8-16-1996

Online Analytical Processing (OLAP) Tuning and Performance Issues: A Study Using Microsoft Access

John A. Hoxmeier
Colorado State University

David Schoch
Sandia National Laboratories

Steven Jujan
Anderson Consulting

Online Analytical Processing (OLAP) Tuning and Performance Issues: A Study Using Microsoft Access

John A. Hoxmeier, Colorado State University
David Schoch, Sandia National Laboratories
Steven Lujan, Anderson Consulting

A database is designed to be a self-describing collection of integrated records which provides data integrity and security, as well as other important design objectives, but its most desirable feature is good performance [Khoshafian, 1992]. Database performance is a critical factor in the success of an OLAP database application, yet benchmarks and other performance tuning information for desktop or departmental database management systems being used to manage OLAP applications are almost non-existent.

A series of performance studies was run on a production OLAP database using the Microsoft Access database management system. The goals were twofold: to determine the efficacy of using Access to construct and manipulate a relatively large database that would be representative of a departmental OLAP application and to evaluate the impact of Access version and operating system upgrades on baseline performance comparisons. One would presume that adding additional memory resources, increasing the system parameters, and upgrading the tool set would improve system performance leading to better database performance.

With the release of Windows 95 and the performance improvements it claims to provide, the user must now determine what operating system, mixed with user-controlled variables, will provide optimal performance. With Windows 95 comes the expectation that the operating system will enhance the performance of existing applications.

The Application and Database Design

The database used in this research was developed for a large U.S. manufacturer to maintain customer registration data. Registration cards are packaged with various products. The registration cards included typical demographic and marketing data as well as questions regarding the purchase and use of the product. The responses to the questions were grouped, for purposes of this databases, as single and multiple responses. Single responses were stored as 1:1 with the personal data such as name and address in a single table called Registration. Multiple response questions (1:M) were handled as separate tables. The database consisted of multiple tables with approximately 1,000,000 records. The database is a good and fairly typical example of a departmental OLAP application.

Parsaye describes the OLAP environment as consisting several *data spaces* [Parsaye, 1996]. According to Parsaye, traditional OLTP applications deal only with the data space, while OLAP applications must deal with the data space as well as the much more challenging tasks of aggregation, influence, multidimensionality, and variation. A suite of queries was designed that would analyze the data in a variety of ways including discrete data queries (What is the serial number of a given Product), multidimensional queries (Product by Type broken down by Region), aggregation queries (Likert-type Satisfaction Response Averages by CustomerType) and influence queries (What factors influence purchase decisions). The query suites were designed in Access SQL and run from an Access Basic module. Access SQL supports the non ANSI-standard verbs, *pivot* and *transform,* which make it possible to perform the cross-tabulations required for multidimensional query support.

The Configurations

The query suites were run against the database in designed combinations of system configurations. The hardware environment for all tests within the study consisted of an 80486 DX2/66 with memory from 8-20 Mbytes, IDE hard drives (1Gbyte), Disk Accelerator, Access and the database files stored on first HD, Windows swap file stored on a second HD, and 30 pin, 70 nanosecond RAM.

The configuration variables included:

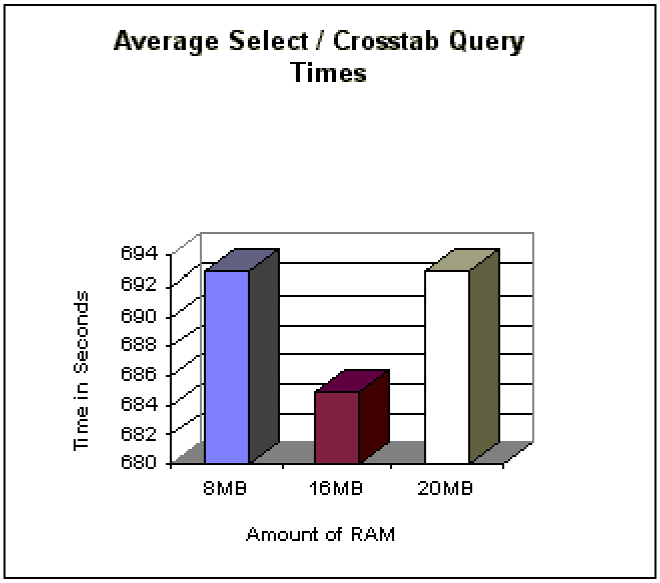
- RAM sizes of 8MB, 16MB, and 20MB
- SmartDrive sizes of 0K, 512K, 1024K, and 2048K
- MaxBufferSize settings of 512K, 1024K, and 2048K

All combinations of RAM, SmartDrive, and MaxBufferSize were tested.

