Microsoft

Practical SQL Server 2008 for Oracle Professionals

SQL Server Technical Article

Published: October 2008

Applies to: SQL Server 2008

Summary: This white paper reviews many of the key comparisons between SQL Server 2008 databases and Oracle databases. It highlights SQL Server and Oracle database architecture and provides information on the latest features in SQL Server 2008. This paper is intended for Oracle professionals looking to extend and use their Oracle knowledge to manage SQL Server. Topics covered in this document include backups, database security, management options, and high availability options.

Copyright

Contents

Introduction

Both Oracle and Microsoft use the defined concepts of instances and databases. They share a lot of similarities and provide their own unique approaches. Since the products have been in the marketplace for many years and they continue to improve, some terms (“server,” “systems,” “instance,” and “database”) have been used to describe one in comparison to another. Even the vendors have been guilty of adding to the confusion. Oracle’s Database is more than just a set of database files for a single instance to use. Similarly, the product name Microsoft® SQL Server® does not limit a single instance to run on the physical computer server.

Oracle’s Implementation of Instances and Databases

An implementation of the Oracle Database consists of a set of binaries. These include the executable files for the software, network files, and administration tools. “Database files” are also installed. These items consist of control files, tablespace files, and log files.

When an instance of Oracle is started on a Windows® server, it accesses the initialization files (SPFILE or init.ora) to determine the parameters necessary to begin the service. Oracle then reserves an amount RAM from the operating system into a special buffer named the System Global Area (SGA). Then Oracle manages background processes. These background processes use Windows threads to provide code executions to the system processors and provide isolation from other applications running on the system. Once the instance is running, the database can be mounted by reading information from the database control file. Finally, the database can be opened for client requests. A database does not have to be opened or even mounted. The instance remains running and can listen to network requests.

An instance may be associated with only one database. In some environments, the design might be set to have a single database be mounted to more than instance. With the power of today’s computer systems, it often makes sense to provide multiple instances on the system. These multiple instances are separate processes and memory allocations in the operating system to provide isolation and security benefits. The limitations to the number of instances running on single systems are dependent on the processing power and memory size on the box. Of course, common sense should be used in applying resources. The additional instances can be started by using the same set of executables provided by a single installation. Multiple installations can be used to separate the access of specific binaries and versions of the product on a single system.

A single instance may support the transactions of more than just one external application. It is important to introduce some topics that distinguish an Oracle and a SQL Server solution from each other. In order to provide multiple users and application support, Oracle has provided schemas to its architecture. A schema is simply a collection of objects fulfilling a common purpose. For databases, it is a collection of logical structures of data supporting an operation or user. The products also provide management and security options. Most often a schema, which is associated with a specific user account, is used to separate different applications from each other. The objects of the schema are many and varied in Oracle.

The most obvious and used object by database administrators (DBAs) and developers is the two-dimensional table object. Other objects usually support the use of tables. The table represents a single entity that is described by it attributes (or columns). Additionally, it is represented by separate instantiations of the entity, by providing specific information associated with the abstract attributes. These are often referred to as rows.

SQL Server's Implementation of Instances and Databases

An instance created by SQL Server is a set of binaries isolating the SQL Server Database Engine services from other applications. Separate database files are also used. The binaries provide the executables, network files, and administrative tools necessary to execute or run a database on a Windows system. When an instance is executed, it gathers its configuration values and allocates memory pools (buffers) into RAM and generates threads to provide the CPU with instructions sets.

A single computer may have multiple SQL Server instances. The primary objective behind the multiple instance architecture is to help in the isolation of resources for different kinds of database activity within the same physical system. Every instance independently utilizes the memory and processor resources available on the system. SQL Server 2008 supports up to 50 instances on the Standard and Enterprise editions.

Does this sound familiar? It should. Oracle and SQL Server are both relational database management systems (RDBMSs), a variation of the DBMS type of solutions. Despite the similarities of the two products as described so far, there are some differences. Unlike Oracle, where the same binaries may be used for multiple instances, SQL Server requires a separate set of binaries to be installed for each instance. Microsoft has decided to protect not just the processes and memory of separate instances but also the physical file in order to provide isolation in case of media problems directed at the storage location holding the set of binaries. Another protection is for instances that might require an upgrade or placement of a specific file to them. This can be accomplished without affecting other instances' binaries, and in turn, applications that have not been tested with the update or may have no need for the change.

An instance is not associated with only one database. In fact, each instance installed has its own set of four "system" databases, which store specific configuration parameters for that instance. These databases provide similar functionality as the control files do in Oracle. It can be viewed as a "container" for one or more databases. Does this sound familiar as well? If you are starting to think of databases as the equivalent to schemas, you would be partially correct.

SQL Server DBAs will install an instance onto a computer. Then they may create a database, separate from the system databases mentioned earlier. This database will have objects in it, including tables. These objects will support some specific functionality. The isolation of activity provides management and security benefits. This isolation is not just for the data files but also for the transactions of each database. Oracle uses the redo logs to capture transaction activities for the instance and its one database. Like Oracle, every database is required to use a file, the log file, to track and manage transactions. Therefore, every database has its own unique transaction log system.

Storage Architecture

The introduction into blocks, extents, and segments evokes two questions. What are the consequences of having a fixed block size of 8 kilobytes (KB), and what is the effect of a fixed extent size of 64 KB with respect to internal fragmentation due to varying extent sizes? This topic is of major concern for the DBA.

