

Technical Comparison of Oracle9i
Database vs. SQL Server 2000:
Focus on Manageability

An Oracle White Paper
November 2002

Technical Comparison of Oracle9i Database vs. SQL Server 2000: Focus on Manageability

Executive Overview	3
Introduction	4
Management Tools	5
Schema Management	6
Table and Index Online Reorganization	6
Partitioning	7
Space Management	9
Row Size Limit	9
Handling Out-of-Space Situations	9
Performance Management	10
Identifying and Tuning SQL Statements	10
Execution Plan Stability	11
Managing Locking Conflicts	11
Managing CPU Resources	13
Backup & Recovery	13
Automatic Log Archiving	13
Backup Throttling	14
Self-Contained Backups	14
Recovery from Human Errors	14
Block Media Recovery	15
Installation and Configuration	16
Instance Creation	16
Database Duplication	16
Cross-Platform Portability	17
Scalability	17
Conclusion	19

Technical Comparison of Oracle9i Database vs. SQL Server 2000: Focus on Manageability

EXECUTIVE OVERVIEW

Today acquisition costs represent only a small portion of the total cost of ownership of any system or software. The ongoing costs of managing and maintaining a system far outweigh the initial acquisition costs. Many factors contribute to this cost. Software that is not easy to manage requires highly trained administrators, which makes maintenance operations inherently expensive. Direct costs to businesses resulting from undue complexity translate into the increased expense of hiring, training, and retaining experienced personnel. There are also hidden costs due to poor availability and performance since systems that are difficult to diagnose or tune tend to have poor availability. Furthermore, complex management procedures increase the likelihood of administrator error, further damaging availability. All these factors make it essential that database systems not only excel in performance, availability, scalability, etc., but that they also be easy to use and manage. Manageability, therefore, is a key consideration today when evaluating any enterprise software or application.

The superiority of the Oracle Database in the areas of performance, scalability, and availability to its competitors, including SQL Server 2000, has never been in question. The Oracle9i Database extends this lead to manageability as well, thanks to a number of significant enhancements introduced in this release. For all major DBA management functions, ranging from space and schema management to performance tuning to backup & recovery, the Oracle9i Database offers solutions that are easier to use, more effective and scalable than SQL Server 2000. While SQL Server 2000 may seem easier to use at a glance, it lacks some basic capabilities that make its management extremely challenging. SQL Server 2000 administrators have practically no control over how a database or an application built on top of it performs. To make matters worse, they are burdened with unnecessary management tasks such as managing locking conflicts. SQL Server 2000 does not provide any non-intrusive way of recovering from human errors either, which forces administrators to perform complicated database recovery operations and exposes valuable enterprise data to grave risk. The list of such basic capabilities missing from SQL Server 2000 goes on and on. The Oracle9i Database, on the other hand, provides the most comprehensive manageability infrastructure ensuring that the DBAs can perform all their important functions with minimal effort. The Oracle9i Database, therefore, is not only the most reliable, scalable and high performing

data management solution available today, but is also the easiest to use and manage.

INTRODUCTION

A well-managed IT infrastructure is critical to the success of any modern business. As IT vendors deliver increasingly sophisticated solutions to meet the exacting demands of the “networked society”, the task of systems management has never been more complex. Hiring highly skilled administrative staff to manage such complicated environments is an expensive proposition. This coupled with frequent shortage of experienced administrative personnel often results in spiraling management costs.

In order to meet these challenges, Oracle has made the enhanced manageability of its products one of its primary goals for Oracle9i and beyond. The Oracle9i Database automates a number of routine administrative tasks, reduces the complexity of administration and, includes a number of self-tuning features to deliver optimal out-of-the-box performance.

The manageability enhancements introduced in Oracle9i are not just a bunch of disparate features; instead they are the results of a structured methodology adopted by Oracle to deliver a comprehensive and holistic management solution. Oracle’s methodology is simple. *Eliminate* complexity, whenever possible. What you cannot eliminate, *automate*. And what cannot be automated, *simplify*. As an example, with automatic undo management, an administrator no longer needs to be aware of how the undo or rollback data is managed since the database performs this transparently. The administrative functions that could not be completely eliminated in the current release of Oracle9i have been *automated* in order to reduce the degree of manual intervention required to keep the database operational. As a result, Oracle9i relieves administrators of complex tasks such as tuning SQL execution memory and managing standby databases. But even with the highest degree of automation, there will always be certain tasks that must be performed manually such as adding or removing system resources. Such tasks have been significantly *simplified* with the objective to minimize the time an administrator spends in performing them. So, for adding additional disk space to the database, administrators just need to specify how much space should be added while the Oracle databases automatically creates and manages the underlying operating system files.

All such automation and simplification is not possible unless the product has some built-in intelligence to learn about the workload and database usage pattern and, then use this knowledge to either automatically perform a given activity or help DBAs perform those tasks in a simple manner. The numerous advisories such as Buffer Cache, PGA, etc., introduced in Oracle9i are examples of the features that make the Oracle9i Database more intelligent than any other competing product. Finally, the comprehensive enterprise management solution offered by Oracle Enterprise Manager (OEM) framework makes it possible to manage all components of the IT Stack without having to buy and integrate a number of third party software products.

