Database Performance Study

By

Francois Charvet

Ashish Pande

Table of Contents

Part A

Findings and Conclusions

Scope and Methodology

The authors were contacted by a faculty member to conduct a research project for the IS Board at UMSL. While the original proposal would be to assist in SQL optimization and tuning at Company A[1], the scope of such project would be very time-consuming and require specific expertise within the research team. The scope of the current project was therefore revised, and the new project would consist in determining the current standards and practices in the field of SQL and database optimization in a number of companies represented on the board.

Conclusions would be based on a series of interviews with Database Administrators (DBA's) from the different companies, and on current literature about the topic. The first meeting took place 6th February 2003, and interviews were held mainly throughout the spring Semester 2003. Results would be presented in a final paper, and a presentation would also be held at the end of the project.

Individual summaries of the interviews conducted with the DBA's are included in Part B. A representative set of questions used during the interviews is also included.

Database Performance - Background

Although the Relational Database Management System (RDBMS) has become a de-facto standard, its main strengths have to be found in its ease-of-use and querying capabilities, rather than its efficiency in terms of hardware and system overhead. With the constantly growing amount of data being accumulated and processed by companies' information systems, database performance issues become more likely. In the meantime, user requirements and expectations are constantly on the rise, and delay in response time could considerably affect the company's operations. Database performance tuning, or database performance optimization, is the activity of making a database system run faster. SQL optimization attempts to optimize the SQL queries at the application level, and typically offers the biggest potential for database performance optimization.

[1] Company names have been changed for the public version of this paper.

Factors

One of the first tasks in database tuning is to understand the causes of the problems and find the current bottlenecks. This is already a challenge in itself, given the very diverse factors affecting the overall database performance. A first category can be grouped under 'hardware', and includes processor, memory, disk and network performance. Other factors are more directly related to the database system itself, such as the database design, indexing, partitioning or locking. Finally, problems can also arise at the application level. Discussing each of the factors would go beyond the scope of this paper, and extensive literature can be found elsewhere[2].

One conclusion that can be drawn, however, is that database tuning requires very broad skills and comprises the fields of system administration, database administration and application development. While SQL optimization is the primary focus of this research, the authors cannot stress enough how important it is to identify the current bottlenecks and go through a comprehensive analysis of the root causes. With user complaints triggering the start of performance analysis, the role of DBA's is often reduced to fire fighting. The authors were not able to find evidence of such formal analysis of root causes at the companies interviewed. Having all the required skills or experience is not an easy task, and for most companies, this will have to be accomplished by a good communication between system administrators, application developers, and database administrators.

During our interviews, most DBA's agreed that SQL tuning presented by far the main opportunity to improve database performance. Several DBA's explained that hardware issues were usually easier to detect and solve than other causes. Finally, wrong or sub-optimal indexing and misuses of locking procedures were also seen as a key reason for many performance problems. Although the importance of database design was not played down, the following comment from one of the interviewees summarizes the general thought: "re-designing an existing database is not only complex, but also requires huge resources, and business realities will often rule out such undertaking".

[2] For a quick overview of different bottlenecks, see for example Jones, pp.60-69

Performance Monitoring

For a quick overview of performance monitoring, the reader is referred to Darmawan, pp. 251-286[3]. Again, one of the main conclusions is that the system administrators, database administrators and application developers have to be jointly involved in the monitoring process[4]. Usually, the database administrator would be the most capable of pointing out where the main performance bottleneck can be found. A lot of different tools are on the market to improve performance monitoring. Most DBA's agreed that these tools are valuable. A sample of some of the well-known third-party tools is given in appendix 1.

A distinction can be made between three types of monitoring. *Regular, day-to-day monitoring* has for goal to collect information and keep documentation about the system's day-to-day performance. This historical information should also make it possible to anticipate and avoid future problems. Even with such regular monitoring unexpected delays can happen, which is why *ad-hoc monitoring* is also required. Thanks to the historical, regular monitoring, the database administrator should be able to quickly understand where the problem occurred (e.g. long sorts, I/O bottleneck, deadlock situation…). A number of useful SQL scripts to obtain a quick overview of basic system characteristics can be found in Darmawan, pp. 459-472. Finally, *alert monitoring* allows the DBA to specify threshold values for a number of variables.

Once a performance problem is detected, the database administrator should go to a list of question. Appendix 2 is a good example of such a list. The following rules of thumb can also be useful (J. Dunham, pp. 40-41):

- If the performance problem affects only one application, the cause is probably SQL → look at the SQL code
- If the problem affects all the applications, the system has a tuning problem → look at memory, sorting and I/O usage

[3] See also DB2 "Performance problem determination", available online at http://www-3.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/tutorial_performance.d2w/toc

[4] Cf. earlier, performance 'factors'.

- If all the applications for a given table have similar problems, the bottleneck can be found at this level → look at space, locking and indexing

For most of the companies interviewed, tuning seemed to be driven by performance problems (usually triggered by user complaints). Although a lot of companies collect regular performance data, this is usually used as historical benchmark to understand where the problems occurred. The authors had the impression that use of this data to anticipate or continuously track data was only minimal. The main reason is probably that most companies would not have sufficient resources to allocate a (near) full-time DBA to monitoring tasks. At Company F for example, logs for the data warehouse environment are kept including performance monitoring statistics. The logs are then checked once or twice a week.

It is worth noting that Company B is the only company using alert monitoring. Thresholds are set up-front, such as maximum response time in a transactional environment. Applications are not released until the thresholds are met, and automatic notifications would also alert DBA's if query response time exceeded the threshold in the production environment. When DBA's receive such notifications, the performance problem is analyzed and fixed without delay.

Solving Performance Issues

Our research also tried to identify how performance problems could be reduced and which methods were used in practice. Besides hardware upgrades, the following areas in tuning are known to have major impacts:

- SQL Tuning
- Indexing
- Memory (caching)
- Table Partitioning
- Locking

Our research revealed that SQL Tuning and Indexing are the most used areas in dealing with performance related problems. These two areas will therefore be detailed hereunder. For more information about memory settings, table partitioning, and locking, the reader is referred to the recommended literature.

SQL Tuning

SQL Tuning is believed to have the largest impact on performance (more than 50%). SQL is a declarative language, which only requires the user to specify what data are wanted. The DBMS' query optimizer will then decide which access path should be used (Markl, 2003). This decision can be very hard since there might be hundreds or thousands of different ways to correctly process the query. The best execution plan chosen by the query optimizer is called the query execution plan (QEP). To show some of the complexities involved, a few examples have been included in appendix 3.

First of all, it is important to write the query so that only the required columns and rows are returned. This is usually not a problem, although application programmers might still be tempted to use a '`select *`' when it is not appropriate. Although the DBMS' query optimizers should be able to find the best QEP, they are based on mathematical models with many assumptions and parameters (ibid). In practice, this means that by writing slightly different SQL queries, the processing time can still be affected drastically.

Whether SQL code is efficient depends on a wide range of factors, such as the DBMS, indexing, table design and table sizes, memory… A detailed description of all the factors and techniques that can affect SQL performance would go far beyond the scope of this paper. However, the authors would like to give the following conclusions:

- Although most companies agreed that external tools were useful, only one company used a true[5] SQL optimization tool: Quests' SQL Expert at Company B.

[5] A lot of other tools are aimed towards SQL performance problem detection and analysis. More advanced tools will also have advanced features such as automatic hints, test and learning.

- In the future, it is expected that SQL optimization tools become even more powerful, and DBMS vendors also integrate more advanced features in their new releases.
- In the meantime, efforts should be made to ensure that application developers write efficient SQL code during the development phase. A lot of resources are available, and (database) application developers should be trained and exposed to SQL tuning concepts[6]. Koopman (2003) for example, argues that DBA's should also be the support line; DBA's should perhaps spend more time with users, application developers and management, instead of troubleshooting after-the-fact SQL problems.

In general, when the problem is due to poor SQL code, the following steps can be followed:

- Monitor and detect performance problems (cf. monitoring methods above).
- Identify where the problem is (cf. rules of thumb above).
- Run 'explain' (or other DBMS internal access path analyzer tool), and/or third-party query analyzers and SQL optimizer tools.
- Try improved queries to solve the problem.

Indexing (adapted from IBM Administration Guide)

Indexing a table is a way to search and sort the records in a table. With the use of indexes the speed with which the records can be retrieved from the table can be greatly improved. For example, if a customer has placed some orders and the customer number CUST1 is held in the Order table, the database would first go to the order table and then perform a sequential search on the customer number column to find a value that is equal to CUST1.