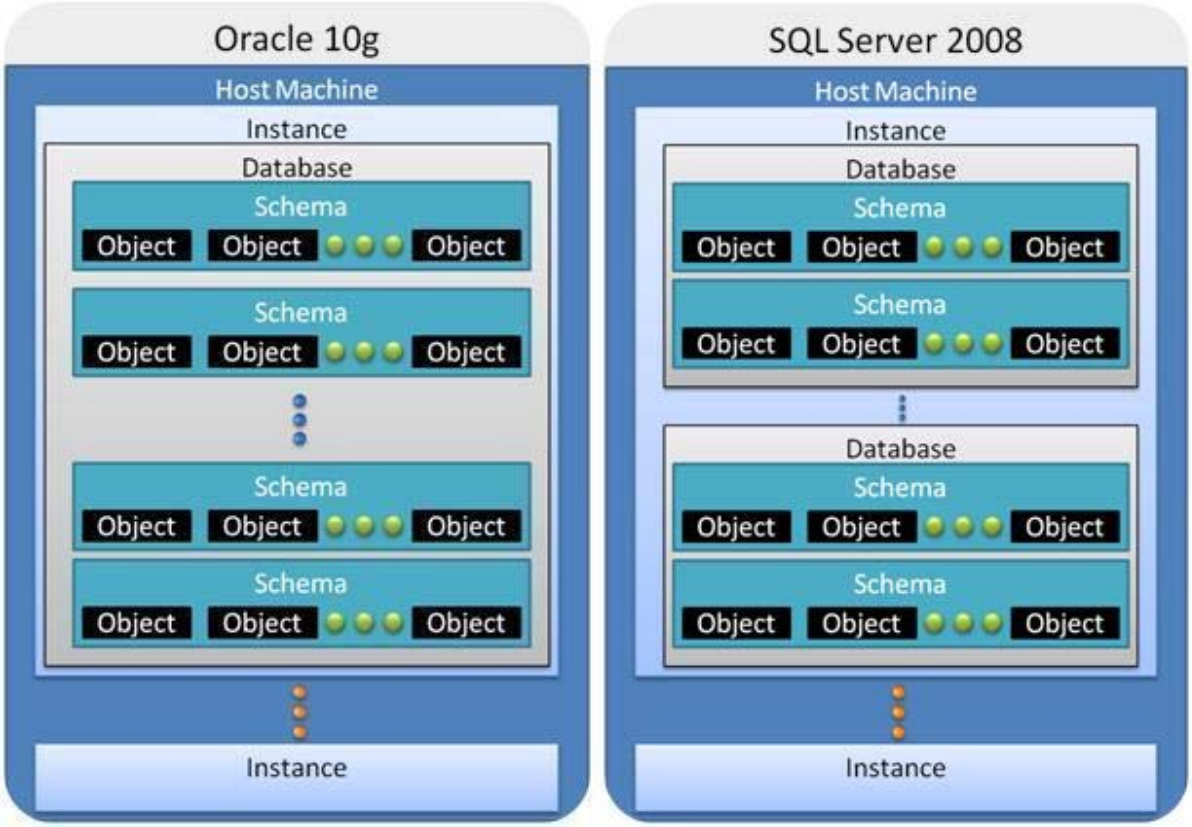

Figure 1 Database Engine organization for Oracle compared to SQL Server

In Oracle, since extents are variable in size, data may at times be written noncontiguously. In SQL Server, since all extents are the same size (64 KB), data can be written and retrieved from one location on the disk instead of reading and writing the data in numerous locations. Fragmentation within the tablespace (filegroup in SQL Server), a hotly debated topic in the Oracle world, is eliminated in SQL Server by the use of fixed size extents.

Oracle has introduced Automatic Storage Management (ASM), facilitating disk drive management. ASM allows the use of raw file system files and automates file management for an Oracle database. ASM provides management of the hard drive space as well as making adding and removing disks an easy process with Oracle rebalancing striped data devices as changes occur.

In both Oracle and SQL Server, the space needed for the definition and usage of many of the schema objects (triggers, procedures, etc.) and system-level objects (users, roles, etc.) come out of the data dictionary.

Database Mapping

Each SQL Server instance has two database structures, **model** and **msdb**, that do not have an equivalent in Oracle. The **model** database forms the template that is used to create new databases. This is roughly analogous to how a starter schema can be created using Oracle's database configuration utility. The **msdb** database is used by the SQL Server Agent application for storing information on jobs, events, schedules, operators, etc. This database combines the functionality found in Oracle's RDBMS_JOBS (DBMS_JOB and DBMS_SCHEDULER) and the Intelligent Agent repository.

The following table lists tablespaces in Oracle and the equivalent storage structures in SQL Server, where applicable.

Table 1 Oracle Tablespaces and SQL Server Storage Structures with Equivalent Function

SQL Server	Oracle
master database	System tablespace
master database	Sysaux tablespace
Resource database	System/Sysaux tablespace
tempdb database	Temporary tablespace
Transaction log	Undo (Rollback) tablespace
Transaction log	Online Redo log
"Application" database-"Data" filegroup	"Application Bigfile Data" tablespace
"Application" database-"Data" filegroup	"Application Data" tablespace
"Application" database-"Data" filegroup	"Application Index" tablespace
model database	Not available in Oracle
msdb database	Not available in Oracle

It is not possible to categorize each of the databases in a SQL Server instance as integral or self-contained in the same sense as an Oracle databases. Following are examples of autonomy as well as that of dependence on central structure:

- In SQL Server, each database has its own transaction log files, which combine the functions of the Oracle online redo logs and undo/rollback segments.
- Each SQL Server database has its own security structures such as users, privileges (permissions), and roles.
- Each SQL Server database has its own administrative roles that bestow privileges on the specific database alone.

