Microsoft SQL Server 2005 for the Oracle Professional

SCALABILITY experts

Microsoft GOLD CERTIFIED Partner

2005 DATA MANAGEMENT & CONSOLIDATION EXCELLENCE AWARD
Advanced Infrastructure Solutions

Microsoft MANAGED Partner

Microsoft SQL Server 2005 Expert Content Provider

Microsoft GOLD CERTIFIED Partner

Business Intelligence Solutions
Advanced Infrastructure Solutions

Contents

Abstract

This whitepaper explains the key differences between Microsoft® SQL Server™ 2005 databases and Oracle databases. It compares SQL Server and Oracle database architecture and provides information on the newest features in SQL Server 2005. This paper is intended for Oracle professionals who want to leverage their Oracle knowledge to manage SQL Server or prepare to migrate an Oracle database to SQL Server 2005. Other topics discussed in this white paper include backups, database security, management options and high availability options.

Database Architecture

In Oracle, a database refers to the entire Oracle RDBMS environment and includes the following components:

- Oracle database processes and buffers (instance).
- SYSTEM tablespace containing one centralized system catalog, which is made up of one or more datafiles.
- Other optional tablespaces as defined by the database administrator (DBA), each made up of one or more datafiles.
- Two or more online Redo Logs.
- Archived Redo Logs (optional).
- Miscellaneous files (control file, Init.ora, Config.ora, and so on).

A database in Microsoft SQL Server 2005 refers to a physical grouping of set of schema objects of a database into one or more physical files. Databases are classified into system-defined database and user-defined database within SQL Server 2005. The system databases consist of the system data and also controls the temporary storage required for application data. The application data is available in a SQL Server user-defined database.

An instance of SQL Server can support multiple databases. Applications built using SQL Server can use databases to logically divide business functionality. There can be multiple instances of SQL Server on a single computer. Each instance of SQL Server can contain multiple databases.

Each SQL Server database can support filegroups, which provide the ability to physically distribute the placement of the data. A SQL Server filegroup categorizes the operating-system files containing data from a single SQL Server database to simplify database administration tasks, such as backing up. A filegroup is a property of a SQL Server database and cannot contain the operating-system files of more than one database, although a single database can contain more than one filegroup. After database creation, filegroups can be added to the database.

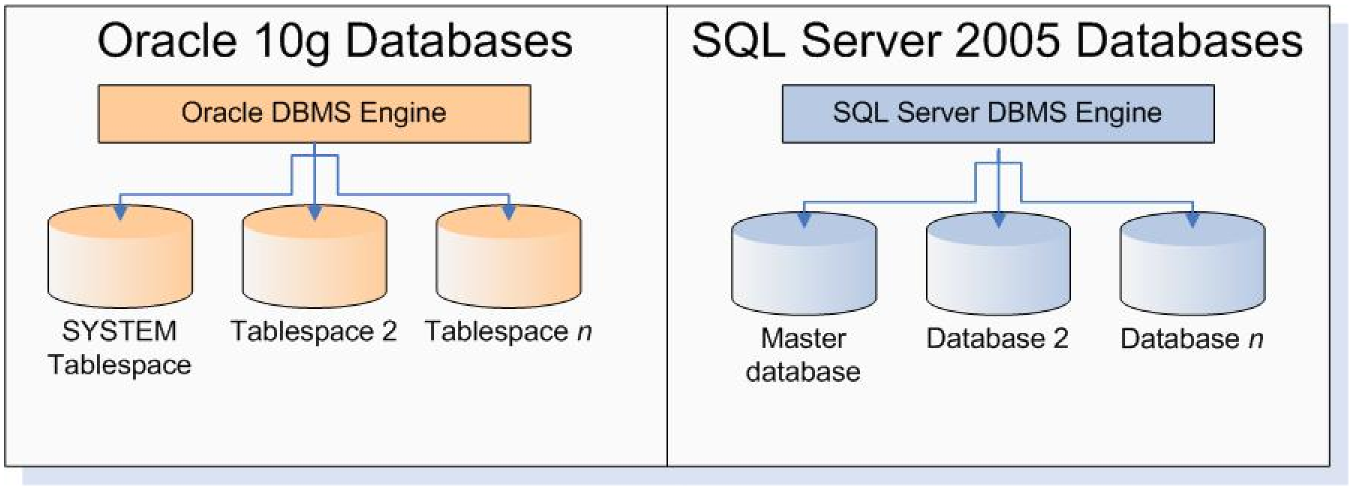

Figure 1: Oracle tablespaces and SQL Server databases compared.

