Technical Comparison of Oracle Database 10g vs. SQL Server 2005: Focus on Performance

An Oracle White Paper
October 2005.
Technical Comparison of Oracle Database 10g vs. SQL Server 2005: Focus on Performance

Introduction.......................................................................................................................... 3
Concurrency model.................................................................................................................. 3
Indexing .................................................................................................................................. 4
   Bitmap Indexes & Bitmap Join Indexes .............................................................................. 5
Partitioning ............................................................................................................................ 6
   Partitioning Options ........................................................................................................... 6
Parallel Execution of Operations ........................................................................................... 8
Clustering ............................................................................................................................... 8
Conclusion .............................................................................................................................. 10
INTRODUCTION

This document summarizes the main technical differences between Oracle Database 10g Release 2, the latest version of Oracle Database, and SQL Server 2005, the new release of Microsoft’s SQL Server product, focusing on the major techniques commonly used to ensure good performance and scalability in modern, enterprise-class, relational database systems: concurrency model, indexing, partitioning, parallel execution, and clustering.

CONCURRENCY MODEL

In multi-user environments, concurrency control ensures that data updates made by one user do not adversely affect those made by other users. Oracle Database 10g and SQL Server 2005 differ in their implementation of concurrency control. The main differences are summarized in the table below:

<table>
<thead>
<tr>
<th></th>
<th>Oracle Database 10g</th>
<th>SQL Server 2005</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-version read</td>
<td>Always enabled.</td>
<td>Not by default. Must be enabled.</td>
</tr>
<tr>
<td>consistency</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-escalating row-level locking</td>
<td>Yes</td>
<td>Locks escalate</td>
</tr>
</tbody>
</table>

Oracle Database’s implementation of multi-version read consistency always provides consistent and accurate results. When an update occurs in a transaction, the original data values are recorded in the database's undo records. Oracle uses the current information in the undo records to construct a read-consistent view of a table's data, and to ensure that a version of the information, consistent at the beginning of the uncommitted transaction, can always be returned to any user.

1 Note: the information about SQL Server 2005 is based on the September CTP documentation and technical articles from the Microsoft MSDN web site.
The default isolation model for SQL Server 2005 uses shared read locks for read operations (READ COMMITTED with locking). Shared locks prevent data that is read from being changed by concurrent transactions.

This implementation restricts the ability of the system to properly service concurrent requests in environments involving a mix of reads and writes. Moreover, it increases the number of locks held by applications and thus, increases the likelihood for the system to perform lock escalation and to further reduce concurrency, with more potential deadlock situations. This is why SQL Server 2005 introduces two new isolation levels:

- read committed with snapshots (statement-level read consistency)
- snapshot isolation (transaction-level read consistency)

These isolation levels correspond to Oracle’s READ COMMITTED and SERIALIZABLE isolation levels, respectively.

In these isolation levels, readers do not block other readers or writers accessing the same data. Similarly, the writers do not block readers.

Snapshot isolation level is based on row versioning, which supports read consistency by following a linked chain of versions containing committed rows of data. This linked chain is placed in a separate version store located in tempdb.

SQL Server 2005’s snapshot isolation does not represent any innovation: Oracle has supported multi-version read consistency for several releases, where it is the default behavior. Moreover, with SQL Server 2005:

1. Administrators have to explicitly establish snapshot isolation at the database level to enable read-committed with snapshot or snapshot isolation.

2. Existing SQL Server applications have to be modified to take advantage of this mode, just as SQL server applications have to be modified to run on Oracle because there are fundamental differences in the way locking mechanisms are implemented. An application designed to run using read locks has to be somewhat re-designed and re-written to run under multi-version read consistency.

INDEXING

Indexes are database structures that are created to provide a faster path to data. Using indexes can dramatically reduce disk I/O operations, thus increasing the performance of data retrieval.