- In SQL Server, the system catalog, which is analogous to the Oracle data dictionary, is broken up among the individual databases, the **master** database, and the (hidden and read-only) Resource database.
- The temporary space (**tempdb** in SQL Server and temporary tablespace in Oracle) is common to the entire instance.

Tablespaces and filegroups provide the ability to better distribute data across multiple files for the purposes of performance. The grouping also aids administration in terms of backup and recovery, maintenance (flipping the status to offline/online for chosen tablespaces), and other techniques such as table partitioning.

Logging Model

Online redo logs are used by Oracle to record transactional changes (DML and DDL) made to the database before those changes are committed to the database files. In SQL Server, the redo logs are called transaction logs. Oracle also uses undo tablespaces to capture an image of data before it is changed (before-images) to facilitate transaction rollback, recovery, and read consistency (multi-version concurrency).

Earlier releases of Oracle Database used rollback segments to store undo information. Oracle9i introduced automatic undo management, which simplifies undo space management by eliminating the complexities associated with rollback segment management.

Oracle uses two or more fixed size online redo logs. When a redo log is filled up, a checkpoint is initiated and the database starts writing transaction records to the next online redo log. In the meantime, all dirty database buffers related to the records in the filled log are applied to the database by a database writer (DBWR) process. As well, a copy of the redo log can be made when using archivelog mode to support future database recovery.

The transaction log in SQL Server combines the functionality of the Oracle redo logs and the undo segments. Each database in SQL Server has one or more transaction log files. The transaction log is a wrap-around log file. If the log contains multiple physical files, portions of each physical log file are used in a round-robin manner to log transactions and wrap back to the start of the first physical log file.

SQL Server segments each database log file internally into a number of “virtual log files”. Virtual log files vary in size, and there is no fixed number of virtual log files for a physical log file. SQL Server chooses the size of the virtual log files dynamically while creating or extending log files. SQL Server tries to maintain a small number of virtual files. The size or number of virtual log files cannot be configured or set by administrators; it is determined dynamically by SQL Server internally.

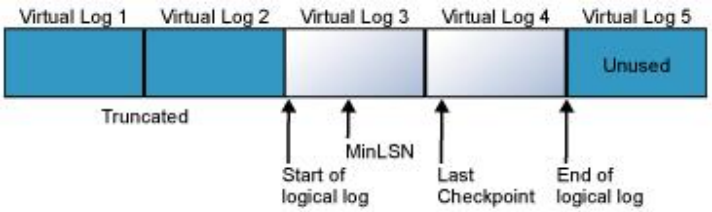

Figure 2 SQL Server transaction log architecture

New log records, identified by log sequence numbers (LSNs), are added at the end of the transaction log. Log records for data modifications record either the logical operation performed or before and after images of the modified data. Log records are stored in a serial sequence as they are created. Each log record is stamped with the ID of the transaction to which it belongs. For each transaction, all log records associated with the transaction are singly-linked in a chain using backward pointers that speed the rollback of the transaction.

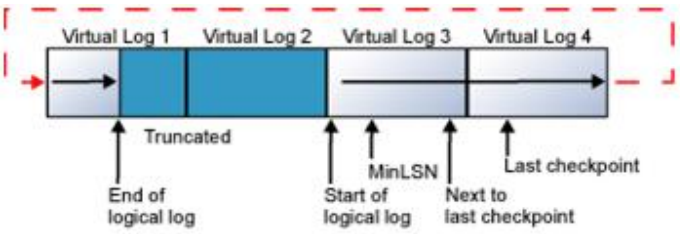

Figure 3 SQL Server transaction log wrap-around

Similar to undo segments in Oracle, virtual log entries can be overwritten if the corresponding transactions are committed. The function and administrative issues with transaction logs in SQL Server are a combination of the issues and functions associated with undo segments and redo logs in Oracle. When the log activity wraps around to the beginning of the log, space can be reused only if transactions have committed (as in undo segments), checkpoints have occurred successfully (as in redo logs), and data has been removed from the virtual log files through a Backup Log activity (similar to a switch). The Backup Log function truncates committed data. As truncation operations occur, the records in the virtual logs, before the LSN of the first log image record, or minimum recovery log sequence number (MinLSN) are deleted. Proper scheduling of backups (like scheduling Archive Log jobs) can provide an effective series of wrap-a-rounds without any contention.

In SQL Server, it is a best practice to size the transaction log appropriately. SQL Server's autogrow feature should be enabled to allow the log files to grow should the database run out of log space, allowing databases to continue running during high or unexpected transactional activity between backups when the DBA has not sized the log appropriately. Autogrow kicks in only after the log has completely run out of space, so it is better to monitor the log's free space and manually grow the log before you actually need it. The autoshrink option should typically not be used. It will cause the log files to shrink, only to have the autogrow functionality increase the

space again. These options are similar in functionality to the undo segment features, autoextend, and shrink.

Backup and Recovery

In Oracle, backup methods can be categorized at a high level as physical and logical backups. There are two ways to perform Oracle backup and recovery: Recovery Manager (RMAN) and user-managed backup and recovery. Oracle offers several options for complete recovery from an instance or disk failure: the redo log, undo records, a control file, and database backups.

