dialog steps; each dialog step may be processed by a different working process. To process a dialog step, a working process copies all the data of the transaction generated by previous dialog steps from the R/3 extended memory into its roll memory, carries out the dialog step, and copies the updated data back into the R/3 extended memory. If the data of a transaction grows larger than the size of the roll memory, then the working process carries out the dialog step directly in the R/3 extended memory; in this case, only pointers to the data are copied into the roll memory. As a result, the size of the roll memory determines whether work processes operate using private (local) or shared memory. In fact, starting with Release 4.0 SAP recommends to set the size of the roll memory to 1 in order to force the system to carry out all dialog steps directly in the R/3 extended memory because shared memory access has become cheap and a great deal of copying data can be saved this way.

In situations in which the R/3 extended memory is exhausted and cannot hold all data of all active transactions, the data of a transaction is copied into the R/3 heap. In the R/3 heap, data can only be read and updated by the same working process that wrote the data into the heap. As a result, all dialog steps of a transaction must be processed by the same work process once the data of the transaction has been written to the R/3 heap. Configuring the size of the R/3 extended memory, therefore, impacts the dispatching of dialog steps.

From this discussion it should have become clear that tuning the main memory allocation is very difficult for the R/3 application servers. (Buffer allocation for the database server is difficult, too, – for different reasons – and must be carried out in addition to the main memory management of the R/3 application servers.) A list of recommendations is given in [Sch99]. Furthermore, R/3 ships with a tool called *quick size* that helps R/3 administrators. Interestingly, Microsoft provides a special feature in the Windows NT operating system to help the administrator. The idea is to dynamically adapt the size of the R/3 extended memory so that the administrator need not set this parameter explicitly and need not consider the tradeoffs between executing dialog steps in the R/3 extended memory and the R/3 heap. This initiative is called *zero main memory administration.*

3.3 Caching

In order to reduce the load on the database server, a potential bottleneck, all R/3 application servers make use of caching. Any kind of data including application code, constraints, schema information, and user data such as *Customer* information, can be cached in the buffers of an application server. Hit rates of 90% and even higher are not unusual for OLTP workloads in practice. The decision of what and how to cache can again be made by the R/3 system administrator. For example, the administrator can determine that *Sales* information which is bulky and frequently updated should not be cached, whereas information about *Regions* should be cached because such data is small and rarely updated. The administrator can also determine whether whole tables or individual records of a table should be cached (i.e., the granularity of caching). For all (13004) tables that are part of the *standard vanilla* R/3 schema, R/3 provides default settings. For new, user-defined tables, no caching is the default.

4 Query Processing in SAP

4.1 SAP's Query Language Features

As shown in Figure 2, the SAP R/3 system provides the two query interfaces *Native SQL* and *Open SQL.* Since almost all built-in applications access the database via the *Open SQL* interface we will concentrate on its query facilities. The two basic query constructs of the Open SQL interface are the SELECT … ENDSELECT and the SELECT SINGLE statements:

```
SELECT ⟨attribute list⟩                SELECT SINGLE ⟨attribute list⟩
FROM ⟨table⟩                           FROM ⟨table⟩
WHERE ⟨predicate⟩                      WHERE ⟨unique predicate⟩
... process current tuple              ... process the only tuple
ENDSELECT.
```

The basic SELECT command accepts any kind of predicate in its WHERE-clause. The SELECT SINGLE command, on the other hand, requires predicates on *unique* fields of a table so that at most one tuple qualifies and is returned for further processing.

Up to recent R/3 releases, both SELECT commands were restricted to a single SAP table or view. That is, unless a (join-)view was defined, it was not possible to implicitly describe a join, as is possible in SQL by referencing several relations in the FROM-clause. Join views could only be formulated over transparent tables (i.e., non-encoded tables that can be interpreted by the RDBMS without the SAP data dictionary) and only along primary key/foreign key relationships.