Indexing can be thought of as a two dimensional matrix independent of the table on which the index is being created. The two dimensions can be a column that would hold data which is sorted extracted from the table on which the index is being created and an

[6] See for example appendix 5. Other resources can be found in the 'recommended literature' section.

address field that identifies the location of the record in the database. After creating the indexes it is important to collect statistics about the indexes using the RUNSTATS utility.

When a table is referenced in a database query and there is no index a table scan must be performed on that table. The larger the table the longer it would take for the table. However, if indexing is used an index scan would be performed. An index scan is much faster than a table scan. Indexed files are smaller and require much less time to be read than a table especially when the table grows bigger.

A database manager uses a B+ tree structure for storing its indexes. In this structure, the top level is called the *root node.* The bottom level consists of *leaf nodes*, where the actual index key values are stored, as well as a pointer to the actual row in the table. Levels between the root and leaf node levels are called *intermediate nodes*.
In looking for a particular index key value, Index Manager searches the index tree, starting at the root node. The root contains one key for each node at the next level. The value of each of these keys is the largest existing key value for the corresponding node at the next level.
Indexes can reduce access time significantly; however, indexes can also have adverse effects on performance. Before creating indexes, consider the effects of multiple indexes on disk space and processing time:

- Each index takes up a certain amount of storage or disk space. The exact amount is dependent on the size of the table and the size and number of columns included in the index.
- Each INSERT or DELETE operation performed on a table requires additional updating of each index on that table. This is also true for each UPDATE operation that changes an index key.
- The LOAD utility rebuilds or appends to any existing indexes.
- The `indexfreespace` MODIFIED BY parameter can be specified on the LOAD command to override the index PCTFREE used when the index was created.

- Each index potentially adds an alternative access path for a query, which the optimizer will consider, and therefore increases the query compilation time.

All the DBA's interviewed talked about the benefits of indexing. Indexes should be carefully chosen to address the needs of the application program.
According to our research indexes should be carefully chosen to meet the needs of the system. In one of the interviews we found that wrong indexing can prove to be harmful also. A typical example is when data and requirements change after the initial design, while the indexing is not adapted accordingly. During the research we found that we should not exceed 10 indexes. Another thing that was found was that if you are going to retrieve/process more than 20% of a table, it should be scanned instead of using index access.

Thus, it can be seen that indexing is a very important thing for fine tuning databases. However, our research and literature show that it has to be done very judiciously otherwise it can backfire. It is very important to see the other factors like disk space, I/O etc. before making any changes as indexing can create an extra load on the system.

Organizational Aspects

The interviewed DBA's were asked to comment on the existing organizational structure and the role of the different parties involved in the database environment. As an introduction, we would like to introduce Rob and Coronel's view of the type of players involved in the database system environment[7]:

- *(Database) System Administrators* oversee the database system's general operations.
- *Database Designers,* or data architects, are responsible for the design of the database structure.
- *(Application) Database Administrators* manage use of the DBMS and ensure correct functioning of the database.
- *System Analysts and programmers* design and implement application programs.
- *End-Users* are the people actually using the applications querying the database.

[7] Rob and Coronel, p19

Most companies' structure coincide closely with the one presented above. In general, data architects are in charge of the database logical model (table structures, relationships…), while the physical implementation is left to the DBA's. The degree of specialization can vary from company to company, probably due to the total size of the IT workforce and number of DBA's in general. For example, only Company D mentioned to have a dedicated optimization group. While this is not the case for the other companies, at Company B for example, three DBA's are known to be more experienced in database tuning. Most companies would have 'system DBA's' next to 'application DBA's'. System DBA's would be in charge of general aspects of the database environment (physical implementation, upgrades, memory, partitioning…), while application DBA's would be much closer to the application developers.

Our interviews paid special attention to the role of the DBA's in SQL optimization and the interaction between DBA's and application developers. The following guidelines can be put forward:

- Application DBA's should work closely with application developers and each specific project should have one or more DBA's formally assigned to it.
- Application DBA's should also have a good functional knowledge. Depending on the size of the company, this could mean that the application DBA's would be organized in 'functional' groups (e.g. billing, inventory…).
- Developers and DBA's on the same projects should report to the same project manager in order to enforce accountability.
- Key tasks of the application DBA's include guidance, review of SQL code, and testing.
- Application DBA's should also be involved in setting the performance requirements (i.e. which response times are acceptable and should be strived for).
- Once in production, the application DBA's would perform monitoring tasks and (assuming enough time was spent up-front) help tune the database

parameters due to changing user patterns (e.g. increased number of users, changing query pattern).

While the practices at the different companies matched the above description quiet well, the real question seems to be how much time and resources are really spend up-front, versus the 'troubleshooting' activities of the DBA's. DBA-B from Company B emphasized the importance of setting rigorous performance requirements during the development and testing phase. Although reviewing all SQL code going into production does require more time and resources, this will avoid much more trouble and rework afterwards. Re-work should not be required once the application goes into production. Hayden also pointed out that writing efficient SQL requires considerable experience and training. As a final note, DBA-E from Company E commented: "In my experience, poor project management and the lack of experience are very often the real causes behind performance problems at later stages".

The authors would like to stress the following points as potential risks and/or problems:

- (Application) DBA's might not have enough experience, training, or functional knowledge to ensure efficient SQL code during the development phase.
- Application DBA's should be given the authority and resources to thoroughly review *all* the SQL code. It might be tempting to only review portions of SQL code, or trust more experienced programmers. While this seems to take more of the DBA's time, the benefits of avoiding troubleshooting performance problems later on will more than offset this.
- As DBA-D (Company D) observed, "testing response times and performance during the development phase can be hard to perform accurately due to the difference between the development environment and the real production environment". For example, certain queries might not receive the same priorities in the testing environment.

- Changing indexing, locking parameters, tuning memory and I/O will often require good communication and coordination between application DBA's and system DBA's or with DBA's more experienced in the tuning field.

Training

The interviewees agreed unanimously about the benefits of having formal training for the DBA's. At Company C and Company B, DBA's receive training twice a year. Interviewees stated Washington University and Quilogy as external training providers. Ideally, training should be adapted to the experience and needs of the different DBA's. At Company E and Company G for example, everyone is responsible for his/her own training path and training type. Unfortunately, the financial situation of the company can impact the training policy. At Company D for example, training could not currently be offered although it is part of the company's policy. Perhaps formal internal training could be encouraged where specific topics are taught to relatively less experienced DBA's.

Type of Database Environments and Normalization

Since the table structure can also impact the overall database performance, the interviews also addressed this topic. A first objective was to find out if databases should be normalized[8], and to which point. Before we answer this question, it is important to understand the difference between different types of database environments. The discussion below will simplify this discussion by considering transactional systems (OLTP, containing operational data) versus decision support systems, or DSS (typically data warehouse environments)[9].

Transactional systems typically handle the bulk of the day-to-day activities of a company, where transactions are processed real-time or 'online'. Some of the key characteristics are the high number of concurrent users, and a high number of simple SQL queries (mainly simple read, insert, update and delete queries). On the other hand, decision support

[8] For readers unfamiliar with normalization concepts, see for example Rob and Coronel, chapter 4, pp.174-207.

[9] For a more detailed approach of the different database profiles and workload, see for example Darmawan, pp.49-59, or Rob and Coronel, pp.613-618.

systems typically extract data from various sources and act as snapshots. Perhaps the most common form is the data warehouse, which typically contains large amount of historic data. In terms of workload, a DSS will usually receive long and complex queries, which can take a long time (minutes or even hours) to complete. The queries are usually function-oriented and require either full-table scans, or summary table scans. One of the main functions of DSS consists of reporting. DSS systems are typically not normalized, and the most frequent schema is the star-schema. The concepts of dimensions and fact table are also very important, and the reader should be at least slightly familiar with these concepts[10].

For DSS, our research revealed that most DBA's agreed that data warehouse environments were more likely to cause performance problems than transactional systems. Of course, one should be aware that the goal of data warehouses is not to provide real-time responses, and 'performance problems' will be subject to the company's and users' expectations in terms of performance. In the interviews, the DBA's unanimously agreed that denormalization was an intrinsic element of data warehouse environments. Because of his experience and background, we discussed DSS more in depth with DBA-F from Company F. In his view, the archiving strategy used for the data warehouse can be very useful to reduce performance problems, especially when the problems are linked to the large quantity of data used. The idea is simple: instead of taking the increasing amount of data for granted, the goal is to select appropriate data to pull out from the database. Moreover, the archived data should be able to be retrieved as quickly as possible. 'Active archiving', a relatively new archiving strategy, seems to be promising for large data warehouse systems[11].