RMAN is used to back up, restore, and recover databases.

Another approach is the user-managed backup and recovery, which is executed through the operating system for backups and then uses SQL*Plus for recovery. Oracle segments its backups by consistent and inconsistent states. These can also be viewed as cold or hot backups.

SQL Server offers full, differential, partial, transaction log, and tail backups, which aid in complete recovery of databases during disk, server, or instance failure. There are a variety of hot and cold backups available in SQL Server to suit any business environment. SQL Server databases can also be quickly detached and the files copied, and then they can be attached to another instance.

Logical Backups

The goal of a logical backup is to be able to recover at the individual schema object level. In Oracle, logical backups are mainly performed using the Export utility. This utility exports the schema objects into a binary file, which can only be read by the Import utility and imports the schema objects into a database.

In SQL Server, individual schema objects can be backed up to flat files in any of the supported file formats. Then flat files can be restored using tools such as the **bcp** utility (a command-line tool that uses the Bulk Copy Program, or BCP, API), the BULK INSERT command, the Import and Export Wizard, or the SQL Server Integration Services tools.

Physical Backups

Physical backups are copies of the physical database files. In Oracle, these files include data files, control files and, if the database is in ARCHIVELOG MODE, archived redo log files.

The process is similar in SQL Server. Generally, a backup in SQL Server is viewed to be at the database level. Larger databases can utilize filegroup or file backups to back up sections of a database. These are typically implemented to reduce backup time and/or data volume. Transaction Log backups hold the transaction logs. Important security components like the Service Master Key, Database Master Key, Database Encryption Key, and Certificates can be backed up in SQL Server.

Consistent/Cold Backups

In Oracle, a consistent backup of a database or part of a database is a backup in which all read/write datafiles and control files are checkpointed with the same System Change Number (SCN) which is a unique value assigned to every committed Oracle transaction. Cold backups are not an option in most mission-critical environments with high availability requirements, since they require system downtime.

In Oracle, to create a consistent whole database backup, the database must be shut down with the NORMAL, IMMEDIATE, or TRANSACTIONAL options. The next step is to make the backup while the database is closed.

SQL Server databases can be backed up while the database is online. The full online database backup offers functionality similar to Oracle's consistent or cold backup. This method will back up the entire database and part of the transaction log. However, it does not require the database to be offline; the database can have active connections while the backup is being done. After restoring from this backup, a database recovers to a consistent state and can be opened for access if you do not need to apply subsequent transaction log backups.

In SQL Server, full backups represent the database at the time the backup completed. A full online backup by itself is sufficient in all respects for a consistent database recovery in SQL Server. The backup represents the state of the database at the point when the backup completed.

Inconsistent/Online Backups

With today's demand for 24x7 availability, it is difficult to take an entire database offline. In these situations, Oracle may perform an online backup. An online backup captures the database in an inconsistent state where not all changes have been made to the SCN. Setting the mode of the database to ARCHIVELOG allows the DBA to define different tablespace and control file backup schedules. This staggers the backup over a period of time and provides the benefit of only affecting specific tablespaces versus all of them and allows individual backups to proceed faster. This can be very important when the window of low usage against the backups is less than the time necessary to do a whole backup. When using Oracle RMAN to perform the online backup, it is not necessary to place the tablespaces in backup mode.

With SQL Server, a full online database backup contains the complete database and includes part of the transaction log. Full, differential, and partial file and filegroup backups can be made using the BACKUP DATABASE statement or through SQL Server Management Studio. Transaction logs can be backed up separately as well. A full database backup does not clear the transaction log. A backup process should be initiated to clear the log periodically to prevent the log from filling up. Usually, a good backup strategy will include periodic transaction log backups, which will mitigate this issue. Any transactions that were in progress will be included in the backup.

Backing up the transaction logs also allows you to subsequently recover the database to any point in time.

Incremental Backups

Physical incremental backups are performed to capture only the changed blocks, thereby reducing the time and space needed for the backups. Incremental backups are performed after an initial complete backup has been performed. Although recovery takes longer, incremental backups are popular because of the reduction in time to perform the backup and the decreased backup space required.

In Oracle, incremental backups can be performed using RMAN or third-party tools. In the case of RMAN, datafiles are scanned and only those blocks that have changed are backed up to storage. RMAN offers two different types of incremental backups – incremental or differential. Incremental backups (level 1) affect only the data modified since the last full (level 0) or incremental (level 1) backup. A differential backup (also a level 1) gathers all data modified since the last full backup (level 0).

With SQL Server, the following backups are somewhat analogous to Oracle's incremental backups:

- **Transaction log backups**—SQL Server transaction log backups serve the same purpose as incremental backups in Oracle. This type of backup stores all the transactions that have occurred since the last transaction log backup.
- **Differential backups**—SQL Server will capture all the data changed since the last full backup. Differential backups are cumulative from the last full backup only; they are not incremental. Differential backups do not allow point-in-time recovery or marked log recovery.
- **Partial backups**—SQL Server also has partial backups, which contain only the data in the Primary and read/write filegroups of the database. Partial backups do not archive read-only filegroups. Specifying READ_WRITE_FILEGROUPS in the BACKUP statement backs up only the read/write filegroups and omits the read-only ones, assuming you already have backups of those.

Each backup type has benefits and potential drawbacks based on your environment. When implementing a backup solution, you must take into account several factors including time, volume, and recoverability. For instance, a partial backup can be performed in a shorter maintenance window than a full backup, and it may eliminate the need for more frequent transaction log backups. However, it cannot be used for complete or point-in-time recoveries.