---

3 See Choosing Row Versioning-based Isolation Levels in SQL Server 2005 documentation
4 Tempdb is the database that provides a storage area for temporary tables, temporary stored procedures, and other temporary working storage needs.
The differences in the indexing schemes supported by both products are summarized below:

<table>
<thead>
<tr>
<th>Type of index</th>
<th>Oracle Database 10g</th>
<th>SQL Server 2005</th>
</tr>
</thead>
<tbody>
<tr>
<td>B-tree indexes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>B-tree cluster indexes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Hash cluster indexes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Reverse key indexes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Bitmap indexes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Bitmap join indexes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Function-based indexes</td>
<td>Yes</td>
<td>No(^5)</td>
</tr>
<tr>
<td>Domain indexes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Index-organized tables</td>
<td>Yes</td>
<td>Yes (clustered indexes)</td>
</tr>
</tbody>
</table>

Both Oracle and SQL Server 2005 support traditional B-Tree indexing schemes, which are ordered lists of key values, associated with the storage location of the table row that contain these values.

Both also support index-organized tables, which are called clustered indexes in Microsoft’s terminology. Index-organized tables provide fast access to table data for queries involving exact match and/or range search on the primary key because table rows are stored in the leaf nodes of the primary key index.

However, Oracle also supports static bitmap indexes and bitmap join indexes, whose usage can provide huge performance benefits for typical load and query operations in data warehousing environments. These indexing schemes are explained further in the following section.

**Bitmap Indexes & Bitmap Join Indexes**

A bitmap index uses a bitmap (or bit vector) for each key value instead of a list of the table rows’ storage locations. Each bit in the bitmap corresponds to a row in the table. The bit is set when the table’s row contains the key value.

Bitmap representation can save a lot of space over lists of rows’ storage location, especially for low cardinality data. Bitmap indexes lend themselves to fast Boolean operations for combining bitmaps from different index entries. Bitmap indexing efficiently merges indexes that correspond to several

\(^5\) You can create an index on a computed column but the column has to exist in the table
conditions in a WHERE clause. Rows that satisfy some, but not all, conditions are filtered out before the table itself is accessed. This improves response time, often dramatically\(^6\).

With Oracle, it is also possible to create bitmap indexes on index-organized tables, thereby allowing index-organized tables to be used as fact tables in data warehousing environments.

A bitmap join index is a bitmap index for the join of two or more tables. A bitmap join index can be used to avoid actual joins of tables, or to greatly reduce the volume of data that must be joined, by performing restrictions in advance. Queries using bitmap join indexes can be sped up via bit-wise operations.

Bitmap join indexes, which contain multiple dimension tables, can eliminate bit-wise operations, which are necessary in the star transformation with bitmap indexes on single tables. Performance measurements performed under various types of star queries demonstrate tremendous response time improvements when queries use bitmap join indexes.

### PARTITIONING

Partitioning allows large database structures (tables, indexes, etc.) to be decomposed into smaller and more manageable pieces. Although it is primarily considered a feature for manageability and availability, partitioning also provides a number of performance benefits\(^7\).

### Partitioning Options

Several partitioning techniques are usually considered to handle different application scenarios:

- 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.

---

\(^6\) See Key Data Warehousing Features in Oracle Database 10g: A Comparative Performance Analysis, An Oracle White Paper, April 2005
http://www.oracle.com/technology/products/bi/db/10g/pdf/twp_bi_dw_key_features_10gr2_0405.pdf

\(^7\) For more information about Oracle’s partitioning options and the performance benefits of each of them, see Partitioning in Oracle Database 10g, an Oracle white paper, February 2005
http://www.oracle.com/technology/products/bi/db/10g/pdf/twp_dss_partitioning_10gr1_0205.pdf
and
Database Performance with Oracle Database 10g, an Oracle white paper, December 2003
• Hash partitioning uses a hash function on the partitioning columns to stripe data into partitions. Hash partitioning is an effective means of evenly distributing data.

• 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.

• Composite partitioning allows users to combine various partitioning techniques: a table can be first partitioned following one scheme, and then each individual partition is sub-partitioned following another scheme.

Indexes can be of 3 types:

• 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.