The Oracle9i Database, therefore, is the only product that can satisfy the stringent scalability, availability and manageability requirements of modern eBusinesses. Whereas, Oracle is generally accepted as the clear front-runner in the areas of scalability, availability, and performance, SQL Server is often given high-marks for its ease-of-use. However, this perception belies the actual reality. SQL Server's inferiority in scalability, availability, and performance relative to Oracle means that DBAs have to resort to non-optimal and inefficient methods to overcome its shortcomings, which enhances the complexity of their applications and makes it extremely difficult and more expensive to manage. This fact was substantiated in a recent study conducted by an independent market research firm, INPUT, which found that the cost of running packaged applications such as SAP, Siebel, and PeopleSoft, was significantly higher in a SQL Server environment compared to Oracle. The study established that the annual per user cost of running packaged applications on SQL Server was twice that of Oracle, and that Oracle DBAs had a much higher productivity than their SQL Server counterparts, since the number of users they supported was on average four times higher than SQL Server administrators¹. Moreover, Oracle has made tremendous headway in making its system easier to manage over the last few years without compromising any of its main strengths. In this paper we will take an in depth look at the various capabilities of the two systems and evaluate them in the area of manageability.

MANAGEMENT TOOLS

Both Oracle and SQL Server offer GUI tools for the management and maintenance of their systems. Coincidentally, both the tools are called Enterprise Manager. Oracle Enterprise Manager (OEM) is Oracle's single, integrated solution for administering and monitoring global eBusiness enterprises. It offers a single point of control to manage all the components of an eBusiness infrastructure from the application through to the middle-tier and database down to OS and the network. In essence, OEM lets administrators manage the end-user experience, not just the individual system components.

Looking from a database-centric point of view, OEM significantly simplifies the day-to-day database administration by combining simplicity with a rich feature set to offer a complete solution for managing any size environment regardless of platform. It includes a comprehensive set of events that automatically monitor the entire Oracle environment. Administrators do not need to decide how to monitor their systems or what to monitor. In addition, it automates crucial management tasks and extends the reach of administrator beyond his/her desktop by providing capabilities such as:

- Production-ready monitoring and proactive alerts to anticipate problems and fix them automatically using "fix-it" jobs.
- Built-in advice driven diagnostics that allow administrators to diagnose and resolve performance problems without extensive research or training.

¹ Detailed study findings can be found at http://www.oracle.com/ip/deploy/database/oracle9i/db_sqllex.html

- Comprehensive, automated reports that can be published to the web to measure and manage service level agreements.
- Lightweight browser interface for access to management tools from anywhere.

"Enterprise Manager provides a centralized management framework for our distributed open systems. Oracle9i and the applications Oracle has developed to meet BT's requirements will put us a long way toward our goal of having a single management tool for all DBAs." — British Telecom.

A large number of Oracle customers such as British Telecom, Boeing, Telstra, and Shell have standardized on OEM to manage their production systems. Their experience is live testimony of how effective Oracle management tools are in boosting administrator productivity and increasing the number of databases each DBA can manage.

SQL Server 2000's Enterprise Manager tool lacks the depth of management capabilities provided by OEM. It does not allow for end-to-end management and diagnostics of the entire system from a single Console. Metrics relating to the OS, middle-tier application servers, etc., cannot be monitored by SQL Server's management tool. It does not have any service level reporting capability either. Oracle Enterprise Manager includes over a hundred pre-canned service level reports dealing with application performance and system availability. These reports make it easy to see where potential performance problems may be developing. They also provide a record of the system's service over time. Furthermore, SQL Server has no answer to Oracle Enterprise Manager's built-in advice driven diagnostics for problem resolution. OEM highlights problem areas with visual alarms so potential performance issues are easily seen. Powerful diagnostics and tuning tools then enable diagnosis of the problem, whether it is in the application, middle-tier application server, database, host or network. For example, an Oracle DBA can instantly see what the "Top" resource consuming sessions or SQL statements are in the database, and get instant advice on indexes to create to increase query performance. To do the same in SQL Server, a DBA would have to turn on tracing on the database (this has obvious performance implications), rerun the workload (which is generally a most inconvenient event), and manually parse through the trace files to determine most resource consuming SQL (a very time-consuming and cumbersome exercise). Only then can the DBA get advice on what indexes to create to resolve the problem. Thus, SQL Server, unlike Oracle, requires its administrators to invest considerable time in training and in customizing its tool to support essential administrative functions.

SCHEMA MANAGEMENT

Table and Index Online Reorganization

Oracle9i Database enables database administrators to perform almost all schema maintenance operations online, i.e., data is fully available for queries, updates, and deletes. Schema evolution allows table definitions to be modified while the data table is in service. Tables can be relocated, defragmented, reorganized or have their storage parameters changed. Indexes can be added, rebuilt or defragmented online.

“In previous years, due in part to the extremely large volume of data maintained at Amazon, we could spend hours with our systems offline while we performed indexing operations. Online indexing operations have eliminated this downtime, and helped us optimize performance and availability throughout the site.”