Similarly, partial differential backups may take less time than partial backups, but during recovery both the partial backup and the differential backup may need to be used, increasing the time and complexity required for recovery operations.

Database Security

Database security is a layered approach with various mechanisms in place to regulate access to the database, the data dictionary, objects, and data. This section presents these mechanisms available in Oracle and compares them to the ones in SQL Server.

Oracle supports several important security concerns in the enterprise like data integrity and privacy, authentication, single sign-on (SSO), and access authorizations. The ubiquitous Secure Sockets Layering (SSL) is available as well as an Oracle Net native encryptor to provide data encryption. Strong authentication is a must and is implemented through standards such as Kerberos, smart cards, and digital certificates.

In SQL Server, strong authentication mechanisms and data encryption are built into the system. SQL Server provides data encryption, password protection to access databases, certificates for executing stored procedures, SSO, and, of course, Kerberos and SSL for protocol encryption.

SQL Server secures data with a hierarchical encryption and key management infrastructure. Each layer secures the layer below it, using a combination of certificates, asymmetric keys, and symmetric keys. The diagram illustrates the approach.

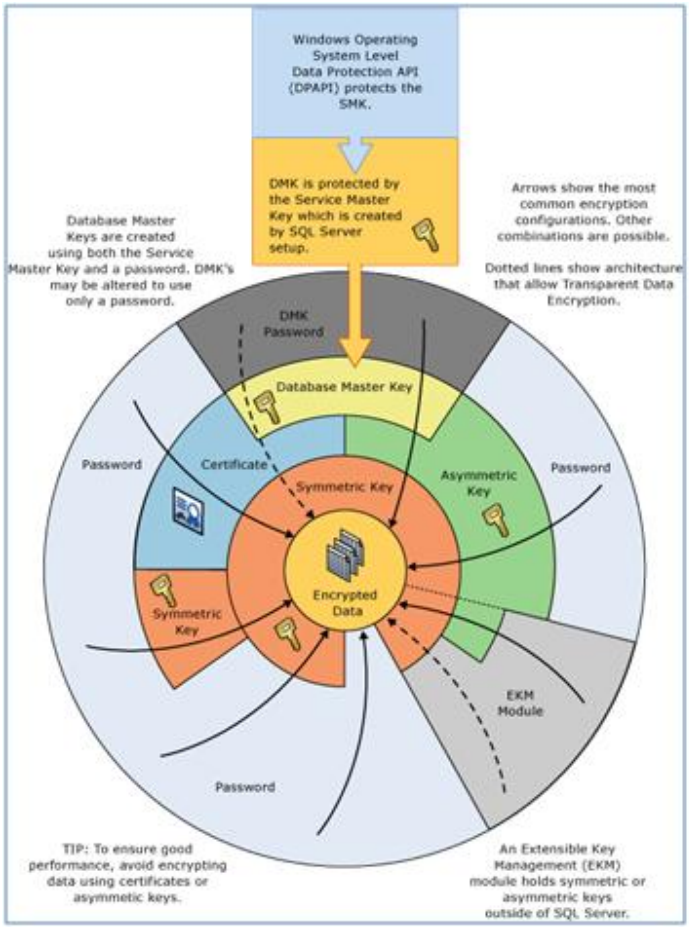

Figure 4 SQL Server data encryption

SQL Server 2008 expands upon this key management by allowing the keys to be retrieved from an external key management (EKM) source, perhaps hardware or software. This allows SQL Server to fit seamlessly into an enterprise-wide key management infrastructure.

Transparent Data Encryption

Oracle and SQL Server use encryption to help protect data. SQL Server 2008 introduces Transparent Data Encryption (TDE), and Oracle provides a form of TDE though the definition of a master key and a wallet. Oracle defines its use of TDE through table definitions and sets it at the column level.

In SQL Server 2008, TDE encrypts the data, log, and backup files of the whole database. This prevents a malicious person from obtaining the database files and simply attaching them to another SQL Server instance. The files are encrypted via a certificate, and the database can only be accessed with that certificate. The actual encryption/decryption occurs as the pages of the data or log files are written to or read from the I/O system. The data inside SQL Server's cache is in normal plaintext while the data on the disk is in encrypted form.

Once TDE is enabled, the entire database and log is then encrypted by background threads, and pages will be encrypted as they flow from/to the disk. The impact on the performance will vary depending on the access patterns of your database. The **tempdb** database will be encrypted if any database on the instance is encrypted, so performance may be impacted by any **tempdb** usage.

Of course, it is very important to back up the database's encryption certificate. Without the certificate, it is impossible to access the database or restore it. The backup of the certificate must be handled securely. As with any set of keys and locks, the data is only as secure as the key itself.

Logins

Both Oracle and SQL Server provide logins for authorized users to connect to the database. In Oracle, a login is referred to as the user or username. In SQL Server, this is called the login identifier or simply login. Any operation the user can perform is controlled on the privileges granted to the login.

There are three types of users in an Oracle database:

- Schema owners—users who create and maintain objects related to an application
- Application users—users (or systems) who manipulate data in the owning users' tables
- Administrative users—users with special roles such as database administrator, or security administrator

In SQL Server, the login enables a user to connect to an instance. However, access to other databases within the instance is not automatic and is controlled by additional accounts (called users) that are created in each of the databases to which the login requires access. The privileges at the instance level are assigned to the login, and privileges inside a database are

given to the related database user. A database user is mapped back to an instance login, though the user's name in any given database can take on a pseudonym. A SQL Server login serves as an authentication mechanism, whereas a database user supplies authorization access to database objects.