Microsoft SQL Server also installs the following databases by default:

- The model database—a template for all newly created user databases and tempdb.
- The tempdb database—is similar to an Oracle temporary tablespace in that it is used for temporary working storage and sort operations. Unlike the Oracle temporary tablespace, users can create temporary tables that are automatically dropped when the user logs off.
- The msdb database—which supports the SQL Server Agent and its scheduled jobs, alerts, and replication information.
- The AdventureWorks and AdventureWorksDW databases—are sample databases for testing and training purposes that can be optionally installed.

For more information about the default databases, see SQL Server Books Online.

Note: You can download and install SQL Server Books Online without installing SQL Server at the following http://www.microsoft.com/technet/prodtechnol/sql/2005/downloads/books.mspx. Books Online is a comprehensive resource for SQL Server which includes detailed documentation, code samples and tutorials.

Database System Catalogs

Each Oracle database runs on one centralized system catalog, or data dictionary, which resides in the SYSTEM tablespace. Each Microsoft SQL Server 2005 database maintains its own system catalog, which contains information related to:

- Database objects (tables, indexes, stored procedures, views, triggers, and so on).
- Constraints.

- Users and permissions.
- User-defined data types.
- Schema Objects
- Files used by the database.
- Replication definition
- Snapshot definitions

The system level information for an instance of SQL Server 2005 is recorded by the master database. The recorded information includes the following:

- Database names and the primary file location for each database.
- SQL Server login accounts.
- System messages.
- Database configuration values.
- Remote and/or linked servers—a feature in SQL Server 2005 which enables you to execute commands against OLEDB Data sources on remote servers (including Oracle).
- Current activity information.
- Endpoint information—endpoints are means of implementing TCP protocol-based interfaces for implementation of certain features including HTTP web services, Service Broker, Database mirroring.
- System stored procedures.

In SQL Server 2005, the system objects are not stored in the master database but stored in a hidden database, called the resource database.

The system catalog is used in SQL Server 2005 in order to retrieve metadata information about objects in a database. The system catalogs are accessible to the user as catalog views. In order to monitor health of a server instance, diagnose problems and tune performance, the SQL Server 2005 Dynamic Management views can be used. The dynamic management views in SQL Server are similar to the V$_ views in Oracle which are used for performance monitoring.

Like the SYSTEM tablespace in Oracle, the SQL Server master database must be available to access any other database. As such, it is important to protect against failures by backing up the master database after any significant changes are made to the database. Database administrators can also mirror the files that make up the master database.

For a detailed information about the system catalog and the Dynamic Management views, search for “System Catalog” and “Dynamic Management Views” in Books Online.

Physical and Logical Storage Structures

The Oracle RDBMS is comprised of tablespaces, which in turn are comprised of datafiles. Tablespace datafiles are formatted into internal units called blocks. The block size is set by the DBA when the Oracle database is first created. When an object is created in an Oracle tablespace, the user can specify its space in units called extents (initial extent, next extent, min extents, and max extents). If no extent size is explicitly defined, a default extent is created. An Oracle extent varies in size and must contain a chain of at least five contiguous blocks.

SQL Server utilizes a fixed 8 KB page size which is the basic unit of IO. To better manage pages, SQL Server organizes these pages into groups of eight pages which are physically contiguous to each other, called extents. Although IO is performed a page at a time, space is managed in terms of extents. Each page belongs to one object and is of one type (data, index, GAM, IAM, etc), but an extent may belong to multiple objects (up to eight).