Matt Swann
Director, DB Services
Amazon.com

SQL Server 2000 does not support the extensive set of online operations that Oracle9i Database does. It does not have online schema evolution, table reorganizations, table redefinitions and secondary index creations on index-organized tables. Since many maintenance operations are done quite frequently and can take hours to complete, SQL Server 2000 applications can suffer significant data unavailability. As a result, SQL Server DBAs have to skillfully devise elaborate plans on when to perform these operations so as to minimize the impact of system unavailability on users. This significantly adds to the management workload of the DBA, and is something that Oracle DBAs do not have to consider at all in their day-to-day operations.

Partitioning

Partitioning allows large database structures (tables, indexes, etc.) to be decomposed into smaller and more manageable pieces and, at the same time, improves query performance and resource utilization. The Oracle9i Database offers several partitioning options designed to fulfill different requirements²:

- Range partitioning uses ranges of column values to map rows to partitions. Partitioning by range is particularly well suited for historical databases. Range partitioning is also the ideal partitioning method to support 'rolling window' operations in a data warehouse.
- Hash partitioning uses a hash function on the partitioning columns to place data into partitions. Hash partitioning is an effective means of evenly distributing data for optimal performance.
- List partitioning allows users to have explicit control over how rows map to partitions. This is done by specifying a list of discrete values for the partitioning column in the description for each partition.

In addition, Oracle supports range-hash and range-list composite partitioning.

Oracle also provides three types of partitioned indexes:

- A local index is an index on a partitioned table that is partitioned using the exact same partition strategy as the underlying partitioned table. Each partition of a local index corresponds to one and only one partition of the underlying table.
- A global partitioned index is an index on a partitioned or non-partitioned table that is partitioned using a different partitioning-key from the table.
- A global non-partitioned index is essentially identical to an index on a non-partitioned table. The index structure is not partitioned.

Oracle allows all possible combinations of partitioned and non-partitioned indexes and tables: a partitioned table can have partitioned and non-partitioned indexes, and a non-partitioned table can have partitioned and non-partitioned indexes.

SQL Server 2000 does not support table or index partitioning but has what are called partitioned views. A partitioned view joins horizontally partitioned data

² For more information about Oracle9i's partitioning options, see Oracle9i Partitioning, Hermann Baer, Technical White paper, Oracle Open World 2001, Berlin.

from a set of member tables across one or more servers, making the data appear as if from one table. The data is partitioned between the member tables using range partitioning concepts. Each member table has a CHECK constraint that defines the range on the partitioning column, ensuring that the right set of data goes to the right member table. A view is then created that uses UNION ALL to join all the members tables, thus creating a single partitioned view.

Partitioned views, which the Oracle Database has supported since version 7.3, solve a small part of the challenges involved in managing large data sets. While partitioned views provide a simple way for users to select data from a set of underlying tables, each of these tables must be managed separately. For example, data must be loaded in each table individually, indexes must be created on them separately, etc. Also, a DBA has to manually create and maintain triggers on each underlying table in the view. Therefore, from a DBA perspective, partitioned views do very little to simplify the management of large volumes of data.

The table below summarizes some of the other differences between Oracle9i Database and SQL Server 2000 with regard to the partitioning options:

Table 1: Partitioning options.

Feature	Oracle	SQL Server
Range partitioning	Yes	Yes (partition views)
List partitioning	Yes	–
Hash partitioning	Yes	–
Composite partitioning	Yes	–
Local index	Yes	Yes
Global partitioned index	Yes	–
Global non-partitioned index	Yes	–

SQL Server 2000's partitioned views do not support global indexes. This means that queries that do not specify a search condition on the "partition" column will have to search all the tables in the partitioned view, since there are no global indexes to let the system know which "partition" contains the desired data. This is a major problem in OLTP environments where global indexes are a must for efficient data access. In partitioned view configurations, SQL Server application designers have no flexibility in defining their indexing strategies. This lack of support considerably restricts the ability of SQL Server 2000 to be used in real-world OLTP applications.

Another advantage that Oracle has over SQL Server is that it offers much easier 'rolling window' support. A 'rolling window' allows the data for a defined period, e.g., a week or a month, to be kept online by continuously replacing the oldest data with the most recent one. In SQL Server, all the steps associated with 'rolling window' support, such as creating new member tables, creating constraints, etc., have to be done manually. Oracle's Change Manager, on the other hand, provides a user-friendly, intuitive tool that provides complete 'rolling window' support requiring no manual steps at all.

All these limitations in SQL Server make the task of managing large volume of data quite complicated. Even Microsoft has finally acknowledged SQL Server's weakness the area of partition management. Peter Spiro, leader of Microsoft's SQL Server engine team and an 8-year veteran of the database group, in the October issue of the SQL Server Magazine agreed that in SQL Server, *"partitioned views are difficult for customers to use."* He goes on to say that *"distributed partitioned views are a stopgap measure for people who are using multiple machines."*³ Thus, Oracle's partition management solution is far superior to SQL Server, as it offers a comprehensive set of features that not only enhance manageability but also improve performance.