SQL Server users can also be classified into the same three types as Oracle:

- Schema owners – users who create and maintain objects related to an application.
- Application users —users (or systems) who manipulate data in the owning users' tables.
- Administrative users - users with special roles such as database administrator, or security administrator.

The account and privilege requirements for each of the three types differ. For administrators of the instance, a login will suffice, but administrators of databases will need user accounts in the target database.

Database object owners in SQL Server will need a login for the server and a user account in each of the databases that they will be creating objects in. Similarly, application users will need a login for the server and a user account in databases they will be accessing.

A special user called 'dbo' for database owner is present in every SQL Server database and has the privileges to perform any activity inside the database. An administrative login with **sysadmin** role is mapped inside each database to the dbo user. This can be viewed as being similar to the way a user connecting to an Oracle database as SYSDBA or SYSOPER is automatically connected as SYS. SQL Server logins can also be mapped to the dbo user in a database, thereby giving them administrative privileges in that database.

Authentication

Authentication is the process of verifying that the login ID or username supplied by a user to connect to the database belongs to an authorized user. Both Oracle and SQL Server allow authentication through the operating system or through the database (server). In SQL Server, the operating system mode is called Windows Authentication, and the database mode is called SQL Server Authentication.

Oracle Advanced Security supports the following authentication methods:

- Kerberos
- RADIUS
- SSL
- Entrust/PKI

SQL Server supports Kerberos Authentication as part of the Windows Server® Active Directory® domain installation. Endpoints in SQL Server can also be configured to support Kerberos authentication.

SQL Server can use SSL to encrypt all data transmitted between an application computer and a SQL Server instance on a database computer. The SSL encryption is performed within the SQL Native Client Net-Library and applies to all intercomputer protocols supported by SQL Server.

In addition, Internet Protocol security (IPsec) encryption is supported by the Windows operating system and is configured within the operating system itself and not within SQL Server. The three IETF standards-based authentication methods to establish trust between computers are as follows. All of the following types of encryption are supported in Windows:

- Kerberos v5.0 authentication provided by the Windows domain
- Public/Private Key signatures using certificates
- Passwords, termed preshared authentication keys, used strictly for establishing trust—not for application data packet protection

Passwords

Oracle passwords have the following features and functionality:

- Restrictions on composition and complexity of passwords
- Function-based complexity enforcement
- Password aging and expiration settings
- Password life-cycle enforcement (reuse, and other attributes)
- Account locking based on failed attempts, password expiration, etc.
- Password encryption

Since the SQL Server security architecture is designed to leverage Windows security, it is tightly integrated with the Windows authentication and authorization mechanism. Therefore, when you use Windows authentication in SQL Server, you can take advantage of the password protection, composition, expiration, profile, failed attempts, and encryption enforcements. SQL Server 2008 strengthens SQL Server authentication. SQL Server also makes use of the NetValidatePasswordPolicy API, available in Windows Server 2003 and later, to enforce the Windows password policies of the computer on which SQL Server is running.

The following options are available to strengthen SQL Server logins:

- CHECK_POLICY
- CHECK_EXPIRATION

- MUST_CHANGE

The options can be set with the CREATE LOGIN DDL statement or by using SQL Server Management Studio.

Scalability and High Availability

Scalability is the ability to support very large databases (VLDB) and/or large-volume OLTP. The size of the VLDBs may be due to a few large tables or a large number of smaller tables or a combination of both. OLTP scalability is measured in the number of user connections, response time, and transaction throughput.

In a broader sense, scalability is also measured in the scale-up or scale-out capabilities of the underlying architecture. Scale-up refers to increasing the capacity of server or system by adding more resources such as processors and memory.

Some of the scalability features available in Oracle and SQL Server, starting with database internal facilities and structures and progressing to major features, are discussed in the following sections.

Dynamic Memory Management

Oracle and SQL Server each have the ability to dynamically resize the components offered by their respective memory pools to handle varying workloads. This reduces the chances of paging. SQL Server's releases any unused memory when requested by the operating system, and this allocation usually goes onto the free list. If the operating system is not utilizing all of the memory for itself or other applications, SQL Server will assign the memory into its buffer cache as required. In SQL Server, the minimum amount of memory that needs to be allocated for a query execution can also be specified by the user. This user-configurable value can range from 512 KB to 2 gigabytes (GB) for a query.

Parallel SQL Statements

Parallel SQL statements are dynamically subdivided into distinct tasks distributed among multiple processors. These parallel operations:

- Are supported in symmetric multiprocessing (SMP), massively parallel processing (MPP) and clustered systems.
- Offer performance gains for queries, index creation, bulk operations (DML and data loads), aggregations, copying, and creating a table using SELECT INTO statement.
- Support intra-operation and inter-operation parallelism.

Oracle and SQL Server have very similar architectures for parallel SQL operations. Oracle and SQL Server optimizers are parallel-aware and use parallel query slaves and worker threads, respectively, to carry out the execution plan.SQL Server automatically determines the degree of parallelism (DOP) for each query, much like Oracle.

Direct-Path Inserts

During a direct-path insert in Oracle, the table data is appended into a data file, bypassing the database cache. This feature can be used by INSERT statements and the SQL*Loader utility. Direct-path insert can be executed in serial or parallel modes.