An extent that has all eight pages belonging to the same object is referred to as a Uniform Extent; otherwise, it is referred to as a mixed extent. Although a row may not span more than one page (8060 bytes – page minus header space), SQL Server 2005 may move data to ROW_OVERFLOW_ALLOCATION unit. Extents live within a filegroup. The following diagram provides a graphical comparison of how blocks and extents in Oracle map to pages and extents in SQL Server.

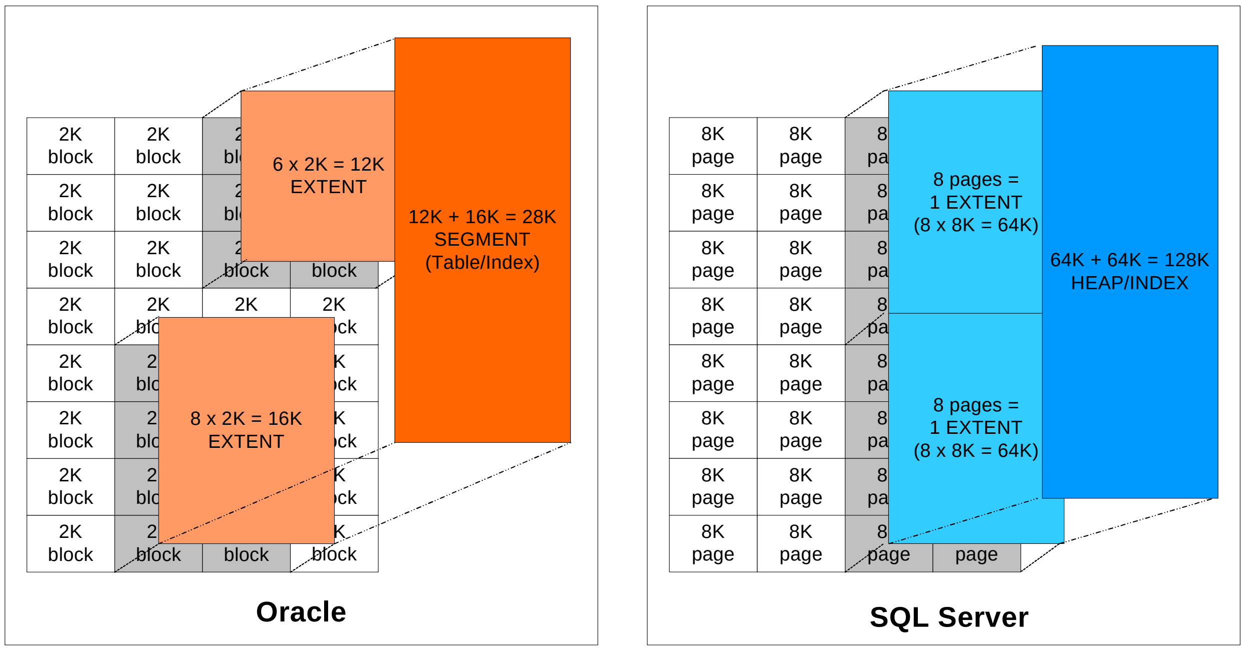
2K block
6 x 2K = 12K EXTENT
8 x 2K = 16K EXTENT
12K + 16K = 28K SEGMENT (Table/Index)
Oracle
8K page
8 pages = 1 EXTENT (8 x 8K = 64K)
8 pages = 1 EXTENT (8 x 8K = 64K)
64K + 64K = 128K HEAP/INDEX
SQL Server

Figure 2: Comparing Oracle and SQL Server pages and extents.

SQL Server uses filegroups at the database level to control the physical placement of tables and indexes. Filegroups are logical containers of one or more files, and data contained within a filegroup is proportionally filled across all files belonging to the filegroup.

If filegroups are not defined and used, database objects are placed in a default filegroup that is implicitly defined during the creation of a database. Filegroups allow you to:

- Distribute and partition large tables across multiple files to improve I/O throughput.
- Store indexes on files different from their respective tables, to improve I/O throughput and disk concurrency and also to partition these indexes.
- Store text, ntext, and image columns (large objects) on separate files from the table.
- Place database objects on specific disk spindles.
- Back up and restore individual tables or sets of tables within a filegroup.