For transactional systems, opinions regarding the impact of normalization on performance were much more divided and company practices could diverge substantially. The answers from DBA's at Company C, Company A and Company D were very similar: while the goal for transactional systems is to design the database in third normal form,

[10] See appendix 5.
[11] Several articles about active archiving are included in the recommended literature. See for example Cash, L., "Database strategy: Active archiving".

denormalization is used in certain cases to improve performance. DBA's The following reasons were mentioned:

- One of the main problems with 3NF is that for complex queries, a lot of smaller table might be involved. Large multilevel joins are sometimes too inefficient.
- Even with the right indexing, response time appears could be too long when very large amounts of data are involved.
- While integrity problems may occur with denormalization, referential integrity (especially with partial dependencies) can be enforced at the DBMS level by defining referential integrity constraints.

At Company B, in contrast, tables are always kept in third normal form for transactional systems. According to B. Hayden, denormalizing tables for performance reasons is not recommendable. Usually, a lot of the problems would rather be due to wrong indexing (e.g. wrong order used in multi-column indexes, over-indexing), or due to inefficient SQL queries. In the same line, N. Swamy agrees that DBA's and programmers blame a lot of mistakes on normalized databases. It is true that writing good SQL code can become more complex with normalized tables, but normalization in itself does not create the performance problems.

References

CASH, L., "Database strategy: Active archiving", Computer Technology Review, Los Angeles, Vol. 22, No., 2002

DARMAWAN, B., Groenewald G., Irving A., Monteiro S., and Snedeker, A. "Database Performance Tuning on AIX", IBM Redbook series, available online at www.ibm.com/redbooks, January 2003, pp. 251-286.

DUNHAM, J. "A guide to large-database tuning", UNIX Review's Performance Computing, San Francisco, May 1999, pp.35-41.

IBM, "DB2 Performance Problem Determination", tutorial available online at http://www-3.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/tutorial_performance.d2w/toc

JONES, D. "The definitive guide to SQL Server Performance Optimization", available online at www.realtimepublishers.com, 23 may 2003.

LANSING, M., "Turbo Charge Your SQL", Powertimes: International Journal for Application Developers, December 2002.

ROB, P. and C. Coronel. "Database Systems: Design, Implementation and Management", Course Technology, 2002, pp.8-35.

SHASHA, D., "Tuning Databases for high performance", AMC Computing Surveys, Baltimore, Vol. 28, No. 1, 1996.

Recommended Literature

Database Performance Tuning, General Resources

BARRY, N., "Troubleshooting your database", Network World, Farmingham, Vol. 19, No. 42, 2002.

DUNHAM, J. "A guide to large-database tuning", UNIX Review's Performance Computing, San Francisco, May 1999, pp.35-41.

KOOPMAN, J., "DBA Call to Action: Make an Impact", Database Journal, 15 May 2003, available online at www.databasejournal.com/sqletc/article.php/26861_2205131_1

MACVITTIE, D., "Clearing database network clogs", Network Computing, Manhasset, Vol. 14, No. 8, 2003.

SHASHA, D., "Tuning Databases for high performance", AMC Computing Surveys, Baltimore, Vol. 28, No. 1, 1996.

WINTER, R., "Large Scale Data Warehousing with Oracle 8i", Winter Corporation white paper, available online at www.wintercorp.com, accessed June 2003.

ANTHES, G., "Database Horizons", Computerworld, available online at www.computerworld.com/databasetopics/data/software/story/0,10801,73164,00.html, accessed 14 May 2003.

GUKUTZAN, P., and Peltzer, T., "SQL Performance Tuning", Addisson Wesley Professional, 2002.

DB2

IBM, "DB2 Performance Problem Determination", tutorial available online at http://www-3.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/tutorial_performance.d2w/toc, accessed April 2003

SHIRAI, T., Dilworth L., Reutlinger R., Kumar S., Bernabe D., Wilkins B., and Cassels B. "DB2 UDB V7.1 Performance Tuning Guide", IBM Redbook series, available online at www.ibm.com/redbooks, December 2000.

MARKL, V., Lohmann G. M., and Raman V. "LEO: An automatic query optimizer for DB2", IBM Systems Journal, Armonk, Vol. 42, No. 1, 2003.

LECCO Technology, Inc., "Adopting LECCO SQL Expert for Maximum DB2 Application Performance", Product white paper, available online at www.leccotech.com, accessed, February 2003.

SONGINI, M., "IBM readies DB2 upgrade with self-tuning technology", Computerworld, Framingham, Vol. 36, No. 31, 2002.

Data Archiving Strategy

CASH, L., "Database strategy: Active archiving", Computer Technology Review, Los Angeles, Vol. 22, No., 2002

LEE, J., "Active archiving and HSM: Best-practices", Computer Technology Review, Los Angeles, Vol. 23, No. 2, 2003

GARRY, C., "Archiving: Databases on a Diet", META Group, 30 January 2003, available online at www.princetonsoftech.com/XXX

Additional resources at: www.princetonsoftech.com

Various

Transaction Processing Performance Council (TPC), www.tpc.org
(benchmarking)

CONNOR, D., "NextGig boosts database performance", Network World, Framingham, Vol. 19, No. 12, 2002

DBAzine
http://www.dbazine.com/

Databasejournal
http://www.databasejournal.com

Part B
Interview Summaries

The May Company		02/06/03

Database System(s) – General

- Company A just upgraded to DB2 version 8.
- The following persons were interviewed:
  DBA-A1 (Director Application Development), DBA-A2 (DBA Project Manager), DBA-A3 (System DBA), DBA-A4 (Application Project Manager), and DBA-A5 (Application Project Manager)
- There are 3 DBA's for approximately 100 application developers

Role of DBA – Organizational Structure

- Training should normally be provided. However, it is expensive and in the current circumstances it is difficult to provide training to everyone.
- DBA's do not necessarily get involved with a project right from the start. Usually, when performance problems arise, DBA's start looking into the problem and try to find out the cause and resolution of the problem.

Performance Problems – Detection, monitoring, general policies/practices

- DBA's usually become involved when a problem arises. There is no real monitoring/tracking. The only way to find out about performance related problems is through user complaints.
- No formal service level standards or thresholds are set up (such as response times, or cache hit ratios).
- Most of the performance problems occur with reporting and decision support environments. There was no real consensus on whether performance problems also occur with transactional systems. This is probably due to the fact that 'performance problem' is a subjective concept in itself.
- Sometimes, queries are slow even with indexes on all the required rows.
- Table sorts can slow down the processing. Sometimes even index scans take a considerable amount of time.

- Multilevel joins are also an important problem.

Troubleshooting Performance Problems

- A query analyzer was made in house. However, it is not extensively used to detect bad queries.
- Writing good SQL is one of the most important factors for fine-tuning databases. This requires experience and regular training.
- Making small changes to the SQL code are often able to drastically change the response times. However, the type of changes required varies widely and it is hard to predict the outcomes.
- Prioritizing of queries is also done.

Normalization – Denormalization

- Normalization is not always enforced in the design process. It is believed that 3NF can decrease performance. It can be useful to have some redundancy. However, normalization is good for transactional systems.

Various

Java was used in the front-end. A Dynamic SQL builder was also developed in-house to create dynamic SQL queries from user inputs.

Company D	DBA-D	7 March 2003

Database System(s) - General

- Roughly 20% relies on Oracle, 80% on DB2.
- 192 DB2 subsystems.
- 15 CPS complexes (for different processing locations). There are several CPUs per CPS.

Role of DBA – Organizational Structure

- There are a number of centralized database system administrators.
- Database application administrators are usually in teams of 5 DBA's for groups of approximately 200 application programmers. For on going production support issues, 2 DBA's are on call, with a rotation system in place every two months.
- Company D also has a separate optimization and design group.
- For each functional area, DBA's and application developers report to the same director.
- Application DBA's support application developers. Usually, the DBA's pay more attention to review the SQL code from programmers who can benefit from a more experienced SQL technician.
- The company policy is to provide training to DBA's, although this also depends on the financial performance of the company. For example, no budget is currently available for training.

Performance Problems – Detection, monitoring, general policies/practices

- Tuning is driven by performance problems. The DB2 performance monitoring tool is used to compare the 'normal performance', and troubleshoot when sudden drops or anomalies are found.
- Focus on the critical path within a batch cycle. The batch cycles may contain hundreds of individual batch jobs.

- The optimization group also looks at CPU usage, and regularly identifies jobs that use CPU excessively.