SPACE MANAGEMENT

Row Size Limit

SQL Server imposes a limit of 8060 bytes on the size of a row⁴. This is because in SQL Server, just like in IBM DB2, a row cannot span multiple pages. Since the maximum size of a SQL Server page is 8KB, a row — after discounting for overhead — can be no larger than 8060 bytes. In order to overcome this limitation, a DBA would have to vertically divide the table into multiple tables, thereby compromising the optimality of the application design. Thus, a sub-optimal design was forced upon the DBA because of yet another artificial limitation imposed by SQL Server. Oracle allows a row to span across multiple pages (or blocks in Oracle's parlance) thereby completely avoiding all the complications outlined above.

Handling Out-of-Space Situations

Operations such as data loads or batch updates can encounter errors when they run out of space. Sometimes these errors occur just when the operation is about to finish. Oracle's Resumable Space Allocation feature enables it to handle such errors in a very graceful manner. Whenever an operation encounters an out-of-space situation, it is held in a "suspended" state while the administrator is notified of the problem and given a chance to fix it. The "suspended" operation automatically resumes as soon as the error condition is corrected. In case of a transient problem, such as a query running out of temporary space, no administrator intervention may be required since Oracle will resume the operation automatically as soon as the transient problem disappears. Virtually any kind of operation, be it a PL/SQL or Java stored procedure, a DML or DDL statement, or an export/import or loader session, can all be run in the "resumable" mode.

This capability is unique to Oracle with no parallel in SQL Server 2000. It saves Oracle DBAs enormous time that their SQL Server counterparts spend in monitoring and re-executing failed long running operations. In SQL Server, if an operation runs into an out-of-space situation, the entire operation may have

³ *Reving the SQL Server Engine*, SQL Server Magazine, October 2002, ID #26435.

⁴ <http://www.devx.com/codemag/articles/2002/March/sqlconfig/sqlconfig-4.asp>

to be repeated after the space issue has been addressed. This not only results in wastage of time but it can also hamper normal database performance, for instance, if the out-of-space situation forces the DBA to re-run the batch job during normal workload hours. Thus, Oracle's Resumable space allocation feature avoids all the pitfalls that a SQL Server user would encounter by offering the user a graceful way to manage out-of-space situations.

PERFORMANCE MANAGEMENT

Identifying and Tuning SQL Statements

Deficiencies in application design are the most common cause of performance problems. DBAs, therefore, spend a significant amount of their time identifying and tuning resource intensive SQL statements. Here again, Oracle makes life simple for administrators by automatically compiling the resource consumption statistics for active SQL operations. All that the administrator needs do to identify problem SQL statements is to either launch the SQL Analyze GUI tool, a part of the EM product suite, or execute a simple query against the database⁵. To do the same in SQL Server, one must go through a series of time-consuming steps of first setting up a trace event through the SQL Profiler, enabling the relevant events and then re-running the workload so that run-time statistics are captured in the trace files. Once trace files have been generated, they then have to be analyzed manually. This is a very significant manageability issue for SQL Server administrators for a couple of reasons. First of all, monitoring resource consuming SQL statements is a common database administrative task and hence, spending anything more than a few minutes to get this information is completely unacceptable. Secondly, for a busy system, the generated trace file can easily grow to a very large size quite rapidly, making it extremely difficult to extract meaningful information from it. SQL Server, therefore, lags far behind Oracle in this very basic but most important DBA task.

Once the high impact SQL statements have been identified, they have to be tuned. Tuning SQL involves two main steps: 1) rewriting SQL if it is poorly structured, and 2) providing new access methods that optimize the execution path. Oracle has powerful tools that aid in both these areas. Its SQL Analyze tool has a SQL Tuning Wizard that can automatically check the SQL statements for basic violations such as, inadvertent disabling of an index by the use of certain operations, or using unnecessary filter and sort operations. If any violation is detected, it can correct the problem by rewriting the SQL and also provide an estimate of the projected performance improvement based on the reduction in optimizer cost of the rewritten SQL. The second step in tuning SQL has to do with optimizing access methods. This involves creating any new indexes or materialized views that will speed up the query. Oracle offers the Virtual Index Wizard and Summary Advisor for identifying and recommending the optimal indexes and materialized views for a given workload. SQL Server 2000, on the other hand, does less than half the job. It offers no help with

⁵ The dynamic performance view V\$SQLAREA provides the resource consumption statistics for SQL statements in memory.

materialized views or with rewriting poorly structured SQL queries. The only area where it does assist is with the creation of indexes, thus leaving the more challenging and time consuming of tasks to the DBAs and application developers.

Execution Plan Stability

DBAs crave predictability. One of the areas where DBAs desire it the most is in query performance. The performance of a query depends on the optimality of its execution plan. The more efficient an execution plan, the better the query performance. However, cost based optimization — an optimization model used by both Oracle and SQL Server — among other factors also relies on the database environment in generating an execution plan of a query. As a result, changes in database environment such as optimizer statistics of the underlying objects, memory parameters, etc., can suddenly change the execution plan of a query, mostly improving it but sometimes making it drastically worse.