The difference in the partitioning options available for the 2 products is summarized below:

<table>
<thead>
<tr>
<th>Partitioning Option</th>
<th>Oracle Database 10g Release 2</th>
<th>SQL Server 2005</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Hash</td>
<td>Yes</td>
<td>No(^8)</td>
</tr>
<tr>
<td>List</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Composite</td>
<td>Yes:</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Range-hash</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Range-list</td>
<td></td>
</tr>
<tr>
<td>Local Indexes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Global Indexes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

\(^8\) See CREATE PARTITION FUNCTION (Transact-SQL) in SQL Server 2005 documentation
Oracle offers a wider selection of partitioning schemes, providing better support and greater flexibility to meet the business and technical requirements of all types of application scenarios.

Additionally, The two products differ greatly in their capacity limits, as can be seen in the next table.

<table>
<thead>
<tr>
<th>Maximum number of partitions per table</th>
<th>Oracle Database 10g Release 2</th>
<th>SQL Server 2005</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1024K (more than 1 million)</td>
<td>1000</td>
</tr>
</tbody>
</table>

**PARALLEL EXECUTION OF OPERATIONS**

Parallel execution of SQL operations can vastly improve the performance for operations involving large volumes of data. It helps reduce response time for data-intensive operations on large databases typically associated with decision support systems and data warehouses.

Oracle will execute INSERT, UPDATE, DELETE, and MERGE statements in parallel when accessing both partitioned and non-partitioned database objects.

With SQL Server 2005, INSERT, UPDATE, and DELETE statements are executed serially (MERGE is not supported).

**CLUSTERING**

Clusters are groups of independent servers, or nodes, connected via a private network (called a cluster interconnect), that work collaboratively as a single system. Clusters allow applications to scale beyond the limits imposed by single node systems when processing loads exceed the capacity of large individual servers.

Only Oracle provides real support for clustered configurations: with Real Application Clusters, full and transparent scalability can be obtained by simply adding new nodes as the demand increases. As indicated in Microsoft’s documentation, “SQL Server 2000 does not support this type of clustering”, which is still true for SQL Server 2005. Instead, users are forced to use a federation of databases to achieve some kind of scalability. The differences

---

9 From SQL Server 2005 documentation, in Understanding Federated Database Servers, link to Federated SQL Server 2000 Servers
between the 2 approaches are essential and have a strong impact on the performance and scalability of applications\(^{10}\):

<table>
<thead>
<tr>
<th></th>
<th>Oracle Database 10g Release 2</th>
<th>SQL Server 2005</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scalability</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Applicability to real-world pre-packaged applications</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Processing capacity</td>
<td>Unlimited</td>
<td>Limited</td>
</tr>
</tbody>
</table>

The fact that federated database servers are composed of independent databases, with no common data dictionary and no support for global indexes, imposes severe performance and scalability penalties on applications. Oracle does not have any of these limitations.

SQL Server 2005’s approach is not applicable to real-world applications either. Popular business application suites, such as the Oracle eBusiness Suite, SAP, or PeopleSoft, have thousands of tables. With SQL Server 2005’s federated databases all tables must be partitioned or replicated on all nodes to ensure some sort of scalability. Porting these applications to such an environment is a very complex and expensive process.

In contrast, Oracle Real Application Clusters provide full application compatibility: all types of applications can scale effectively without having to be specifically tailored for clustering environments. Migration is transparent: no specific redesign or code changes are required for existing applications, no explicit application segmentation or data partitioning is required.

Finally, with SQL Server, only the node that owns the partition can contribute to reading that partition. The processing power will always be limited to the processing power of the node to which the table belongs.

Oracle does not have this limitation: it can potentially deploy the whole processing power of the system, i.e., all parallel execution servers, from all nodes, to process one particular partition if necessary.

---

CONCLUSION
Oracle Database regularly outperforms its competitors on a wide variety of industry-standard and ISV-specific benchmarks, and is widely recognized as the industry leader in database scalability.

SQL Server 2005 does not come close to Oracle Database 10g’s outstanding performance and power for a wide-range of applications and demands.