To evaluate a general join within the Open SQL interface, the implementor had to code an ABAP/4 program with nested SELECT...ENDSELECT or SELECT SINGLE loops. This is demonstrated in the following program fragment:

```
SELECT ⟨attribute list⟩
FROM ⟨outer table⟩
WHERE ⟨simple predicate⟩.
        SELECT ⟨attribute list⟩
        FROM ⟨inner table⟩
        WHERE ⟨join predicate⟩.
        ... processing of the current inner (and outer) tuple
        ENDSELECT.
... processing of the current outer tuple
ENDSELECT.
```

Such a program evaluates the join of the tables, without making use of the join methods of the underlying database system. In essence, it corresponds to an (index) nested loops join with the additional overhead of “crossing” the interface between database system and ABAP/4 program for every tuple of the outer relation in order to find the matching tuples of the inner relation.

Furthermore, groupings and aggregations could not be incorporated into the Open SQL SELECT statements of older SAP R/3 releases. As a consequence, all groupings and aggregations had to be performed by the SAP system, thereby possibly transferring huge amounts of data from the RDBMS to the SAP application server.

Recently, SAP has incorporated joins, groupings, and simple aggregations into the Open SQL SELECT statements. These operations can now be delegated to the underlying RDBMS and benefit from the RDBMS's advanced join and groupby algorithms. However, one has to keep in mind that it requires reprogramming the many (thousands of) built-in applications before this takes effect. Delegating these operations to the RDBMS will, of course, “strain” the underlying database server (in particular the query optimizer and the query engine) even more than current SAP installations already do.

4.2 SAP Query Optimization Features

To optimize query processing, SAP R/3 implements two techniques which take effect if the Open SQL interface is used: (1) *Cursor caching* which reduces the overhead of calls to the RDBMS by using the same cursor for, say, all the queries that retrieve the matching tuples of the inner relation in a nested SELECT statement. Cursor caching is possible because most database systems allow parameterized queries and provide a cursor REOPEN command in their API. (2) *Caching* data in SAP R/3 application servers in order to avoid calls to the RDBMS altogether (cf. Figure 2). For caching, the typical tradeoffs between read and update frequency apply; in addition, SAP R/3 does not fully guarantee cache coherency in a distributed environment as updates are only propagated periodically.

Cursor Caching In order to enable cursor caching, SAP is transforming queries containing literals (constants) into parameterized queries—in anticipation that the same query with a different parameter will be issued again in the near future. As an example, consider the following query:

```
SELECT ...                     SELECT ...
FROM VBAP                 ⇝    FROM VBAP
WHERE KWMENG < 9999            WHERE KWMENG < ?
ENDSELECT.
```

That is, the query on the left is translated into a parameterized query, and "?" denotes the query parameter. Due to this translation, the optimizer of the RDBMS cannot estimate the selectivity of the predicate of the translated query and thus *blindly* generates a plan. In some cases, the optimizer chooses bad access plans—due to the fact, that the selectivity could not be estimated correctly.

Caching If a relation is cached in the application server the Open SQL interface is exploiting this. SAP differentiates between several caching levels for a particular relation:

- complete buffering: The entire table is cached and any SELECT on this table can (and will) be performed by the application server on the cached copy.
- single-tuple and generic buffering: Only those tuples are cached whose key attributes have the specified values. For single tuple caching, all key attributes are specified; for generic caching only a subset of the key attributes is specified. Queries can only be evaluated on the cached copy if the WHERE clause specifies a superset of the attributes that determine the caching.

As pointed out before, the SAP system "comes with" pre-configured caching rules for all its built-in tables which may, however, need to be adapted for particular workloads.

In addition to the implicit caching, ABAP/4 allows the materialization of query results in internal (i.e., temporary) tables in order to use this data for further processing. For example, it is possible to materialize the inner relation of an Open SQL nested-loops join report and avoid repeated calls to the RDBMS this way. Using such a materialized table, SAP provides a language construct called

...FOR ALL ENTRIES IN ⟨materialized table⟩ ...

to obtain the matching tuples of another table—i.e., to obtain the semijoin result. For this purpose, the SAP query processor generates a corresponding WHERE clause containing a disjunct for every tuple of the materialized table. This way, users can implement their own join routines; for example, they could implement a semijoin-supported sort/merge join executed in the application server—thereby taking load off the database system.