This is a serious problem in SQL Server, not only because the database has no way of providing execution plan stability, but also because the database server, by default, continually regenerates optimizer statistics, hence opening up the possibility of continually changing execution plans. The only thing a DBA can do in this situation is to turn off auto-creation and update of optimizer statistics so that execution plans do not change. But this opens up a new set of problems because now the table and index statistics can become stale and the optimizer will be forced to generate execution plans based on stale data, resulting in poor optimization. In reality, SQL Server has no real solution to the issue of execution plan stability, and in fact its default behavior of continually updating optimizer statistics actually exacerbates the problem.

Unlike SQL Server, Oracle provides DBAs the ability to freeze execution plans of queries by creating *stored outlines*. This allows the DBA to ensure plan stability for those queries where they cannot risk changes in execution plans, as well as during critical periods, e.g., quarter-close. This gives the DBA the predictability in query behavior that they crave without compromising good administration practices like regularly refreshing optimizer statistics. In this way, Oracle presents an elegant solution to a serious performance management concern of DBAs, whereas SQL Server offers no practical way of dealing with the issue.

Managing Locking Conflicts

One of the biggest strengths of Oracle from the performance management point of view comes from its fundamental architecture. Rejecting the old model based on lock-based concurrency control — used by SQL Server — Oracle chose to implement its own multi-version read consistency model ensuring that readers and writers never block each other. In Oracle, whenever a change is made by a transaction, the original data values are copied in database structures called undo segments. Consequently, unlike SQL Server, which uses locks to prevent records from being changed by others while being read, or to prevent queries from reading uncommitted changes, Oracle uses the undo information stored in the database to construct a read-consistent version of data. This

ground-breaking technology allows the Oracle database to service queries without requiring any read locks.

SQL Server 2000 does not provide multi-version read consistency. Instead it requires applications to either use shared locks for read operations with various levels of isolation, or to accept dirty reads. Shared locks prevent data that is read from being changed by concurrent transactions. Clearly, this implementation restricts the ability of the system to properly service concurrent requests in environments involving a mix of reads and writes, as explained in Microsoft's documentation: "*SQL Server, in contrast, uses shared locks to ensure that data readers only see committed data. . . . A reader waits for a writer to commit the changes before reading a record. A reader holding shared locks also blocks a writer trying to update the same data.*" As a consequence, releasing locks quickly for applications that support high numbers of users is far more important in SQL Server than in Oracle.⁶

The only alternative developers have to this problem is to build separate workload environments, where intensive read activities, such as reporting, cannot interfere with on-line transactional applications. Regardless of which approach is used, SQL Server 2000 developers usually have to find some compromise in their application design in order to get acceptable data concurrency and accuracy.

In Oracle, writers and readers never block each other. Oracle's powerful multi-version read consistency allows mixed workload environments to function properly without incurring any performance penalty for the users.

"Locking at a smaller granularity, such as rows, increases concurrency, but has a higher overhead because more locks must be held if many rows are locked. Locking at a larger granularity, such as tables, are expensive in terms of concurrency because locking an entire table restricts access to any part of the table by other transactions, but has a lower overhead because fewer locks are being maintained".

Microsoft SQL Server documentation.

Another problem that occurs as a result of SQL Server's lack of multi-version read consistency and its locking model is lock escalation. SQL Server puts a limit on the maximum numbers of locks that can be supported in a database. This coupled with the fact that SQL Server uses many more locks (due to read locks), means that as the transaction volume increases, databases can easily reach a threshold value at which row level locks will escalate to page or table level locks to conserve memory. This in turn means that fewer users can access the data at the same time – users will have to wait, and the chances of getting false deadlocks is also greatly increased. According to SQL Server documentation, "*Locking at a smaller granularity, such as rows, increases concurrency, but has a higher overhead because more locks must be held if many rows are locked. Locking at a larger granularity, such as tables, are expensive in terms of concurrency because locking an entire table restricts access to any part of the table by other transactions, but has a lower overhead because fewer locks are being maintained.*"⁷

Oracle imposes no limit on locks, its locks never escalate and, as a consequence, Oracle users never experience false deadlock situations due to lock escalation. Oracle DBAs, therefore, need not even think about some of the most time consuming tasks that their SQL Server counterparts have to perform on a daily basis i.e., monitoring lock escalations and resolving deadlocks.

⁶*Migrating Oracle Databases to SQL Server 2000*, SQL Server Resource kit, p. 57.

⁷ Microsoft SQL Server documentation: Understanding Locking in SQL Server http://msdn.microsoft.com/library/default.asp?url=/library/en-us/acdata/ac_8_con_7a_7xde.asp

Managing CPU Resources

The ability to easily and accurately perform system and resource management is critical to maintaining application and database performance, scalability and availability. The Oracle Database Resource Manager enables administrators to align the distribution of system resources with enterprise goals by allowing allocation of CPU resources among database users and applications according to business priorities. Its ability to automatically limit the resources consumed by batch jobs helps in ensuring that such operation do not adversely impact online users in a mixed workload environment. Furthermore, Database Resource Manager also provides the ability to limit the number of concurrent long operations and prevent execution of highly resource intensive queries during certain times of the day. The Database Resource Manager, therefore, makes it extremely easy to deliver predictable service level with minimal human intervention and facilitates almost unlimited system scalability without compromising performance⁸.