All INSERTS in SQL Server must go through the database cache. Pages are flushed at routine CHECKPOINT intervals or when there is memory pressure.

Shared Servers (Formerly MTS)

Oracle offers the shared server architecture, which uses dispatcher processes to direct user requests to a set of shared server processes, eliminating the need for dedicated server processes for each connection. This can help scale the Oracle database to handle hundreds of user connections.

Shared server mode is the default mode of operation in SQL Server. Since SQL Server runs only on the Windows operating system, it is designed to efficiently schedule connections at the thread level or in fiber mode. This ability allows SQL Server to handle hundreds or thousands of concurrent user connections.

Resumable Space Allocation

The Oracle resumable space allocation feature allows operations resulting in space-allocation errors (out of space, maximum extents reached, space quota exceeded) to be suspended instead of being rolled back. The operation in question is resumed once the database administrator corrects the problem. This feature provides huge savings during bulk operations as well as routine operations in VLDBs as long as the disk where the Oracle database resides does not run out of space.

Although resumable operation capability is not available in SQL Server, it is rarely an issue, as database files can be set to autogrow. Unless the disk partition runs out of space, the database files will not run out of space. Events can be prepared to alert DBAs when a specific threshold is reached, allowing proactive planning to add more disk space. Space quotas need not be set unless so desired and there is no limit on maximum extents.

Access Methods

To speed data access, Oracle provides B*-tree indexes (ascending/descending), Bit-mapped indexes, Function-based Indexes, Index Organized Tables, Partitioned Indexes, Join Indexes, Bitmapped Join Indexes, and Reverse-key Indexes.

SQL Server speeds data access by providing nonclustered indexes, clustered indexes, partitioned indexes, XML primary and secondary indexes, full-text indexing, and aligned indexes. Covered indexes are queries where every column used in the query is part of the index. Covered indexes can be created to improve query efficiency, and in some cases, they can physically distribute data to avoid storage hot-spotting.

Summary Management

The Data Warehousing feature in Oracle consists of mechanisms to create, use, and dynamically manage summary tables and transparently rewrite queries to use the summaries though an intelligent query rewrite mechanism. This facility automatically redirects queries against detail tables to make use of existing summary tables.

SQL Server offers indexed views, which provide the capability to materialize summary information. The optimizer automatically determines whether indexed views can be used to resolve all or part of a query that references this information.

Multiple Block Sizes

Oracle supports multiple block sizes (2KB/4KB/8KB/16KB/32KB) within the database so that objects can be placed in the right set of files for improved I/O performance. This ability to fine-tune I/O does incur management costs, since moving tables may require full export/import operations even with transportable tablespaces if the block sizes are different.

SQL Server uses a fixed block size as a page of 8 KB. While it reduces a DBA's ability to fine-tune I/O operations, it does provide for greater transportability and automation. With a fixed block size, SQL Server can automatically optimize I/O operations and memory utilization since it also knows exactly how much space is utilized with each I/O operation.

External Tables

External tables are read-only tables whose data is stored in flat files outside the database. They provide a way to access data in external (nondatabase) sources as if it were within the database without having to load the data into the database. This is a valuable tool for ETL operations in data warehousing.

SQL Server can reference external data sources through linked servers or using the OPENDATASOURCE command.

External files, such as flat files, can be read using the OPENROWSET (BULK…) command from SQL Server Management Studio or using command prompt commands with SQL Server Management Studio's SQLCMD mode. Linked servers allow connections to be made to various data sources, including spreadsheets and text files. Users can then perform insert, update, and delete operations using SQL statements against these data sources.

RAC

The Oracle Real Application Cluster (RAC) architecture, in the context of scalability features, delivers high performance and increased throughput by horizontally scaling the database on to a clustered set of servers. The server nodes access a single shared database, sharing the storage and database resources, while using their own memory and CPU. RAC distributes the work among the servers. The new Cache Fusion technology minimizes performance problems due to cache pinging (synchronization).

SQL Server uses federated database servers architecture and distributed partitioned views (DPVs) to implement a scale-out, shared-nothing solution. A federation of database servers

facilitates spreading of the process load across a group of servers by horizontally partitioning the data in a SQL Server database. These servers are managed independently, but they cooperate to process requests on the database.

Table Partitioning

Table and index partitioning addresses availability, manageability, and enhanced performance issues of very large tables providing the following:

- Smaller units of data management
- Easier placement of data on disk for improved performance
- Availability during failures or management operations
- Increased parallelism during data load, update, and batch operations
- Increased performance through the partition-cognizant optimizer, which performs partition pruning, partition-wise joins, and parallel DML on partitions

All partition functionalities at table and index level are available in SQL Server 2008 and Oracle.

SQL Server further supports a partitioned view, which is a UNION ALL view of horizontally partitioned data stored in tables (in the same or separate databases on the same or separate servers). The view is created with check constraints, which is similar to the partitioned views offered by Oracle prior to the partitioning technology.

Replication

Replication is a great way of scaling applications especially over a WAN. However, this technique is scalable only when the materialized views are used for queries only. The multi-master replication (Oracle) and peer-to-peer replication (SQL Server) techniques, which allow updates at any master site, will work well if there is a horizontal partitioning of data modified at each site that can be propagated asynchronously rather than synchronously. Transactional replication and merge replication in SQL Server also permit flow of changes from the Subscriber back to the Publisher. SQL Server replication permits propagation of schema alterations as well.