Additionally, SQL Server 2005 has two types of temporary tables, user and global. The *user* temporary table is identified with a single leading pound sign (#) and is created in the TEMPDB database. When the connection that created the user temporary table is closed, the table is dropped. The *global* temporary table is identified with two leading pounds signs (##) and is also created in the TEMPDB database. Unlike the user temporary table, the global temporary table will be persisted until it is manually dropped or until server is re-started where the TEMPDB database (and all objects within) is dropped. This differs from Oracle which supports only global temporary tables though the functionality does overlap.

For more information, see "Physical Database Architecture" in SQL Server Books Online.

Data Files on Disk

Oracle-type segments are not needed for most Microsoft SQL Server installations. Instead, SQL Server can distribute, or stripe, data more efficiently with hardware-based RAID or with software–based RAID solutions available through the Windows disk administration infrastructure or from third party software. Though software RAID is supported, hardware RAID solutions are strongly recommended for their superior performance and resilience.

The recommended RAID configuration for SQL Server is RAID 1+0 (mirroring of a striped set) or RAID 1 (mirroring) for the data files. RAID 1+0 is considerably more expensive to implement but it offers the greatest performance and protection from failures. RAID 5 (stripe sets with parity) may be implemented for SQL Server data files that are used mostly for read operations.

Transaction log files operations are sequential in nature so RAID 1 is the typical recommendation. Not many systems will require striping log files for performance unless the system has very high rates of write operations. As such, mirroring the log files for protection against failure is usually adequate so long as the disk can support the number of IO operations required by the application. Further details on SQL Server's transaction log architecture are discussed in the section below on Transaction Logs and Automatic Recovery.

Background Processes

SQL Server 2005 also uses several different background processes to efficiently manage computer resources. One example is checkpoints. SQL Server periodically generates automatic checkpoints in each database. Checkpoints flush dirty data and log pages from the buffer cache of the current database, minimizing the number of modifications that have to be rolled forward during a recovery.

Another background process is called Lazywriter which is unique to each instance. The lazywriter process sleeps for an interval of time then wakes to scan through the buffer cache where it checks the size of the free buffer list. If the free buffer list is below a certain point (dependent on the size of the cache) the lazywriter process scans the buffer cache to reclaim unused pages and write dirty pages that have not been recently referenced, while frequently referenced pages remain in memory. If Lazywriter is used on a NUMA-capable server then a lazywriter process will be spawned for each NUMA node for an instance.

While SQL Server 2005 made LazyWriter to control only data pages in the buffer memory, another thread called ResourceMonitor was introduced to control behavior the rest of the memory caches and clerks. Its role is to react to internal and external memory pressures dynamically. The ResourceMonitor background thread won't allow overall cache memory to exceede 70% of Buffer Pool. If SQL Server's prediction mechanism detects such a possibility, the ResourceMonitor starts running and shrinks the caches. In addition, ResourceMonitor won't allow any single cache to be larger than 50% of the Buffer Pool.

Partitioning

Partitioning in SQL Server 2005 enables more efficient management of large tables in the database. Partitioned Views has been part of SQL Server for many years, but Table and Index Partitioning features, new in SQL Server 2005 make it easier to implement horizontal partitioning of tables and related indexes.

While generally not required for most databases, there are some significant benefits when partitioning very large tables (over tens of millions of rows). The benefits include:

- Improved manageability—various management operations are easier and faster to perform on partitioned tables. For example, data can be loaded or unloaded by partitions then added to or removed from an existing partitioned table. Partitioning allows users to deal with just volatile portions of the data which is usually a small subset of the table instead of the entire table. This is also a significant benefit for users that have “sliding window” scenarios where portions of data need to be removed or archived periodically so that only active data is kept in the main database.

- Improved performance—with the ability to perform resource demanding operations such as loading large volumes of data or performing index management operations against large table offline and in smaller chunks, the overall performance of the system is maintained. For example, even though rebuilding an index can be performed online with SQL Server 2005, doing so with a very large table will consume a sizeable amount of resources. With a partitioned table, users can re-index only the volatile partition(s) leaving the static partitions untouched. That will consume significantly less system resources such as memory.