SQL Server has no answer to this feature. SQL Server DBAs have no control over how CPU resources are allocated which means that important processes can be starved for CPU while those not so important can unduly monopolize it. This is because the database server has no way of distinguishing between higher and lower priority processes. To overcome this shortcoming, SQL Server DBAs have to constantly monitor resources consumed by various processes and when necessary take corrective action manually to ensure efficient use of CPU resources. An obvious problem with this approach is that in most cases a DBA may not have a good option for corrective action, and secondly, manually controlling CPU resources is a very time-consuming and difficult task with limited chances of success.

BACKUP & RECOVERY

Automatic Log Archiving

In order to recover from media failures, both Oracle and SQL Server require the redo logs (or transaction logs in SQL Server terminology) to be backed up. Oracle automatically backs up the redo log by a process called *archiving*, which is enabled by simply putting the database in archive log mode. In contrast, the transaction logs in SQL Server, once they fill up, have to be backed up by the DBA manually. To avoid this manual task, DBAs will often set up a batch job that runs periodically to back up the transaction log. This is not really an acceptable solution because batch jobs cannot adapt to changes in workload. The batch job frequency is based on expected workloads, and if for some reason the database undergoes a higher than expected transaction volume, then the transaction log can easily grow and use up all the free space, resulting in the database coming to a halt. On the other end of the spectrum, if a database has

⁸ For more details please refer to paper titled *Oracle9i Database Resource Manager* at http://otn.oracle.com/products/manageability/database/pdf/9i_Resource_Mgr_TWP.pdf.

very little or no transaction volume, the batch job will still back up the transaction log even though it might be empty. In short, SQL Server's lack of automatic archiving necessitates that DBAs carefully monitor the transaction logs and must back them up as required. An Oracle DBA is completely free of worries in this area.

Backup Throttling

Extending the discussion on “online” operations, a truly online backup should not only ensure that the database stays up during the backups but also limit its performance impact so that users can continue to use the database. This is extremely important in today's round-the-clock economy since users do not care to differentiate between an unavailable and a poorly performing system. Oracle Recovery Manager (RMAN), therefore, allows administrators to throttle the backup read rate in order to contain the performance impact within an acceptable limit. There is no way to achieve this in SQL Server. As a result, when a SQL Server database is being backed up, users may experience performance degradation, since there is nothing a DBA can do about it other than to optimize the system for maximum I/O bandwidth so as to speed up the backup process⁹. This limits the flexibility of a SQL Server DBA significantly as far as scheduling a backup is concerned since they have to be extra careful not to disrupt normal system functioning.

Self-Contained Backups

In Oracle, the backups are complete and fully self-contained. A DBA can recover an Oracle database from any situation as long as there is a good backup of the database. This is not the case in SQL Server. If the SQL Server system database, *msdb*, is lost, a DBA cannot recover the system without undergoing a frantic search for the original install CD even though he/she may have performed regular backups. This is because SQL Server backups are not self-contained. A DBA must first manually recreate the system database, *msdb*, from the original install CD using the command line utility, *rebuilddm.exe*¹⁰. The recovery of application databases can be started only after the master database has been recovered. This makes the complete recovery of a SQL Server instance quite complex and highly intuitive unlike Oracle where the same operation can be performed simply using the recovery wizard. Even in this most critical DBA function such as recovering a down database instance, SQL Server poses a serious manageability impediment to its DBAs.

Recovery from Human Errors

Numerous availability studies have highlighted human error as one of the most prominent causes of application outage. While it is possible to develop safeguards against hardware and software failures, it is nearly impossible to insulate a system from human mistakes such as accidentally deleting critical data. Once again, only Oracle provides easy and completely non-intrusive options to recover from such failures. Oracle's Flashback Query enables

⁹ Whalen, Edward, et al., Microsoft SQL Server 2000 Performance Tuning, p. 281.

¹⁰ SQL Server Books Online: How to rebuild the master database.

administrators and users alike to view data at a point-in-time in the past and use it to reconstruct the lost data. This is a very significant benefit. To illustrate this further, let us take the example of a banking application where a set of bank accounts was accidentally deleted from the ACCOUNTS table at 11 AM. The inconsistencies caused by this mistake were later discovered at 2 PM. Using Flashback Query, the database administrator could execute a simple SQL command to recover from such an error. The following query retrieves the deleted accounts by comparing the current data in the ACCOUNTS table as against what it was just prior to 11 AM and, inserts them back into the table.

```
INSERT INTO ACCOUNTS
(SELECT * FROM ACCOUNTS AS OF TIMESTAMP
TO_TIMESTAMP('13-MAR-02 10:59:58','DD-MON-YY HH24:MI:SS')
MINUS
SELECT * FROM ACCOUNTS);
```

If the bank were using SQL Server, they would have to rollback the database just prior to 11 AM, when the error occurred, in order to “recover” the deleted data thereby losing all subsequently committed transactions, an unacceptable solution for most circumstances. Since rolling back the database requires restoring a backup copy and rolling it forward to the desired point in time, it will also lead to the system being unavailable for the duration of recovery. Clearly, this is not the solution the bank is looking for since it forces the bank to compromise system availability and accept data loss. Flashback Query, on the other hand, made it possible for the bank to correct this mistake without sacrificing any data and with absolutely no effect on the normal operations. This enabled the bank to avoid any revenue loss due to “lost” data and application outage.

