Craig S. Mullins & Associates, Inc.

Database Performance Management

November 1998

Computing : News&Review

SQL Analysis and Review

By Craig S. Mullins

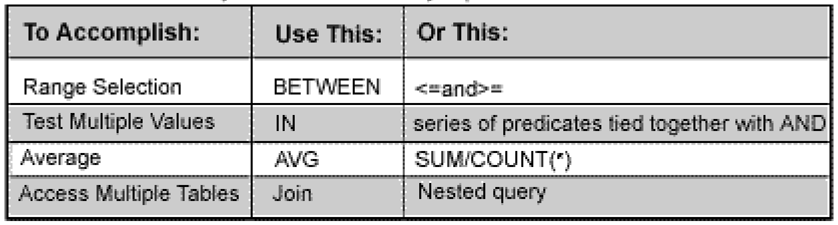
All access to relational data by application programs is done using SQL. Whether it is embedded directly in an application, issued *ad hoc* using a query tool, or coded using an API like ODBC, the underlying access is SQL. Therefore, SQL reviews should be a necessary component of both pre- and post-implementation performance analysis for database applications. Indeed, most experts agree that 70% to 80% of all performance problems in database applications can be traced to poorly written SQL code.

SQL is a very powerful high level language that provides a greater degree of abstraction than do procedural languages such as C++, COBOL, or Visual Basic. Procedural languages require the programmer to navigate the data structures. In other words, program logic must be coded to proceed record by record through the data in an order determined by the application programmer. This information is encoded in the high level language using multiple statements and is difficult to change once it has been programmed. SQL, on the other hand, is designed to allow the programmer to specify what data is needed, not how to retrieve it. It is coded without any embedded data navigational instructions.

How then does the DBMS determine which data to retrieve for a given SQL statement? This is accomplished by the relational optimizer. The relational optimizer analyzes SQL statements and fashions the data navigational instructions "behind the scenes." These data navigational instructions are commonly referred to as access paths. Access paths can be thought of as mini-programs, under the control of the DBMS, that contain specific data access instructions. The DBMS uses the access paths each time a SQL statement is run.

In addition, SQL is a flexible language. SQL statements can be formulated in a number of different, and functionally equivalent manners. This is possible because SQL provides the ability to code a single feature in several ways. One example of this is its capability to access multiple tables in a single statement either by joining tables or nesting queries. A nested query can always be converted to a functionally equivalent join. Some other examples of this flexibility can be seen in the vast array of similar functions and predicates, some examples of which are shown in the table below:

Table 1. Different Ways to Code Funtionally Equivalent SQL



To Accomplish:	Use This:	Or This:
Range Selection	BETWEEN	<=and>=
Test Multiple Values	IN	series of predicates tied together with AND
Average	AVG	SUM/COUNT(*)
Access Multiple Tables	Join	Nested query

The actual performance of these options can fluctuate wildly, but the actual data returned to the application can be equivalent. For this reason, it is imperative that the best option be used, thereby ensuring optimal performance. A SQL review, conducted by experienced performance analysts, can catch these types of potential performance problems and more.

There are many different access paths that can be chosen for any given SQL statement. For example, an index can be used to quickly locate the rows needed, or the entire table could be read, row by row, looking for the specific values requested. Likewise, when multiple tables are accessed in a single SQL statement the tables can be combined in any order and still return the correct results. The performance, however, will vary greatly depending on the volume of data in the tables, the nature of the request, and the indexes available for the table.

RDBMS optimizers usually do a good job of choosing the most efficient access path, but not always. The exact access paths chosen are crucial to the overall performance of any application that accesses a database. As such, a method to monitor and tune access path selection is necessary.

SQL Expert Systems
In addition to policies and procedures for SQL analysis and review, automated tools can be used to minimize the amount of bad SQL code. One of the most useful of these tools is the SQL expert system.

SQL expert systems provide the ability to quickly analyze all SQL in an application program and provide an in-depth explanation of the data access, as well as suggestions for improving the efficiency of the SQL. All of this is provided in an easy to read, textual report. The SQL expert system uses the ?explain? or ?show plan? capabilities of the RDBMS coupled with an expert system of SQL performance rules.

It is important for the SQL expert system to clarify the explain results because the native capabilities provided by the RDBMS can be somewhat difficult to master. Depending upon the RDBMS being used, the system-provided ?explain? or ?show plan? command obtains basic access path information for SQL statements. However, this basic information is either codified and placed into a table or produced textually in a cryptic format. Neither option is an optimal format for supporting SQL tuning.

At the heart of the SQL expert system is a knowledge base of SQL efficiency rules. These rules are based upon SQL efficiency guidelines for the DBMS in question. The SQL expert system reads the SQL statement, compares it to the efficiency rules at its disposal, and generates reports containing in-depth explanations and recommendations. There are three different types of rules that a SQL expert system can use:

SQL Design Rules — These rules are designed to flag potential problems found in application SQL statements. For example, the SQL rules will help you identify poorly coded predicates, statements that are not using an index, and predicate evaluation.

Physical Design Rules — These rules are designed to flag potential problems with the physical database design. For example, disorganized indexes and improper DDL parameters.

DBMS Design Rules — These rules are designed to flag potential problems with features specific to the DBMS. For example, DB2 supports plans and packages that are encapsulated access paths for one or more application programs. Any DB2 SQL expert system should have rules that understand DB2's plan and package options and parameters.

The SQL expert system, of course, is only as good as the expert rules it provides. Most SQL expert systems allow customization of the expert rules. Customization enables organizations to fine tune the rules to site-specific standards.

Another feature typically offered by SQL expert systems is versioning. By keeping multiple versions of the SQL analysis information and the recommendations provided, users can produce historical reports and use the data for future performance and testing needs.

Using a SQL expert system can result in a reduction in the amount of time, effort, and human error involved in SQL tuning. However, the most important benefit will be efficient SQL resulting in efficient applications.

Summary
SQL is at the heart of today's modern database applications. Poorly coded or ill-conceived, application performance will suffer, and business processes will be negatively impacted. Wise organizations will implement policies, procedures, and tools to create efficient and effective SQL. Failure to do so will impact the company's bottom line-and quite possibly your job security.

From *Computing News&Review*, November 1998.