Troubleshooting Performance Problems

- A lot of the discussion was based on the example of the billing system. The billing system has to retrieve the right data in order to create and send the right phone bills to Company D users. For the St. Louis area, 1.4 million usage records have to be read every night. Currently, this takes six hours, which is not a problem since there is time from 6pm till 6am. In California, a problem would occur since more usage records to be processed. Based on the processing time in St. Louis, Company D would need 85 hours. This means that a solution will have to be found.
- I/O is the most important trouble regarding performance. Basically, there are three ways to improve I/O:
    - Reduce I/O itself (e.g. by keeping the most frequently returning tables in memory).
    - Increase memory (for DB2, increase the 'bufferspace').
    - Use of 'pre-fetching'.
- Sorting also takes very long. Here, the solution is to make a good use of indexing. Note that sometimes even when fully indexed, sorting can still require considerable amount of time. As a last resort, it is sometimes possible/better to take the 'select' part and unload it, sort the flat file outside of DB2, and throw it back into the application once sorted. Of course this creates some overhead and is not always a solution.
- If nothing helps, another possibility could be to keep records in the *application* buffer, therefore avoiding going to DB2. Note that this is dangerous and can only be done for records that will not change for a given period of time.
- Partitioning can be used to let processes run in parallel. For the billing process for example, six partitions run in parallel to decrease the elapsed time.

Normalization – Denormalization

- Although in theory tables should be kept normalized (3NF), tables can be de-normalized when it is felt necessary to improve performance. There are no real standards about when to denormalize or not.
- The main issue is that when tables are in 3NF, this leads to smaller tables, which in turns requires multiple joins in order to return query results. These multilevel joins are rather inefficient in SQL.
- Usually, denormalized tables are in 2NF with partial key dependencies. Sometimes transitive dependencies are also encountered.
- Referential integrity is normally ensured through the RDBMS by defining referential integrity constraints (works especially well for the partial dependencies). In a few cases referential integrity is also enforced for transitive dependencies at the application level, although this is not recommended as it is better to keep applications more simple.

Various

- Sometimes, performance problems also occur just because of human errors. For example, one could forget that certain locks have been put on tables. The most common human error is utility jobs running out of sequence.
- There is not always enough attention given during the testing phase to identify potential performance problems. Another problem here is that testing usually runs in the background, which makes it hard to correctly evaluate potentially troublesome queries.

Additional Questions/Notes

- *DBA organization: Are DBA's assigned to development projects from the start?* Yes. *What are their main roles and when do they intervene?* The DBA is responsible for the physical design for the project. DBA's are also responsible for the test database creation and to assist with utilities needed for loading, testing, and conversions.
- *When comparing performance to 'normal performance' (see above), is this mainly a comparison with historical performance.* Yes, primarily.

If yes, who decides how long certain queries/jobs should take in the first place…? The project manager or team leader makes an initial determination on how much time is reasonable for each process.

On which criteria is this based? The amount of time allowable for each batch cycle determines the amount of time each component can take.

- *There is no real agreement on whether performance problems should occur with well-written SQL queries in transactional environments at all. Is it (always) possible to have a fully normalized transactional database without database performance problems if queries are written correctly?* There is no such thing as ALWAYS when discussing relational databases. It is certainly possible to have a 3NF database with such a small volume of data (number of rows in each normalized table), that performance problems would not be significant. But this would be predicated on how well the physical design had been implemented and what the definition of acceptable performance was. But it is possible, unlikely, but possible.
- *Is there a difference in the decision to denormalize tables for transactional environments versus reporting/data warehouse type of environments? If yes, which differences?* We do not normalize database designs for informational based systems, only transactional based systems. So data warehouse efforts use a modeling method known as "multi-dimensional" modeling. The mechanics of multi-dimensional modeling are built on denormalized practices. So, yes the activity employed for denormalizing transaction based systems is quite similar to the activity done to design informational based systems.
- *The example given about the billing system seems to be for reporting purposes.* No, perhaps I misled you during the interview. The purpose of the billing system is not only to produce customer bills, but to produce regulatory documents, management level reports and most importantly to keep the customer accounts up to date.

  Although 1.4 million usage records is a lot, is this the only explanation why this batch cycle requires 6 hours? This CYCLE requires 6 hours because we convert the electronic switching messages to mainframe based usage records. Then they are rated and posted to the correct customer account, company, and usage class. Next the required taxation is applied to the transaction records. Then these records are

processed against the customer database and billing records are created. These billing records are then processed to produce monthly bills, or service order changes to the field technicians and the federal, state, and local regulatory reports are produced.

- *Partitioning can increase speed, but also creates overhead. On what is the decision to use more/less partitions based? Is this trial and error? In practice, are there constraints on the number of partitions that can be used with DB2?* The decision to partition is usually based upon one of two things. First, is the table too large to exist on one partition? (This is a physical limitation of the hardware and software. Both of these limits are constantly being expanded.) Second, can the table be partitioned such that we can reduce the elapsed time for processing the total table? It is not trial and error because a lot of analysis is done to determine the cost/benefit of doing the partitioning. It is trial and error because there is no guideline as to how many partitions will work with each transaction process that may be used against the partitioned table. And processing characteristics are different for each circumstance. Yes, there are constraints to how effective this can be. But those are way too involved to begin discussing them here.

Company C	DBA-C	14 March 2003

Database System(s) - General

- Company C uses DB2 for the mainframe system, and mostly Oracle for the distributed systems. SQL server is only used in a few cases, on a departmental level.
- Oracle is used in St. Louis, and our interview would focus mainly on Oracle given J. Kender's expertise.
- The size of the databases amounts in the Terabytes of data. Financial applications take hundreds of Terabytes, Manufacturing applications tens of Terabytes.
- For IDS (part of Company C in St. Louis), there are 15 Oracle DBA's within the IS organization. An additional 5 to 10 external DBA's are usually also involved. A ratio to compare this with the number of programmers is not applicable, since third-party vendors are used (see below).

Role of DBA – Organizational Structure

- Company C relies mostly on commercial products; only very few applications are developed in-house. (e.g. Peoplesoft for financial and HR applications; Manugistics for manufacturing applications)
- The Datacenter is outsourced to IBM.
- Company C's role is to supervise the outsourcing relationship. In general, when database performance problems occur, vendors are asked to fix it.
- In case Company C develops in-house, the role of the DBA's becomes more proactive. The DBA is involved in design reviews, checks access requests, and walks through SQL code.
- Training is provided to DBA's (Quilogy/Wash. U). Training definitely works and is important. Company C does not hire straight from college; only experienced people are hired.

Performance Problems – Detection, monitoring, general policies/practices

- For the commercial applications, DBA's from Company C would monitor and identify performance problems. Usually, Company C also obtains the right from the vendors to perform indexing on the tables.
- Usually, performance problems do not arise from environmental issues (hardware, network…). Due to the value of the end product and importance of not having any delays, hardware upgrades are relatively easy to justify and obtain.
- When required to justify hardware upgrades: make a business case linking employee working time and delays to cost savings.
- Oracle Explain is used to identify cause of performance problems by DBA's:
  1) Identify which calls create the problems.
  2) For that call, analyze how the call was parsed/optimized.
- Monitoring is a part of every DBA's job. Oracle 'Statspack' runs once every hour. Reports are then examined once or twice a week.
- Threshold can be set using Statspack, for example:
  - Number of calls satisfied from buffer.
  - % served from SQL area (already parsed)
- Besides this, HP tools are also used to monitor Network, Database Server, I/O (here also with system administrators).

Troubleshooting Performance Problems

- Important to use right indexing. Depends a lot on specific situations.
- Increase buffers to reduce IO.
- (Upgrade hardware/system…. see earlier)
- After using explain, see if SQL code could be better… (See earlier)
- In case the SQL code is right, but the response time is still not satisfactory, parallel processing of queries could be an option. Multiple copies are launched at the same time and then brought back together. This creates some overhead, but the trade-off could be beneficial.

Normalization – Denormalization

- During the design phase, 3NF is the goal. However, sometimes denormalization occurs for performance reasons afterwards.
- Access requirements are reviewed to decide whether denormalization is required or not.

Various

Web Server and application server used. Oracle development tools are used for reporting. Apache & Tomcat on web server. J2EE. Also weblogic (BEA) and Websphere (IBM).

Company B	DBA-B	17 April 2003

.

Database System(s) - General

- Company B mostly use Oracle version 8i, and are planning to upgrade to Oracle 9i. They also use a little bit of SQL Server. They have DB2 on mainframe version 7.1. IDMS is also used.
- A lot of front-ends are in Cobol (CICS), but the company is also moving to web-oriented.
- Database Size: ~15 Terabytes (This is a total for all databases.)