- Reduced cost—partitioned tables further allow easy placement of specific partitions onto designated storage devices. For example, in a large table that tracks financial information, historical data that no longer changes (only kept online for read access) can be stored on lower cost storage with minimal rapid recovery features. Current data can be stored on high performance storage with redundancy and rapid recovery capabilities. This can result in significant cost savings especially in systems that need to store historical data for many years for accounting or regulatory compliance purposes.

SQL Server 2005 currently supports range partitioning and other schemes are planned for future support.

Transaction Logs and Automatic Recovery

The Oracle RDBMS performs automatic recovery each time it is started. Oracle verifies that the contents of the tablespace files are coordinated with the contents of the online redo log files. If they are not, Oracle applies the contents of the online redo log files to the tablespace files (roll forward), and then removes any uncommitted transactions that are found in the rollback segments (roll back). If Oracle cannot obtain the information it requires from the online redo log files, it consults the archived redo log files.

Microsoft SQL Server 2005 also performs automatic data recovery by checking each database in the system each time it is started. SQL Server first checks the master database and then launches threads to recover all of the other databases in the system. For each SQL Server database, the automatic recovery mechanism checks the transaction log. If the transaction log contains any uncommitted transactions, the transactions are rolled back. The recovery mechanism then checks the transaction log for committed transactions that have not yet been written out to the database. If it finds any, it performs those transactions again, rolling forward.

Each SQL Server transaction log has the combined functionality of an Oracle rollback segment and an Oracle online redo log. Each database has its own transaction log that records all changes to the database and is shared by all users of that database. When a transaction begins and a data modification occurs, a BEGIN TRANSACTION event (as well as the modification event) is recorded in the log. This event is used during automatic recovery to determine the starting point of a transaction. As each data modification statement is received, the changes are written to the transaction log prior to being written to the database itself.

SQL Server has an automatic checkpoint mechanism that ensures completed transactions are regularly written from the SQL Server disk cache to the transaction log file. A checkpoint writes any cached page that has been modified since the last checkpoint to the database. Checkpointing these cached pages, known as dirty pages, onto the database, ensures that all completed transactions are written out to disk. This process shortens the time that it takes to recover from a system failure, such as a power outage. This setting can be changed by modifying the recovery interval setting by using SQL Server Management Studio or the Transact-SQL sp_configure system stored procedure.

Virtual Log Files

Internally, transaction logs are broken down into smaller granular chunks called virtual log files (or VLFs). VLFs are the unit of truncation for the transaction log. The purpose of VLFs is to allow for easier and automated management of the transaction log. When a VLF no longer contains log records for active transactions, it can be truncated and the space becomes available to log new transactions.

The smallest size for a virtual log file is 256 KB. The minimum size for a transaction log is 512 KB, which provides two 256 KB VLFs. The number and size of the virtual log files in a transaction log increase as the size of the log file increases.

As records are written to the log, the end of the log grows from one virtual log file to the next. If there is more than one physical log file for a database, the end of the log grows through each virtual log file in each physical file before circling back to the first virtual log file in the first physical file. Once all log files are full, the log will begin to grow automatically (if configured to allow auto-grow). The size and number of VLFs you'll have depends largely on the size that the chunk is when it's added to the transaction log.

- If you add a new chunk to the transaction log which is 20 MB (through auto growth or through manual growth) then the number of VLFs that are added is 4.
- If you add a chunk which is greater than 64 MB but less than or equal to 1 GB, you'll add 8 VLFs.
- If you add more than 1 GB then you'll add 16 VLFs.

In general, most transaction logs will only have 20 or 30 VLFs - even 50 could be reasonable depending on the total size of the transaction log. However, in many cases what happens is that excessive auto-grow occurrences can cause an excessive number of VLFs to be added - sometimes resulting in hundreds of VLFs. Excessive VLFs adds overhead to log related activities such as transaction logging, log backups, replication log reader performance, and triggers. The general rule of thumb is to avoid frequent auto growth by pre-allocating the transaction log size such that auto growth is not likely to occur. SQL Server's auto-grow feature (available for both data and log files) is intended to be a "safety-net" in the event of significant unplanned storage use. It is not intended to replace storage capacity planning and deployment.