Standby Database

In Oracle and SQL Server, the scalability-related function of standby databases is their use for read-only purposes, thereby offloading resource-intensive operations, such as reporting and decision support, from the production database. The logical standby database in Oracle allows additional schema objects to be created in the standby database, which makes it more conducive to such operations. The SQL Server standby database, a database snapshot, cannot be modified in any way.

AWE Support

SQL Server and Oracle on Windows can be AWE-enabled, which allows them to reference physical memory beyond the 4-GB process limit on 32-bit systems.

64-Bit Support

The 64-bit architecture has major gains from the hardware and the operating system that positively impact scalability and performance. With the progression from 8-bit to 16-bit and 32-bit systems, we have already experienced these gains. The scalability comes from being able to use 64-bit memory or an address space of 16 million GB. The file sizes have grown from 2 GB in 16-bit file system to 4 GB in 32-bit file system and 16 exabytes (EB) in 64-bit file system. Because this large amount of memory can hold more data, there are fewer I/Os to disk and reduced paging and swapping, resulting in improved performance. Similar gains are obtained from the processors performing 64-bit integer and floating-point operations, thereby vastly improving the throughput. The numbers mentioned here are theoretical in nature. In reality, the size of addressable and direct memory access is limited by the operating system and the hardware architecture.

The only difference between Oracle and SQL Server 2008, using the same code base for both 32-bit and 64-bit versions of the database software, is in the compilation. The features of the two versions are similar. SQL Server 2008 supports AMD's 64-bit Opteron and Athlon 64 processors as well as Intel's Xeon with Intel Extended Memory 64 Technology (EMT64). On 64-bit systems, SQL Server can support up to 32 terabytes (TB) of memory. SQL Server also supports hyper-threaded and multi-core processors.

NUMA

SQL Server 2008 is Non-Uniform Memory Access (NUMA) aware. The primary benefit of NUMA is scalability. NUMA architecture surpasses the limits of Symmetric Multiprocessing (SMP) architecture. With SMP, all memory access is posted to the same shared memory bus. This works fine for a relatively small number of CPUs, but the problem with the shared bus appears when there are dozens, even hundreds, of CPUs competing for access to the shared memory bus. NUMA overcomes these bottlenecks by limiting the number of CPUs on any one memory bus and connecting the various nodes (local memory and remote memory) by means of a high-speed interconnection.

SOA

Service Oriented Architecture (SOA) provides the advancements in applications by having loosely coupled, asynchronous applications for better scalability and improved performance. Some of the SOA design principles are queuing, messages, and services.

SQL Server Service Broker

SQL Server 2005 introduced SQL Server Service Broker, which provides various capabilities for building SOA-based applications. Service Broker provides:

- A robust asynchronous programming model using Service Broker services, queues, and conversations (messages).
- Queuing and reliable messaging using Service Broker queues.

Applications and stored procedures (Transact-SQL or CLR) can take advantage of Service Broker. Typically an application or a stored procedure submits a message into a Service Broker service (this process starts a conversation). The service then puts the message to the respective queue, and then the queue activates the defined stored procedure (which then dequeues and processes the message). The stored procedure can be used to validate the message and sends a response back to the application and end the conversation. The application receives the response and ends the dialog. Service Broker provides the infrastructure for reliable and secure message routing, conversations, queues, services, and activation.

SQL Server Service Broker provides the following advantages:

- Messages can be processed as a SQL Server transaction to ensure data integrity.
- Messages are guaranteed to be processed only once, and in the same order.
- Service Broker provides reliable delivery.
- Conversations along with the data are maintained through system restarts, server/instance failover, and network outages.
- It supports flexible and rapid application development by integrating support for Service Broker within SQL Server Management Studio.
- It is easy to monitor by using System Monitor counters and runtime diagnostics utilities.

Oracle Streams Advanced Queuing

Similar to the capabilities of Service Broker, Oracle provides Oracle Streams Advanced Queuing (AQ). Within an Oracle Streams AQ, producers enqueue a message, and consumer applications dequeue messages. These messages may undergo transformations during enqueue/dequeue. Oracle Streams AQ provides the necessary support for enqueue, dequeue, propagation, transformation, security, scheduling, flow-control, and Asynchronous Notifications. The Oracle Integrated Development Environment provides support for developing Oracle Streams AQ based applications; various monitoring capabilities are built into the Management tools.

Conclusion

Microsoft SQL Server 2008 is a complete end-to-end data management solution empowering users across the enterprise by providing them with a secure, consistent, and productive platform for global data and business intelligence (BI) applications. SQL Server 2008 continues the Microsoft tradition of delivering powerful and familiar tools to information technology professionals and end users with goals to increase business value through reduced Total Cost of Ownership (TCO), ease of use, and broad integration capabilities. SQL Server 2008 also enhances its services and infrastructure by extending business intelligence systems with significantly reduced development and integration effort.

For more information:

http://www.microsoft.com/sqlserver/: SQL Server Web site

http://technet.microsoft.com/en-us/sqlserver/: SQL Server TechCenter

http://msdn.microsoft.com/en-us/sqlserver/: SQL Server DevCenter

Did this paper help you? Please give us your feedback. Tell us on a scale of 1 (poor) to 5 (excellent), how would you rate this paper and why have you given it this rating? For example:

- Are you rating it high due to having good examples, excellent screen shots, clear writing, or another reason?
- Are you rating it low due to poor examples, fuzzy screen shots, or unclear writing?

This feedback will help us improve the quality of white papers we release.

Send feedback.