Role of DBA – Organizational Structure

- DBA's and programmers report to separate directors, which report to a single senior director.
- Logical modelers are responsible for the logical design of the database design and document the businesses rules.
- Design DBA's (sometimes called application DBA's) are in charge of both the physical design and system performance.
- The application DBA's are involved in development project from the earliest stage till the end. DBA's act as teachers and also review the SQL code. Although database tuning is part of every DBA's job description, three DBA's have more advanced expertise in this area.
- Besides the design DBA's, system DBA's are in charge of more general database system issues.
- Training plays a very important role in Company B. It is conducted twice a year. Each person is reviewed.
- There are approximately 25 DBA's for 600 programmers (this would probably also include the logical modelers).

Performance Problems – Detection, monitoring, general policies/practices

- One of the big differences at Company B is that there are no performance problems – or at least, they are fixed.
- Performance problems are mostly because of bad SQL. It is rare that problems arise because of hardware issues. Hardware upgrades are not usually made for database performance issues.
- There is a lot of documentation. More specifically about performance design, normalization, and indexing,
- For Oracle, Quest Software, SQL Lab expert is used to identify bad SQL. There is an AI engine that gives suggestions. For DB2, Omegamon from Candle is used for monitoring and SQL tuning is mostly manual.
- Thresholds are set and there is an automatic monitoring and alert system when query response time is exceeded. In this case, problems are detected and fixed immediately. For example, the CICS threshold would be 300 milliseconds.

Troubleshooting Performance Problems

- Indexing is important. However, it has to be done judiciously. For instance, for 1 million rows it is a waste for indexing as more motion is created. Every insert, update & delete has to keep the index in sync i.e., create extra I/O.
  There really is no hard and fast rule. A good rule of thumb is that if you are going to retrieve/process more than 20% of a table, it should be scanned instead of using index access.
- Writing good SQL. This can be done by having more experienced people and/or regular training. You also need to perform reviews on all SQL going into production.
- Prioritizing of queries is also done.
- The bottom line is to spend more time up-front with DBA's overseeing the work. Although this could be a lot of work, it's certainly worth it. It is also important to fix detect and fix troublesome queries as soon as they occur.
- Note: usually slightly different queries can make a big difference. The following steps should be followed:
    - Establish timing goals.

- Identify which part is bad.
- Run explain, optimizer analysis tools…
- Try better queries and solve existing problems.

Normalization – Denormalization

- The third normal form is emphasized in the development stage itself.
- For a transactional database, 3NF should always be used. Denormalizing tables for performance reason is not recommendable.

Various

- CICS, Java are used as the front-end. Weblogic is used as the application server.
- No real cost justification; the rule is: when there is a problem, fix it immediately.

Additional Questions

- *During the interview, we focused on transactional systems. Do performance problems occur in data warehouse/reporting type of environments? In a typical data warehouse environment, dimensions are used and denormalization is common use - do you agree, and would you have any other comments regarding performance problems in a reporting/data warehouse environment?*

  The data warehouse is a different issue from transactional systems. We have set the expectation that, depending on the query, response times can be long. We do perform some denormalization in the warehouse to pre-calculate summary values and to place the data into Star Schemas.
- *One of the reasons mentioned for denormalization to improve performance was that multilevel joins could slow down queries when large quantities of data are involved. Any comments?*

  With a properly tuned database and properly written SQL on a transactional system, this is not a problem. There are many mistakes that DBA's and programmers make that create many of the problems that are blamed on normalized databases. You will note that I specifically mentioned 3rd normal form in our conversation. I have seen instances where 4th or 5th normal form has been used. These are, in my experience,

very bad for performance. I have seen tables that were performing poorly because the order of columns in a multi-column index was wrong. I have seen systems that perform poorly because they are over-indexed. This puzzled both the DBA and the programmer. They couldn't understand why a query was performing badly because 'it is using the index'!

Company E	DBA-E	7 May 2003

Database System(s) - General

- Various! DBA-E currently works for the Integrated Technology Services group. His background is more as a generalist, including experience in database systems and systems analysis.

Role of DBA – Organizational Structure

- Training is important and it helps. It has to be targeted at different levels for different people. People are very important to the organization. Good project management is required.
- Application DBA's should be involved in projects from the start. It becomes tough to have a good system if DBA's only get really involved with the project at a later stage. While in theory companies might have DBA's assigned from the start, in practice they could be thrown in only at a later stage for review and testing.

Performance Problems – Detection, monitoring, general policies/practices

- Note that as for any problem, it is important to go through a comprehensive analysis first to detect root causes for the problem. (Hardware, network, HR organization/project management, CPU/Memory/ bad programming…?)
- Performance problems are mostly because of bad SQL. One of the reasons is that hardware issues are easily detected and fixed.
- It is easy to write SQL but writing good SQL is not easy at all. It can take considerable time and effort to get used to writing good SQL.
- It is easy to detect and fix hardware issues. Bad SQL remains tougher to detect and rewrite.
- It is recommendable to do things right from the first time; once in production, rework should not be required. Spending time up-front to ensure good SQL code is certainly worth the effort.

Troubleshooting Performance Problems

- Monitoring tools, query analyzers and optimizers are useful.
- Prioritizing of queries is also done.
- Indexing.

Normalization – Denormalization

- For transactional systems, it is always advisable to use the third normal form. Often, normalization is blamed, while the real reason for performance problems could be that SQL-code is not written in the best way. (It is true that 3NF requires more complex SQL queries than denormalized tables).
- However, if the system is not fully normalized costs of redoing the project by making it fully normalized should be evaluated. A comprehensive database re-design requires huge resources (especially with the rework on existing applications), and business realities often make this impossible. Cost savings on a long term basis should be taken into account.

Company F	DBA-F	May 2003

Database System(s) – General

- DBA-F's main expertise is with data warehouse environments.
- Experience with DB2 and SQL server; OLAP, reporting tools (cognos, business objects).

Role of DBA – Organizational Structure

- There are huge benefits when database administrators gain a better functional knowledge.
- It is highly recommended that DBA or experienced data modeler be part of the project for all database designs and build phases. DBA's should be involved from the start of development projects.
- Database design should be developed in collaboration with the functional knowledge expert(s) based on requirements.
- Since database design is the basis for majority of applications, it is essential that the database is designed accurately and properly prior to any development of an application to minimize rework. As far as time spent on reviewing, it depends on the complexity of the model and knowledge of a person reviewing. You could spend a few hours to days or weeks. When we are asked to peer review codes or designs, we usually spend a few hours for SQL query reviews and up to a couple days for database designs.

Performance Problems – Detection, monitoring, general policies/practices

- Setting 'automated' threshold to monitor actual performance versus desired performance would be great… (cf. Company B). Typically, companies cannot afford full-time resources on monitoring alone.
- At Company F (data warehousing), checks are performed once a week (logs filled).

Troubleshooting Performance Problems

- Make sure to use the right archiving strategy. In other words, keep the right amount of historical data in your data warehouse (a few months, one year, several years…. depends on usage, which is why a good functional knowledge is important).
- A lot of problems occur due to wrong indexing. A typical example is when data and requirements change after the initial design, while the indexing is not adapted accordingly.

Normalization – Denormalization

Normalizing and denormalizing depends on the purpose of the system that you are building. Ideally, database should be normalized if it is used as an OLTP system and denormalized for reporting/decision support system.

If used for reporting, obviously adding more data will eventually decrease performance. Based on the requirements, you should incorporate an archiving strategy for the database.

Additional Questions

In a:

(1) normalized relational database,

(2) with well-written SQL code, and

(3) if indexing cannot be further used (i.e.: relevant columns already indexed), is it still possible to encounter performance problems?

Yes, again it all depends on the purpose of the system. If used for reporting, in most cases a large amount of data is consolidated. Therefore, you would want to minimize the joins to gain performance so your database should be denormalized. If it is transactional system, it should be normalized since you are processing single or very small amount of records at a time. Even with the well written SQL code, performance could decrease due to large volume of data in the database.

If yes, what are the solutions at this stage?

If everything is perfect as far as model, SQL code and if the database is properly maintained, data archiving strategy should be implemented. If this is not possible, you

should review the hardware specifications and LAN connection speed. Based on the analysis, the project could potentially add more CPUs, upgrade to faster HD and/or potentially modify the service level agreement with customers.