4.3 Tuning SAP Queries

SAP provides tracing tools for analyzing potential performance bottlenecks resulting from poor query evaluation plans. Having identified an ABAP program with poor query execution performance several "repairs" may be appropriate:

- Reprogram the application using the advanced query features that allow to delegate complex operations to the underlying database server.
- If the RDBMS server is particularly loaded it is possible to perform more of the data processing in the application server, e.g., sorts, groupings, etc.

- Reprogram the application to use the Native SQL interface as opposed to the Open SQL interface. Database-specific features such as user hints and specialized operators can only be exploited via this interface. The SAP built-in applications usually don't use this interface because, among other reasons, it violates portability between database vendors.
- Define new indexes on the underlying database tables. This may, however hurt the "mission-critical" OLTP performance.
- Materialize intermediate results in the application server and reuse these snapshots in query evaluation.
- Update statistics of the underlying database to ensure that the query optimizer has precise numbers.
- Sometimes SAP users may find it necessary to suppress statistics in order to "trick" the optimizer into generating a desired plan that it wouldn't choose otherwise. Note that from the Open SQL interface it is not possible to pass *optimizer hints* to the RDBMS.
- Another trick is to temporarily turn the database system optimizer to a different optimization level (as is possible with some RDBMS products).

4.4 BW: SAP's Data Warehouse

Optimizing the SAP system for query performance may be hurting the OLTP performance—which, in most SAP installations, is more critical than the OLAP performance as the OLTP operations are needed for the companies' operational business. To avoid this, SAP users with OLAP requirements can use SAP's recently developed data warehouse product, called Business Information Warehouse (BW), in conjunction with the R/3 system. Using the BW the OLAP-relevant data is extracted from the operational R/3 OLTP system and stored in pre-defined star schemes [CD97], called *InfoCubes* in SAP terminology.

The key features of the BW Server are (cf. Figure 5):

- Like MicroStrategy, it runs on a variety of third party RDBMSs.
- The BW provides open Business Application Programming Interfaces (BAPIs) for
  - data loading: to accommodate the extraction of data from non-SAP systems besides R/3 systems.
  - OLAP processing: e.g., for third party visualization tools.
- The SAP BW provides a pre-configured meta data repository consisting of InfoCubes catalog, report catalog, information source catalog, etc.
- Many pre-defined InfoCubes for common business applications exist, e.g., market segment analyses, profitability analyses, stock inventory analyses, corporate indicator systems.
- To obtain the data (and subsequently the incremental updates) from R/3 OLTP systems predefined extraction routines are delivered.

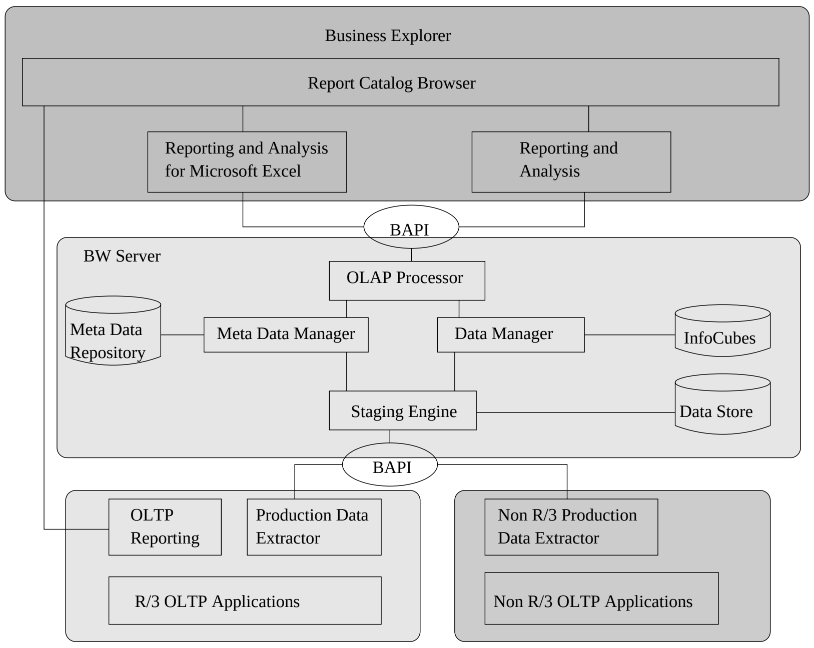

Figure 5: Architecture of SAP's Data Warehouse

5 Monitoring and Performance Evaluation

In the previous two sections we described different options for transaction and query processing in SAP R/3. In this section, we describe tools that can be used to monitor and assess the performance of SAP R/3 installations.