Backing up Data in SQL Server 2005

Like Oracle, Microsoft SQL Server 2005 offers several options for backing up data, including:

- Full Backups
- Full Differential Backups
- Partial Backups
- Partial Differential Backups
- File and Filegroup Backups
- Transaction Log Backups

All backups can be performed while the database is in use, allowing backups to be made on high availability databases. The backup processing and internal data structures of SQL Server maximize their rate of data transfer with minimal effect on transaction throughput.

Both Oracle and SQL Server require a specific format for backup files. In SQL Server, these files are called backup devices and are created using SQL Server Management Studio or the Transact-SQL sp_addumpdevice stored procedure.

Although backups can be performed manually, it is recommended that you use SQL Server Management Studio and/or the Maintenance Plan Wizard to schedule periodic backups or backups based on database activity.

Each backup option is discussed in detail below.

Full Backup

A full backup (also known as a database backup) backs up the entire database, including part of the transaction log (so that the full backup can be recovered). A full back up is a representation of the database at the time of the backup being successfully completed. By developing a backup strategy that combines full backups with partial transaction log backups allows the recovery of the database to a specific point of time.

A full backup can be executed from the Backup task in the SQL Server Management Studio or by using the BACKUP DATABASE statement. Like other backup methods, full backup can also be scheduled using SQL Server Agent or via the Database Maintenance Wizard.

Full Differential backup

After a full backup, backing up incremental changes to the database can be performed using a full differential backup. The full differential backup can be implemented using the SQL Server Management Studio back up task and also using the BACK UP DATABASE WITH DIFFERENTIAL statement. The Differential option backs up only incremental changes to the database. As such, the backup occupies less space and takes less time to complete than a full backup. However, a full backup is required for recovery along with a differential backup.

Partial Backup

Partial backup is an enhanced feature in SQL Server 2005. A partial backup consists of each file in the default Primary Filegroup and every read/write Filegroup and specific read-only files that are specified using the Backup command. The partial backup is a new feature introduced in SQL Server 2005 in order to add more flexibility to the backup procedure in SQL Server databases. It is an ideal model to back up write intensive portions of a database.

Partial Differential Backup

Partial differential backups are used only with partial backups. A partial differential contains only those extents changed in the primary filegroup and read-write filegroups as of the time of backup. If only some of the data captured by a partial backup has changed, using a partial differential backup allows a database administrator to enable creation of smaller and faster backups. However, restoring from partial differential backups will take longer than restoring from a partial backup and be more complex, since two separate backup files are restored.

File backup, Filegroup backup

A file or filegroup backup copies one or more files of a specified database individually, allowing a database to be backed up in smaller units: at the file or filegroup level. This can increase the speed of recovery in some instances by allowing users to restore only damaged files without restoring the rest of the database. For example, a database that has several data files located on several disks and one of the disks fails, only the failed disk needs to be restored.

Transaction log backup

Transaction logs in SQL Server 2005 are associated with individual databases. The default configuration of SQL Server 2005 allows the transaction log to grow automatically as needed. The log will grow until it uses all available disk space or it reaches the maximum configured size. It is best to avoid filling the transaction log in a SQL Server environment. If the transaction log is full, data modifications within in that database can be impacted. Other databases are not affected. Transaction logs can be backed up using the BACKUP LOG statement or by using the Backup task in SQL Server Management Studio. Upon successful completion of a transaction log backup, the portion that has been backed up is truncated and available for use by new transactions.

A database can be restored to a certain point in time by applying transaction log backups or differential backups to a full database backup. A database restore overwrites the data with the information contained in the backups. Restores can be performed using SQL Server Management Studio or the RESTORE DATABASE statement. Online restores are supported in SQL Server 2005.

Network Configuration Components

Oracle Net Services supports networked connections between Oracle database servers and their clients. It communicates with the Transparent Network Substrate (TNS) data stream protocol, and allows users to run many different network protocols without writing specialized code.