Company G	DBA-G	**22nd May 2003**

Database System(s) - General

- Kris works for Company G on an IT systems outsourcing contract with a garment manufacturing company in St. Louis.
- The Oracle software release used is Oracle 7.3.4. Kris has been working in IT for almost the last 30 years and as DBA for almost 20 years.
- Company G is migrating the customer to Oracle 8.1.7 in June 2003.
- The customer needs access to their systems 24x7 with the exception of an eight-hour maintenance window on Sunday morning.

Role of DBA – Organizational Structure

- There is no required training for a DBA in Company G. It is the employee's responsibility to develop and review their career plan with their manager. Training is encouraged and financed by Company G for all employees.
- The Oracle systems supported by Company G at this account were purchased from other vendors. There is a manufacturing application and Oracle's Financials application that Company G supports for the customer. There is some development, but the majority of the work is maintenance of these applications. The project development staff has a great deal of experience and most of them have been on this project over three years. Because of this experience, the DBA's are involved with developers only if developers are facing a problem or want DBA's to get actively involved. Walk-throughs are performed with DBA's and/or their peers for all modifications.
- There are three DBA's assigned to support this project. The primary tasks of each of these DBA's are as follows: one to support the Oracle system software, one to support the migration of business data from the legacy systems, and one to support the application area. They cross train with each other as well as with the other members of the infrastructure team that supports NT, application security, Unix, and automation.

Performance Problems – Detection, monitoring, general policies/practices

- DBA's, end-users, and developers identify performance problems and evaluate solutions. Code walk-throughs are performed to reduce the risk of further problems. There is a promotion process that is used to test all changes. The change is initially tested on the development system by the developer. It is then promoted to the integrated testing system and then to the quality assurance system before it is installed into the production system during the maintenance window.
- Database performance problems are mostly because of inefficient or untuned SQL. It is rare that problems arise because of hardware issues. Usually new developers have the problem with writing good SQL. It can be said that 80% of the problems are because of bad SQL
- Oracle's cost based optimizer is often used to resolve performance problems. The applications default to the rule based optimizer and implementing an Oracle hint can often result in dramatic improvements. One example is a SQL statement that took 2 hours, that now takes 10 seconds with the cost based optimizer.
- BMC perform predict is scheduled to be purchased and implemented. This product will be used to help to identify problems and to predict hardware capacity.

Troubleshooting Performance Problems

- It is important to use right indexing. Depends a lot on specific situations. A rule of thumb is not to exceed 10 indexes per table. Most of the tables in the manufacturing system have a primary key and referential integrity is enforced through Oracle. Most of the tables in Oracle Financials have a unique index and referential integrity is enforced through the application.
- Writing good SQL is key. This can be done by having more experienced people and/or regular training. This can also be done by including DBA's in the development process. The omission of a join clause and including a table that is not included in the join are common performance problems.
- Prioritizing of queries is also done.

Normalization – Denormalization

- During the design phase, 3NF is the goal. However, sometimes denormalization occurs for performance reasons afterwards.

Various

Visual Basic 5 is used in the front-end. This is the original design of the purchased applications. There are no plans to convert to any other software due to the cost involved versus the benefit. SQL sent to the Oracle software from Visual Basic has the literal values substituted for the host variables. This prevents Oracle from reusing the SQL in the shared pool and it must be re-parsed. The shared pool fills up and becomes fragmented, which negatively affects performance. Company G has implemented a flush of the shared pool every fifteen minutes to help alleviate this problem. There is a feature in Oracle 8i that will be researched as another possible solution.

Representative set of questions used for the interviews

- Which type(s) of Database is used at the company? Hardware? Front-ends used? What is the size and traffic (e.g.: # (concurrent) users, # transactions per second or per day, amount of data…)

- How is the internal organization with regard to database administration; what is the role of database administrators (and/or database architects)? Hierarchy: who do the DBA's report to? When do DBA's get involved into projects? How is the interaction between systems analysts – programmers – DBA's…?

- Does the company experience problems due to 'slow' response time to database queries? If yes, in how far is it a problem (how often, what is 'slow'), and how does the company handle this in general?

- Does the company have service levels and standards in place for response times? Is there formal tracking/monitoring of meeting these service levels? If yes, how? Are there specific procedures to handle user complaints?

- Different factors can affect performance: CPU, Memory, Disk Access (I/O), Network, Applications (poor programming), Database design, Database tuning…
  On which of these factors do you believe that the company might gain the best performance improvements? How and why?

- How do you 'value' increases in database performance? How would you make a cost-benefit analysis to justify investments in extra hardware, training, optimizing tools oriented towards the improvement of database performance?

- What are the procedures to detect and troubleshoot 'bad' queries?

- Can specific training help? If yes, for who (e.g. DBA's, application programmers), and which type of training?

- How are indexes used to improve performance (versus using disk space & increasing maintenance…)?

- Does the company uses any specific (external vendor) tools to increase SQL performance? If yes, which ones? How could this investment be justified, and how are the results? If no, have such tools been considered, and what were the conclusions?

- What is the role of a thoughtful database design? How does normalization work in practice? Are standards and procedures in place to ensure appropriate database architecture?

Appendices

Appendix 1: Third-Party Tools

Vendor + Website	Tools + Type	DBMS supported
Quest Software www.quest.com	***QuestCentral*** • Performance Monitoring • SQL Tuning (SQL Lab Expert)	Oracle, DB2, SQL Server
Candle www.candle.com	**Omegamon Series (and additional DB2 tools)** • Performance Monitoring	DB2
Lecco www.leccotech.com	**Lecco SQL Expert** • SQL Tuning (advanced)	Oracle, DB2, SQL server, Sybase
Embarcadero www.embarcadero.com	**DBArtisan** • Performance Monitoring • SQL tuning (limited)	Oracle, DB2, SQL server, Sybase
Precise Software www.precise.com	**Precise/Indepth** • Performance Monitoring • SQL tuning (limited)	Oracle, DB2, SQL server
BMC www.bmc.com	**SmartDBA, SQLExplorer, ARPtune…** • Performance Monitoring • SQL tuning (limited) (see tools and info on website)	Oracle, DB2, SQL server, Sybase, IMS, Informix

See also:

Komadina, M., "IBM Utilities versus Third Party Tools for DB2", 8 May 2003, available online at http://www.databasejournal.com/features/db2/article.php/10896_2203401_1

Appendix 2: Roadmap to debugging performance issues, initial analysis

Source: DB2 Problem Determination Tutorial Series, http://www-3.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/tutorial_performance.d2w/toc

Once you receive report of poor performance, ask the following questions to quickly determine the best place to start looking for a potential cause.

When did the problem first occur?
If the problem has been occurring for some time, and if a database monitor schedule has been implemented, you can use historical data to find differences. This will allow you to focus on changes in system behavior and then focus on why these changes were introduced. Monitoring is discussed later in this tutorial.

Is the performance issue constant or intermittent?
If the poor performance is continual, check to see if the system has started to handle more load or if a shared database resource has become a bottleneck. Other potential causes of performance degradation include increased user activity, multiple large applications, or removal of hardware devices. If performance is poor only for brief periods begin by looking for common applications or utilities running at these times. If users report that a group of applications are experiencing performance issues, then you can begin your analysis by focusing on these applications.

Does the problem appear to be system-wide or isolated to DB2 and its applications?
System-wide performance problems suggest an issue outside of DB2. It's possible that something at the operating system level needs to be addressed. If the poor performance is confined to DB2 applications, obviously we'll focus on what DB2 is doing.

If isolated to one application, is there a particular query that appears to be problematic?
If one application is problematic, then you can further examine if users are reporting a query or set of queries that are experiencing a slowdown. You might be able to isolate the issue to one application and a potential group of queries.

Is there any commonality to the poor performance or does it appear to be random?
You should determine if any common database tables, table space, indexes, etc. are involved. If so, this suggests that these objects are a point of contention. Other areas to potentially focus on are referential integrity constraints, foreign key cascades, locking issues etc.

These brief initial questions are invaluable in developing an action plan to help debug a performance problem.

Appendix 3: SQL Optimization Examples

Example 1: Reducing SQL Calls.

Source: Encore System Consulting Co., available online at
http://www.en-core.com/eng/resources/F51_2_3.asp?dtn=0002W

The section introduces very useful tips related to E-commerce.

Product Category

Number	ProductName	Description	Amount
1	A	A description	100
2	B	b description	100
3	C	c description	100
4	D	d description	100
5	E	e description	100
6	F	f description	100
7	G	g description	100
8	h	h description	100
9	i	i description	100
....	....	....	....

In the product categories above, if the lines in yellow are the products that a customer has chosen, those products will be stored in the table labeled as`shopping cart.'