5.1 Monitoring

SAP R/3 ships with a very powerful monitoring tool. This tool measures, for instance, the lengths of the queues at every application server, the cache hit ratio, the running time of database operations and of ABAP/4 operations, the number of transactions that are committed and rolled back, the response time of dialog steps, and the CPU, disk, memory, and network utilization. All these performance statistics are stored in the relational database. In addition, alerters inform system administrators if the system performs poorly. Furthermore, SAP offers a special *early watch* service. If a company buys this service, then SAP specialists periodically assess the performance statistics of the company's R/3 installation and provide suggestions for performance improvements [BEG96].

5.2 SSQJ Tool

In order to detect performance problems, SAP has also developed a tool called SSQJ. This tool stores the code and running times of queries that have caused performance problems in at least one R/3 installation in the past. The queries recorded by SSQJ include simple select queries involving different kinds of indices as well as complex multi-way join queries; for example, the queries of the TPC-D benchmark for SAP R/3 are maintained by SSQJ (Section 5.3). Currently, this tool is mostly used by database vendors in order to test new releases. Before a new release of an RDBMS is shipped, the vendor runs all the queries stored in SSQJ and SSQJ indicates which queries showed good or bad performance compared to the running times obtained using an older release. SSQJ could also be used to assess the R/3 installation of a customer; that is, before working with the new R/3 installation, a customer could run all queries stored in SSQJ and see whether any queries show particularly poor performance. To date, however, we know of no customers that have used the SSQJ tool to assess their installation.

5.3 Benchmarks

A number of benchmarks have been proposed for SAP R/3. The main characteristic of these benchmarks is that the whole system is tested; i.e., rather than testing the RDBMS in isolation, the benchmarks test the execution of typical R/3 operations (dialog steps and queries) that involve processing by the application and database servers. The most prominent benchmark is the SD benchmark for measuring the performance of processing sales and distribution transactions. Many hardware and database vendors have published results of this benchmark. SAP collects and certifies these benchmark results and publishes them, e.g., on the Web. In previous work, we adapted the TPC-D benchmark [TPC95] for SAP R/3 and published results in [DHKK97]. As stated earlier, the queries of the TPC-D benchmark are also part of SAP's SSQJ tool.

6 Summary

In this paper we gave an overview of various performance aspects of SAP R/3. After overviewing SAP R/3's three-tier architecture, we described tuning options and experiences for transaction processing (OLTP) and query processing (OLAP) and briefly surveyed the monitoring and benchmarking tools for an R/3 system.

References

[BEG96] R. Buck-Emden and J. Galimow. *SAP R/3 System, A Client/Server Technology.* Addison-Wesley, Reading, MA, USA, 1996.

[CD97] S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. *ACM SIGMOD Record*, 26(1):65–74, Mar 1997.

[DHKK97] 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.

[KKM98] A. Kemper, D. Kossmann, and F. Matthes. SAP R/3: a database application system. Tutorial handouts for the ACM SIGMOD Conference, Seattle, WA, USA, June 1998. http://www.db.fmi.uni-passau.de/publications/tutorials/.

[Sch99] Thomas Schneider. *SAP R/3-Performanceoptimierung.* Addison-Wesley, Reading, MA, USA, 1999.

[MW97] B. Matzke and A. Weinland. *ABAP/4 - Programming the SAP R/3 System.* Addison-Wesley, Reading, MA, USA, 1997.

[TPC95] Transaction Processing Performance Council TPC. TPC benchmark D (decision support). Standard Specification 1.0, Transaction Processing Performance Council (TPC), May 1995. `http://www.tpc.org/`.

[WHSH96] L. Will, C. Hienger, F. Straßenburg, and R. Himmer. *R/3-Administration.* Addison-Wesley, Reading, MA, USA, 1996.