The query suite included ten different queries representative of the analytic mix described above. Well over 1,500 query timings were captured. The query times were averaged by RAM size, SmartDrive size, and MaxBufferSize, and graphed to illustrate performance tendencies within categories and to show relationships to certain types of queries. In addition, a second suite of queries was developed to perform updates representative of aggregation type queries. The second suite also included index creation/drop to factor that type of operation into the analysis. An internal timer captured elapsed time and the suites were run several times for each configuration.

The Results

The results of the first series of tests indicated that, for the test database on the test system, increasing resources did not necessarily lead to an improvement in performance and that the optimum configuration was highly dependent upon the type of database and its query mix. The graph below illustrates the comparison of RAM on the first query suite. While not statistically significant, the results of this study indicated that the amount of available physical memory had very little to do with query performance, although update queries benefited by the availability of RAM more than the select and crosstab queries.



The effect of SmartDrive on the select and crosstab query performance was unpredictable at best, but as with changes in RAM size, changes in SmartDrive size appeared to have more influence on the update suites than it did on the select and crosstab suites. Performance suffers dramatically, however, without the SmartDrive cache. The optimum configuration for SmartDrive appeared to be 512K, with minor improvements in update times and sometimes reduced performance in queries as this parameter was increased.

MaxBufferSize's effect on query performance was perhaps the most easily recognizable and predictable performance pattern. Without exception, every increase in MaxBufferSize led to a corresponding increase in update query suite performance. In many instances, the marginal difference in update times between MaxBufferSize settings was enough to negate select and crosstab suite performance. inconsistencies, thereby leading to an overall performance increase as MaxBufferSize was increased.

In general, considering both versions of Access and both Windows 3.11 and 95, the best configuration appeared to be a combination of 16 megabytes RAM, 512K MaxBufferSize, and 1024K SmartDrive size (3.11). Additional resources not only did not necessarily contribute to performance, but in many cases degraded it. In several of the tests, especially with 3.11, the 8 MB configuration outperformed the configurations with more memory.

Access 2.0 vs. 7.0

In addition to evaluating the system configurations, the study set out to determine what versions of Microsoft Windows and Access would provide optimal performance under the different query conditions. The configurations included:

- Access 2.0 running on Windows 3.11 (Dos 6.2)
- Access 2.0 running on Windows 95
- Access 7.0 running on Windows 95

Although there was an increase in overall performance when comparing Access 2.0 under Windows for Workgroups 3.11 to Windows 95, the improvement is not significant. In the case of the select and crosstab queries, performance actually degraded. Select and crosstab queries create a dynaset, a dynamic set of data that can be viewed or set to print as a report which adds to the sheer amount and complexity of work the system must perform in order to execute the query [Irwin, 1994]. Windows 95 does not execute the select and crosstab query suite faster than even the worst Windows for Workgroups 3.11 configuration.

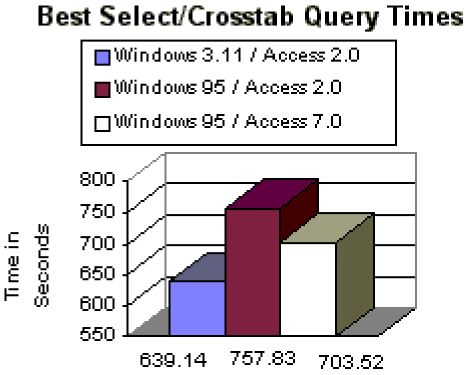
Update queries running on 95 for Access 2.0, on the other hand, outperformed Windows for Workgroups 3.11 significantly. The only explanation is that Windows 95 is able to utilize the dynamic SmartDrive size on queries that make changes to multiple records at a time, or action queries, more efficiently because action queries do not create dynasets and do not bog down the system like select and crosstab queries do. Unfortunately, Microsoft itself has little information regarding this phenomenon.