`Shopping Cart' Table

Number	Customer	ProductName	Qty	Amount
1	John Doe	B	1	100
2	John Doe	C	1	100
3	John Doe	D	1	100
4	John Doe	E	1	100
5	John Doe	F	1	100
6	John Doe	G	1	100
7	John Doe	H	1	100
8	John Doe	I	1	100

If a customer called `John' changes the qty of item 1,2,3,4,5,6,7, it will be shown as below.

Number	Customer	ProductName	Qty	Amount
1	John Doe	B	3	100
2	John Doe	C	4	100
3	John Doe	D	2	100
4	John Doe	E	5	100
5	John Doe	F	10	100
6	John Doe	G	2	100
7	John Doe	H	3	100
8	John Doe	I	1	100

Most developers will update the table using the following SQL statement to modify the quantity of items that need updating.

```
update ShoppingCart
set Qty = ;a1
where Number = ;s1
```

If you just look at the SQL statement, there is no need to tune the table. However, from the application's perspective, you should know that DBMS is called 8 times.

The table needs to be updated 8 times because the DBMS must find the items that were not changed in order to update the other 7 items, although only 7 items have changed. This means that DBMS is called 8 times to update the whole table and maintaining consistency. If one SQL statement as shown below is used to update the items changed by John, you can reduce the number of times DBMS is called. If static SQL is used, the items updated by one SQL are limited to 20.

 # DYNAMIC SQL

If using Oracle

```
  update ShoppingCart
        set Qty = (decode(Number,1,3,2,4,3,2,4,5....)
  where Number in (1,2,3,4,5...)
```

If using MS SQL

```
update ShoppingCart
set Qty = case Number
            when 1 then 3
            when 2 then 4
            when 3 then 2
            when 4 then 5
            ....
            End
where Number in (1,2,3,4...)
```

 # STATIC SQL

If using Oracle

```
Update ShoppingCart
      set Number = (decode (Number,;a1,;v1,;a2, ;v2,;a3, ;v3,;a4, ;v4....,;a20, ;v20)
where Number in (:a1,:a2,:a3,:a4,:a5...,;a20)
```

If using MS SQL

```
update ShoppingCart
set Qty = case Number
            when @a1 then @v1
            when @a2 then @v2
            when @a3 then @v3
          ....
             when @a20 then @v20
           end
where Number in (@a1,@a2,@a3,@a4...,@a20)
```

Example 2: Modifying where clause ordering

Source: LANSING, M., "Turbo Charge Your SQL", Powertimes: International Journal for Application Developers, December 2002.

For example, the following 'inefficient' SQL code could be detected (in this case using a third-party tool)

```
Offensive SQL Statement
Full table scan with table size larger than
the Offensive SQL Full Table Scan Threshold
(1000 Kbytes).(schema.tablename)
SQL Start Position : Line = 1526, Column = 6
(Package Body)
```

Here is the original SQL code:

```
SELECT pa.partypartyadid partypartyadid,
l.country_locationid
country_locationid,
c.departmentid departmentid,
p.partytype partytype,
p.name name,
c.siteid siteid,
c.ctrno ctrno,
e.execno execno,
c.purchasesaleind purchasesaleind
FROM zzcontract c,
zzexec e,
zzpartyad pa,
zzparty p,
zzlocation l
WHERE c.ctrno = e.ctrno
and c.siteid = e.siteid
and c.canceledind = 'N'
and e.canceledind = 'N'
and e.execstatus = 'O'
and c.counter_partyadno =
pa.partyadno
and pa.partyno = p.partyno
and pa.locationid = l.locationid
and c.commodityno = p_commodityno
and l.canceledind = 'N'
```

In this specific case, changes in the ordering in the 'where' clause would be able to improve the response time by a factor 10. From 1.35 seconds to 0.12 seconds. Below is the modified SQL code:

```
SELECT pa.partypartyadid partypartyadid,
l.country_locationid
country_locationid,
c.departmentid departmentid,
p.partytype partytype,
p.name name,
c.siteid siteid,
c.ctrno ctrno,
e.execno execno,
c.purchasesaleind purchasesaleind
FROM zzcontract c,
zzexec e,
zzpartyad pa,
zzparty p,
zzlocation l
WHERE e.ctrno = c.ctrno
AND e.siteid = c.siteid
AND 'N' = c.canceledind
AND 'N' = e.canceledind
AND 'O' = e.execstatus
AND pa.partyadno =
c.counter_partyadno
AND p.partyno = pa.partyno
AND l.locationid = pa.locationid
AND p_commodityno = c.commodityno
AND 'N' = l.canceledind
AND c.canceledind = e.canceledind
AND c.canceledind = l.canceledind
AND e.canceledind = l.canceledind
```

Example 3: Dividing the SQL plan based on the type of condition

Source: Encore System Consulting Co., available online at

www.en-core.com/eng/resources/F51_2_3.asp?dtn=0002V

KEY WORDS

- Dividing SQL statements, use of UNION ALL, dividing the execution plan.

DESCRIPTION

- The following SQL statement groups and extracts rows per department from a table managing some activities for a telecommunication company.

```
 select dept_cd,sum(NVL(stock_amt,0))
into :b0,:b1
from tb_lms02 A
where A.dept_cd like :b2||'%'
  and A.stc_date between :b3 and :b4
  and A.item_no between :b5 and :b6
  and A.maker_code like DECODE (:b9,'','%',:b9)
  and A.stock_amt_new <>0
group by dept_cd
```

```
call     count    cpu   elapsed    disk    query  current  rows
------- ----------- -------- ---------- ---------- -------- --------- -----------
Parse        6      0.05       0.05            0        0       0       0
Execute      6      0.03       0.03            0        0       0       0
Fetch        6   1536.33    1844.25     123774   322313       0       1
------- ----------- -------- ---------- ---------- -------- --------- -----------
total       18   1536.4     1844.33     123774   322313       0       1
```

```
Rows     Execution Plan
-------  ---------------------------------------- ----------------------------------------
      0  SELECT STATEMENT  OBJECTIVE: FIRST_ROWS
      7   SORT (GROUP BY)
     72    TABLE ACCESS   OBJECTIVE: ANALYZED (BY ROWID) OF 'TB_LMS02'
2993860     INDEX   OBJECTIVE: ANALYZED (RANGE SCAN) OF 'TX_LMS02_PK'
(UNIQUE)
```

- Index
  - TX_LMS02_PK : ITEM_NO+ STC_DATE + SEQ
  - TX_LMS02_03 : DEPT_CD + STC_DATE + MAKER_CODE

- If you look carefully at the above SQL statement, the condition that can be used as the driving condition is used as the range condition. In this case, the optimizer selects only one driving condition and uses the rest of them as the check condition. In the above example, the optimizer has selected the pan of using the 'TX_LMS02_PK' Index. However, it has resulted in random access to all rows by the index in the table. It has likewise worsened the performance speed. The SQL statement above was intended to use`item_no,'but but instead,`dept_cd' was used.

- You can find this case in companies under this situation. However, it is impossible to make the execution plan, using one SQL statement, that uses different indexes depending on the user's given condition. Of course, you can solve this problem by implementing a dynamic SQL statement that creates SQL statements depending on the pertinent conditions. However, you cannot avoid serious overhead because you cannot share identical SQL statements in different conditions. In this case, you need to divide the execution plan depending on the type of condition employed by the user. In the example above, you must divide the cases that use `dept_cd' and `item_no.'

- Look at the following SQL statement.

```
  select /*+ index(a tx_lms02_03) */
       dept_cd,nvl(sum(stock_amt),0)
  from tb_lms02 A
  where :gbn = 1
    and A.dept_cd = :b1
    and A.stc_date between :b2 and :b3
    and A.item_no between :b4 and :b5
    and A.maker_code like DECODE(:b6,'','%',:b6)
    and A.stock_amt_new <>0
  group by dept_cd
union all
  select /*+ index(a tx_lms02_pk) */
       dept_cd,nvl(sum(stock_amt),0)
  from tb_lms02 A
 where :gbn = 2
  and A.dept_cd like :b1||'%'
    and A.stc_date between :b2 and :b3
  and A.item_no = :b4
  and A.maker_code like DECODE(:b6,'','%',:b6)
  and A.stock_amt_new <>0
group by dept_cd
```

```
------------------------------------------ ------------------------------------------
  0     SELECT STATEMENT Optimizer=FIRST_ROWS
  1   0   UNION- ALL
  2   1     SORT (GROUP BY NOSORT)
  3   2       FILTER
  4   3         TABLE ACCESS (BY ROWID) OF 'TB_LMS02'
  5   4           IND EX (RANGE SCAN) OF 'TX_LMS02_03' (NON-UNIQUE)
  6   1     SORT (GROUP BY NOSORT)
  7   6       TABLE ACCESS (BY ROWID) OF 'TB_LMS02'
  8   7         INDEX (RANGE SCAN) OF 'TX_LMS02_PK' (UNIQUE)
```

- Response time was reduced to 0.3 or less second when we used the above method, no matter which `item_no' or `dept_cd' was used as a condition.