With Microsoft SQL Server, client communication with a server supports primarily two components which include the following:

- Network Protocol on the Client and Server
- SQL Server Native Client (SNAC)—This is a new technology introduced in SQL Server 2005 which combines the functions of OLEDB and ODBC for SQL Server into one native library. The native client supports Net-libraries and TDS (Tabular Data stream) endpoints for multiple protocols in SQL Server. TDS is the internal client/server data transfer protocol in SQL Server.

  SNAC is loaded with the installation of SQL Server 2005 and does not require a separate install. It can be installed separately on machines which do not have SQL Server installed, but requires SQL Server access.

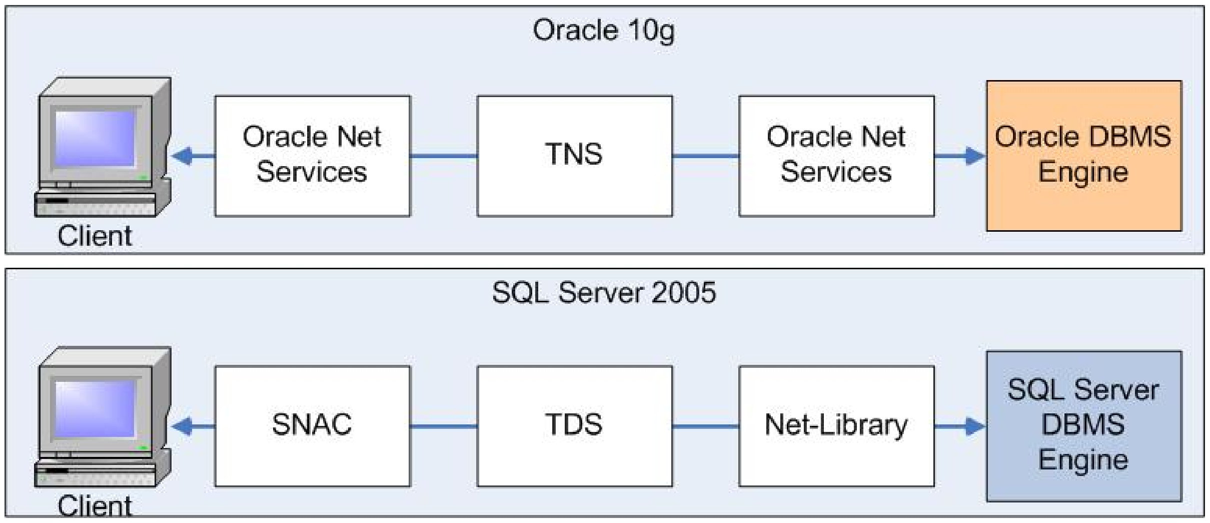

Figure 3: Comparing the network components of Oracle and SQL Server

SNAC configuration can also be performed after the installation of SQL Server using the SQL Server Configuration Manager.

SQL Server Security Model

Whether you are trying to map your Oracle skills and knowledge to SQL Server or trying to migrate Oracle applications to Microsoft SQL Server 2005, it is important to understand how SQL Server implements database security and roles. This section describes SQL Server's security architecture and features and maps them to Oracle equivalents where appropriate.

Network Security

Microsoft SQL Server 2005 can use Secure Sockets Layer (SSL) to encrypt data transmitted across a network between an instance of SQL Server and a client application. The SSL encryption is performed within the SQL Native Client Net-Library and applies to all inter-computer protocols supported by SQL Server 2005.

SSL encryption works with instances of SQL Server running on a computer that has been assigned a certificate from a public certification authority. The computer on which the application is running must be configured to trust the certificate's root authority. (Encryption with a self-signed certificate is possible and described in the next section, but a self-signed certificate offers only limited protection.)

The Net-Library encryption is implemented using the Secure Sockets Layer API. The level of encryption, 40-bit or 128-bit, depends on the version of the Microsoft Windows operating system that is running on the application and database computers.

Login Accounts

A login account allows a user to access a SQL Server Instance and its administrative options. It allows connection to the instance but does not carry data access privileges. A login account is typically mapped to a database user name. Accessing data and/or administration functions are controlled via user names.