Although Windows 95 outperformed Windows for Workgroups 3.11 as a whole, not all Microsoft Access 2.0 databases can be generalized to follow this example. A database that utilizes a majority of select and crosstab queries would deliver better performance under Windows for Workgroups 3.11, but a Microsoft Access 2.0 database that utilizes mostly update queries is more optimal under Windows 95. Basing the optimal performance of each query run on Windows for Workgroups 3.11 on the data from Windows 95, some consistency can be seen. By setting the SmartDrive disk cache size at 512K, and the MaxBufferSize at anywhere from 512K to 2048K, a user can closely configure Windows for Workgroups 3.11 to perform close to or better than Windows 95 depending on the type of query.

As was expected, Access 7.0 (95) outperformed Access 2.0 (3.11). The primary reason for the increase in performance of the update queries is the new JET database engine in Access 7.0. The engine has been rewritten in Visual Basic 4.0 as a 32-bit application, and according to Microsoft, "it is specifically optimized for action queries" [Microsoft Support Line, 1996]. Action queries are queries that let the user create new tables or change data in existing tables by using delete, update, and append [Irwin, 1994]. In action queries, changes can be made to many records during a single operation, as opposed to Select queries where changes are made one record at a time. The new JET engine most likely has little effect on the crosstab queries because they are calculated or computed by individual cells, not many cells at one time. Unfortunately, a 12 percent increase in performance of the update suite from a new, 32-bit JET engine is disheartening. The new JET engines was built with action queries in mind but it had little effect on the queries in this study.

An interesting aspect of Access 7.0 was the speed in which it displayed the tables. The Many Response table, made up of over 300,000 records, could be displayed in around five seconds. Under Access 2.0 however, it took almost 20 seconds to generate the same display.

As was expected, Windows 95 outperformed Windows for Workgroups 3.11 when the database used was Access 2.0, and Access 7.0 outperformed Access 2.0 when both databases were run under Windows 95. Unfortunately though, the increases in performance were hardly significant, and probably not enough so to warrant every user to rush out to purchase upgrades. In the case of a user who would rather stay with Windows for Workgroups 3.11 instead of opting for Windows 95, they are able to configure their system in a manner that delivers performance close to or even better than Windows 95 as illustrated below .



On the other hand, the user who is upgrading to Windows 95 and Access 7.0 can still take advantage of performance increases, depending on the type of queries being performed.

In the case the user has many select and crosstab queries within the database, Windows for Workgroups 3.11 would be the best decision, seeing as how they would be losing performance by moving to Windows 95 or Access 7.0. However, if the database primarily uses update queries, Windows 95 would be the better choice based on the data.

Response Time

Elapsed times for the discrete select queries ranged from 1 to 25 seconds. The more complex multidimensional cross-tabulation query (which involved several joins) responses ranged from 1 to 8 minutes. Updates took up to 15 minutes including adding and dropping indices. While the timeframes for these activities are acceptable to the company and probably acceptable for a (fairly) large departmental OLAP application, some observations are worth noting:

- Database table structures can have a significant impact on query performance. Several smaller tables (3 X 250,000) will execute queries significantly faster than one larger table (1 X 750,000).

- A properly configured system (combination of memory, MaxBufferSize, SmartDrive parameters) can reduce most queries by as 25% and most updates by over 50%.
- The time it takes to index and then query an attribute is essentially the same as the time it takes to query the attribute without an index. Of course, if the attribute will be queried repetitively, the time to index is worth taking. Conversely, updates run approximately 3 times faster on a nonindexed attribute.
- Access was able to handle the volume of this 1,000,000 record application. Whether a user can wait 15 minutes for a multidimensional query to respond is, of course, completely dependent on the user's environment.
- Once queries had been run, they executed quickly if the system was not rebooted. Although the constructs of the study required the system to be rebooted between tests, this would probably not be the case in most OLAP applications.

In terms of improvement expectations based on the literature, the results of this study are disappointing. However, users of OLAP applications must accept the fact that response time delays are inevitable. The queries that make up OLAP applications are designed to return information that represent sets of records that incorporate unanticipated views, a fact that contributes to slower response times than one would associate with an OLTP system.

References

Irwin, Michael R., and Cary N. Prague, Microsoft Access 2 Bible. 2nd ed. (Cal: IDG Books Worldwide, 1994).

Khoshafian, Setrag, Arvola Chan, Anna Wong, and Harry K. T. Wong. A Guide to Developing Client/Server SQL Applications. (California: Morgan Kaufmann Publishers, Inc., 1992).

Parsaye, K. , "New Realms of Analysis: Surveying Decision Support," Database Programming and Design, 9(4):27-33, April, 1996.