Appendix 4: SQL optimization tips (SQL server)

Source: Chigrik, A., "Transact-SQL Optimization Tips ", Databasejournal, July 9, 2002, available online at www.databasejournal.com/features/mssql/article.php/1437391

1. Try to restrict the queries result set by using the WHERE clause.
This can result in a performance benefit, as SQL Server will return to the client only particular rows, not all rows from the table(s). This can reduce network traffic and boost the overall performance of the query.

2. Try to restrict the queries result set by returning only the particular columns from the table, not all the table's columns.
This can result in a performance benefit as well, because SQL Server will return to the client only particular columns, not all the table's columns. This can reduce network traffic and boost the overall performance of the query.

3. Use views and stored procedures instead of heavy-duty queries.
This can reduce network traffic as your client will send to the server only stored procedures or view name (perhaps with some parameters) instead of large heavy-duty queries text. This can be used to facilitate permission management also, because you can restrict user access to table columns they should not see.

4. Whenever possible, try to avoid using SQL Server cursors.
SQL Server cursors can result in some performance degradation in comparison with select statements. Try to use correlated subquery or derived tables, if you need to perform row-by-row operations.

5. If you need to return the total table's row count, you can use an alternative way instead of the SELECT COUNT(*) statement.
Because the SELECT COUNT(*) statement makes a full table scan to return the total table's row count, it can take an extremely long time for large tables. There is another way to determine the total row count in a table. In this case, you can use the sysindexes system table. There is a ROWS column in the sysindexes table. This column contains the total row count for each table in your database. So, you can use the following select statement instead of SELECT COUNT(*):
SELECT rows FROM sysindexes WHERE id = OBJECT_ID('table_name') AND indid < 2
This way, you can improve the speed of such queries by several times. See this article for more details: Alternative way to get the table's row count.

6. Try to use constraints instead of triggers, whenever possible.
Constraints are much more efficient than triggers and can boost performance. So, whenever possible, you should use constraints instead of triggers.

7. Use table variables instead of temporary tables.
Table variables require fewer locking and logging resources than temporary tables, so table variables should be used whenever possible. The table variables are available in SQL Server 2000 only.

8. Try to avoid the HAVING clause, whenever possible.

The HAVING clause is used to restrict the result set returned by the GROUP BY clause. When you use GROUP BY with the HAVING clause, the GROUP BY clause divides the rows into sets of grouped rows and aggregates their values, and then the HAVING clause eliminates undesired aggregated groups. In many cases, you can write your select statement so that they will contain only WHERE and GROUP BY clauses without the HAVING clause. This can improve the performance of your query.

9. Whenever possible, try to avoid using the DISTINCT clause.
Because using the DISTINCT clause will result in some performance degradation, you should use this clause only when it is absolutely necessary.

10. Include SET NOCOUNT ON statement into your stored procedures to stop the message indicating the number of rows affected by a T-SQL statement.
This can reduce network traffic, as your client will not receive the message indicating the number of rows affected by a T-SQL statement.

11. Use select statements with the TOP keyword or the SET ROWCOUNT statement if you need to return only the first n rows.
This can improve performance of your queries, as a smaller result set will be returned. This can also reduce the traffic between the server and the clients.

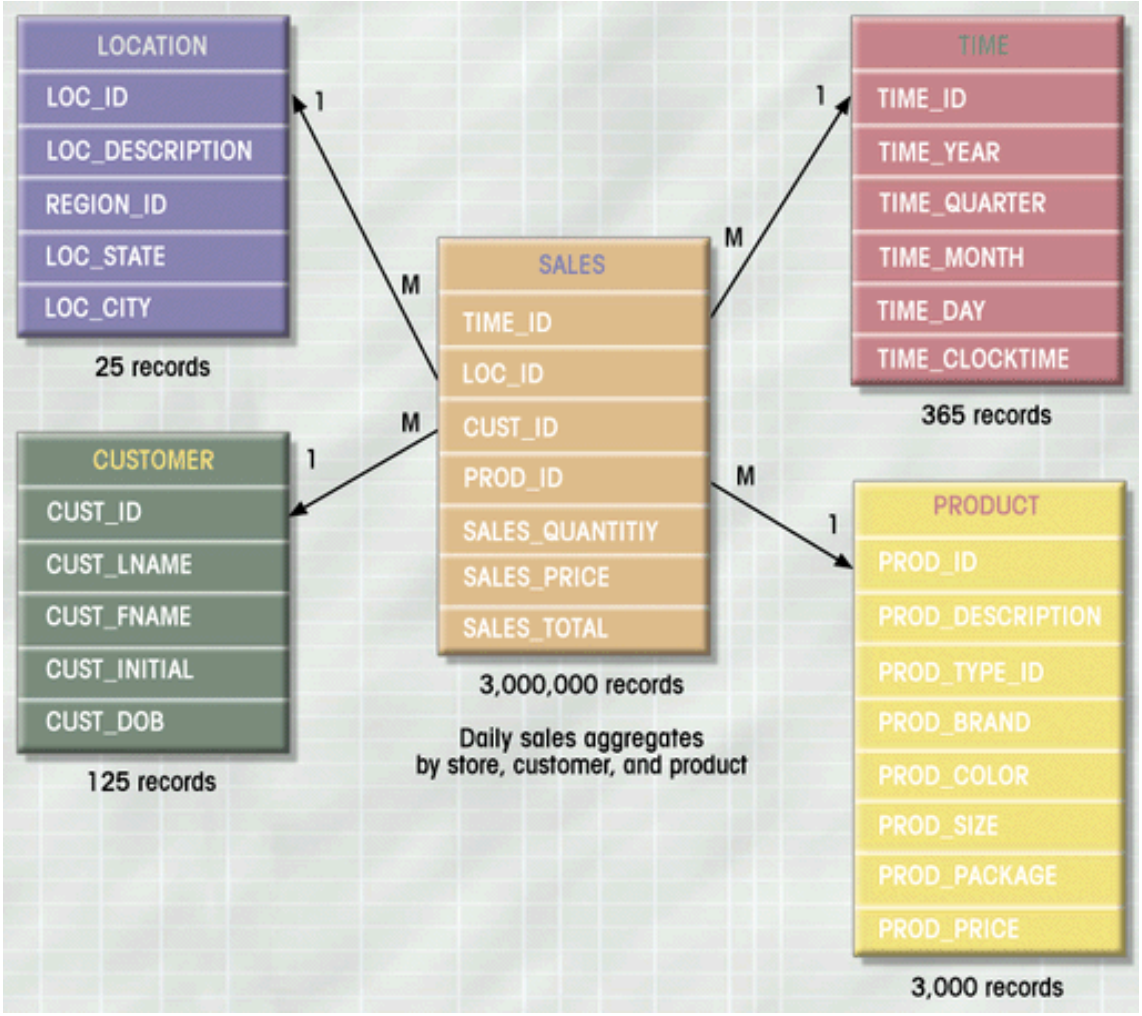
12. Use the FAST number_rows table hint if you need to quickly return 'number_rows' rows.
You can quickly get the n rows and can work with them when the query continues execution and produces its full result set.

13. Try to use UNION ALL statement instead of UNION, whenever possible.
The UNION ALL statement is much faster than UNION, because UNION ALL statement does not look for duplicate rows, while the UNION statement does look for duplicate rows, whether they exist or not.

14. Do not use optimizer hints in your queries.
Because the SQL Server query optimizer is very clever, it is highly unlikely that you can optimize your query by using optimizer hints; more often than not, this will hurt performance.

Appendix 5: Fact tables and dimensions

Source: ROB, P. and C. Coronel. "Database Systems: Design, Implementation and Management", Course Technology, 2002, pp.641-652.



The above figure represents the STAR schema. The STAR schema is data-modeling technique used to map multidimensional decision support into a relational database. Two key concepts are fact tables and dimensions. Facts are numeric measurements (values) that represent a specific business aspect or activity, such as sales. Facts are typically stored in a fact table, which is central in the star schema. Dimensions are characteristics that provide additional perspective to a given fact. For example, sales could be viewed by locations, customers, products, or time (see figure).