Oracle provides another extremely powerful tool called “Log Miner” to find and correct unwanted changes. Using Log Miner, administrators can scan the redo log files to precisely determine when and how the incorrect change was made and, who was responsible for this change in case it was done with malicious intent. The tool can display the change history for a given set of data, if desired, and provides the SQL statements needed to “undo” the erroneous transaction. Once again, this is much easier than performing a point in time recovery and is completely transparent to the normal functioning of the database. SQL Server does not have any log-mining tool. As a result, any time there is logical corruption, SQL Server DBAs are left with the dreadful option of recovering the database to a specific point in time in the past and the loss of all transactions from that point onwards.

Block Media Recovery

Oracle provides the option to perform recovery to the granularity level of data blocks. Using the block media recovery feature of Oracle, if only a single block is damaged then only that block needs to be recovered, while the rest of the file and the table containing the block remains online and accessible. This not only speeds up the recovery process but also increasing data availability. Here too, SQL Server is not able to compete with Oracle. It cannot recover data in single block units, thus requiring the entire file to be taken offline, restored, and recovered.

INSTALLATION AND CONFIGURATION

Instance Creation

Instance creation is among the most basic and important DBA functions. Oracle has a simple, easy-to-use GUI tool, Database Configuration Assistant (DBCA), for creating database instances. DBCA performs all the necessary steps for the user with minimal input and eliminates the need to plan in detail the parameters and structure of the database. Furthermore, DBCA also makes recommendations about certain database settings based on user input and thus helps database administrators make correct decisions for optimal database configuration depending on whether it will be used for Data Warehousing, OLTP or mixed workload applications.

In SQL Server, on the other hand, every instance creation requires a complete new installation with all the binaries and scripts. This is in contrast to Oracle where multiple instances on the same machine share binaries and other relevant files. Consequently, any time multiple instances are needed on a single machine, a SQL Server DBA has to contend with considerable amount of wasted space, as each instance needs its own private binaries and scripts. Furthermore, during installation and instance creation SQL Server also offers no help in setting database parameters, thus adding to the workload of its DBAs.

Database Duplication

Creating a standard definition for enterprise wide databases is a fairly common practice. Having all databases conform to a standard greatly simplifies their management by allowing the use of uniform administrative and monitoring procedures. DBA's also often want to duplicate databases for the purpose of setting up development and test environments. Oracle database templates, which allow for storing database definition in XML format, make both these tasks extremely simple. The template definition includes all characteristics of the database, such as initialization parameters, redo log settings, etc., and can be used to create identical databases on local or remote machines. There are two kinds of templates, those that contain only the structure of a database, and those that contain both the structure and the data. This functionality allows administrators to either create a new structurally identical database or clone a database along with its data. A template can be created by reverse engineering an existing database. This ability is extremely useful since it saves administrators a significant amount of time and effort in creating and testing scripts for duplicating a database.

SQL Server has no comparable feature to this. The only way to create a new SQL Server database is to use the Copy Database Wizard. There are many drawbacks to this method. First, in this method the database structure alone cannot be duplicated; instead both the data and the database have to be copied. Second, the database being copied must not have any active sessions on it, that

is, the database must be offline to users¹¹. If the SQL Server DBA wants to duplicate just the structure of the database, which is the more common case, the database and all its objects have to be recreated manually using scripts or the GUI tool. This is an extremely cumbersome and error-prone method and, is certainly not comparable to the functionality provided by Oracle.

CROSS-PLATFORM PORTABILITY

One of the major and unique strengths of Oracle's architecture is its portability to different platforms. Oracle is available on practically all platforms and it is by far the most portable database than any of its competitors. This means that an Oracle database on Unix has the same code base as the one running on a Windows OS, ensuring that the management of the Oracle database on one platform is no different than on another. This makes it possible for an Oracle DBA to manage databases on any and all platforms without additional training. Another benefit of extensive portability is that an application can be designed independent of the platform with the sole objective of addressing the business requirements in the best possible manner.

The same cannot be said about SQL Server as it runs only on the Windows platform. As a result, any future growth of a business is limited to the Windows platform. Outgrowing Windows means a huge investment of time and money to upgrade all the hardware and software systems to Unix, migrate the data to a more scalable database, rewrite all the applications to run against the new database, and hire or retrain DBAs. With Oracle, you never need to worry about outgrowing your hardware, since Oracle data and applications are fully compatible and portable across all major hardware and operating systems platforms.

SCALABILITY

Another of Oracle's great assets is its scalability. Oracle databases can scale seamlessly from hundreds to thousands of users. In addition, Oracle9i Real Applications Cluster (RAC) offers the most transparent cluster database solution from a deployment and management perspective. Nodes can be added to a RAC database on the fly with no impact on existing nodes. Also, since all nodes in a RAC environment access a single database, addition or removal of nodes does not generate any additional tasks for a DBA. The most unique aspect of RAC, however, is that unlike other database products (including SQL Server) where an application has to be rewritten for a clustered environment, moving from a single node to a RAC environment requires absolutely no application change thanks to its patented Cache Fusion technology. This technology has been enthusiastically welcomed by customers and ISVs alike, as evidenced by its fast adoption. Already, there are more than 50 ISVs that are using RAC as their development platform. For more details on RAC and Cache Fusion, refer to the white paper titled "Oracle9i Real Application Clusters:

¹¹ SQL Server 2000 Books Online, Administering SQL Server.

Cache Fusion Delivers Scalability”, which can be found at http://otn.oracle.com/products/oracle9i/pdf/cache_fusion_rel2.pdf.

Microsoft’s recommended implementation of a scalable model for SQL Server is either in a distributed database environment, or in a federated database architecture, where many independent databases are connected together with no common data dictionary while a single application accesses data across all these databases. Imagine a simple task such as adding a node in SQL Server’s federated architecture environment. A DBA or system administrator will have to do all of the following:

- Add hardware
- Configure new instance (set instance-specific parameters, etc.)
- Create new database
- Disconnect all users
- Unload data from existing tables
- Redefine partitioned tables and indexes
- Redefine triggers on partitioned or replicated tables
- Redefine distributed partitioned views
- Reload the data to spread it across a larger number of partitions
- Reconnect all users.

This represents a highly complex and elaborate set of management tasks that involve unloading and reloading of data, redefining of views, modification of triggers, etc. Furthermore, when these tasks are being performed all the databases have to be unavailable for the entire duration of the operation. Compare this to Oracle9i Real Application Clusters. The administrator needs to do the following two tasks only when adding a node:

- Add hardware
- Configure new instance (set instance-specific parameters, etc.)

The DBA did not have to unload and reload data, change any schema definitions, nor was the system taken offline at all. Thus, when compared to Oracle, SQL Server’s scalability solution is a manageability ordeal even for basic tasks.

“Microsoft sells ease of use and hesitates to point out that a SQL Server environment is often just as complex — and expensive — to manage as its UNIX counterparts”.

Brian Moran
Editor, SQL Server Magazine

Another point to note about SQL Server is that the complexity of management goes up dramatically when there are tens of databases to manage, administer, maintain, backup and upgrade. As an example, to achieve a high TPC-C benchmark Microsoft had to use 32 different databases. Thus, Microsoft’s approach is to distribute rather than contain complexity. Managing so many databases is quite an unappealing scenario for DBAs when compared to implementing and administering a single, more scalable database like Oracle. Brian Moran, SQL Server Magazine editor says it all: “Microsoft sells ease of use and hesitates to point out that a SQL Server environment is often just as complex — and expensive — to manage as its UNIX counterparts.”¹²

Oracle’s scalability advantage is not restricted to the multi-node RAC model only. Being the most portable database in the market, it runs on all major OS platforms including large SMP boxes. SQL Server, on the other hand, is

¹² SQL Server Magazine UPDATE News Editor, Jan. 2001.

restricted to the Windows operating system. This means that SQL Server's scalability within a box is restricted to the smaller Windows machines, while Oracle has no such limitation as it runs on all major platforms. It can easily scale on large 64-cpu SMP boxes without any problem. From a manageability perspective this is a great advantage because as the applications grow over time, an Oracle DBA can address the growth by just adding more CPUs to the machine, whereas in the case of SQL Server an application can quickly outgrow the smaller Windows machines. This leaves the DBA with no option but to add more machines and then hope that the application can be redesigned for a federated architecture. Even when possible, this is an extremely painful and time-consuming task.

To overcome the lack of scalability of SQL Server 2000, a DBA has to administer an environment that is complex and virtually unmanageable. Simple tasks like adding a new node can involve very elaborate redistribution of application data requiring deep understanding of the application logic and the database layout. These are just some of the manageability issues that a SQL Server DBA has to contend with on a regular basis, whereas Oracle's scalable architecture makes all these non-issues for its DBA.

CONCLUSION

Oracle9i Database is not only rich in functionality but, thanks to its bold initiative to enhance manageability, it is also easier to manage than SQL Server 2000. With management costs of software quickly outpacing acquisitions costs, ease-of-use or manageability has become a most crucial factor affecting purchase decisions. Oracle is the only solution in the market that offers the best of both worlds. SQL Server 2000 lacks many of the basic management functionality critical for the effective management of a database environment. Consequently, SQL Server DBAs are often left with no choice but to undertake cumbersome workarounds that are unintuitive, complex, and frequently a management nightmare. Oracle's comprehensive and structured approach to manageability will only further its lead over its competitors in the future releases.



Technical Comparison of Oracle9i Database vs. SQL Server 2000: Focus on Manageability
November 2002

Authors: Mughees A. Minhas, Sushil Kumar

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
www.oracle.com

Oracle is a registered trademark of Oracle Corporation. Various product and service names referenced herein may be trademarks of Oracle Corporation. All other product and service names mentioned may be trademarks of their respective owners.

Copyright © 2000 Oracle Corporation
All rights reserved.