There is a guest login account that allows users to log in to SQL Server and only view databases that allow guest access. However, inline with security best practices, the guest account is not set up by default and must be created manually if needed.

Database User Name

Each Windows account or SQL Server login must be associated with a user name in each database that the user is authorized to access, or the database must have guest access enabled. Database user names are defined by members of the db_owner (database owner) or db_accessadmin (databases access administrator) fixed database role, and are stored in the sys.database_principals table found in each database.

SQL Server 2005 supports two types of login names which are managed and authenticated using different methods; Windows authentication and SQL Server authentication.

Windows Authentication

A DBA can specify which Windows login accounts can be used to connect to an instance of SQL Server 2005. Users logged into Windows using these accounts can connect to SQL Server 2005 without having to specify a separate database login and password. When using Windows Authentication, SQL Server 2005 uses the security mechanisms of NTLM or Kerberos to validate login connections, and relies on a user's Windows security credentials. Users do not need to enter login IDs or passwords for SQL Server 2005 because their login information is taken directly from the trusted network connection. Windows authentication is similar in functionality to the IDENTIFIED EXTERNALLY option associated with Oracle user accounts.

SQL Server Authentication

A DBA can also define a separate database login account. A user must specify this login account and its password when they attempt to connect to SQL Server 2005. The database login is not related to the user's Windows login. SQL Server authentication is similar to the IDENTIFIED BY PASSWORD option associated with Oracle user accounts.

Authentication Modes

With the two authentication methods, SQL Server 2005 provides two modes for authenticating users. DBAs can choose the most appropriate mode based on application and system requirements and constraints:

1. Windows Authentication Mode (also known as Integrated Security in earlier versions of SQL Server) — in this mode, SQL Server 2005 allows only connections that use Windows Authentication. Users will not be able to login using a SQL Server login ID.
2. Mixed Mode — in this mode, connections can be made using either Windows Authentication or SQL Server Authentication. Users can connect to SQL Server using either their Windows login account (if they have been granted permission) or a valid SQL Server login account.

For more information about these security mechanisms, see SQL Server Books Online.

Groups, Roles, Schemas and Permissions

SQL Server and Oracle use permissions to enforce database security. SQL Server statement-level permissions are used to restrict the ability to create new database objects (similar to the Oracle system-level permissions).

SQL Server also offers object-level permissions. As in Oracle, object-level ownership is assigned to the creator of the object and cannot be transferred. Object-level permissions must be granted to other database users before they can access the object. Members of the sysadmin fixed server role, db_owner fixed database role, or db_securityadmin fixed database role can also grant permissions on one user's objects to other users.

Permissions can be granted, denied or revoked on any SQL Server securable. Granting specific permissions to a principal (logins, users or roles) allows the principal to perform specific tasks. Revoking that permission takes away the ability of the principal to perform the tasks. Deny places an explicit blocker on a securable that prevents a principal(s) from performing specific actions against the securable and always takes precedence over all other permissions.

For example, if user JOE belongs to the SUPERUSER role and the SUPERUSER role has been granted select, insert, update and delete permissions on the *contacts* table, JOE will be able to perform those four tasks against the *contacts* table because JOE inherited the permissions from the SUPERUSER role. However, if a DENY is explicitly set against user JOE on the *contacts* table for all four actions, JOE will not be able to perform any of the actions since DENY always takes precedence.

SQL Server statement-level and object-level permissions can be granted directly to database user accounts. However, it is often easier to administer permissions to database roles. SQL Server roles are used for granting and revoking privileges to groups of database users (much like Oracle roles). Roles are database objects associated with a specific database. There are a few fixed server roles associated with each installation, which apply across all databases. An example of a fixed server role is sysadmin. Windows groups can also be added as SQL Server logins, as well as database users. Permissions can be granted to a Windows group or a Windows user.

A database can have any number of roles or Windows groups. The default role public is always found in every database and cannot be removed. The public role functions much like the PUBLIC account in Oracle. Each database user is always a member of the public role. A database user can be a member of any number of roles in addition to the public role. A Windows user or group can also be a member of any number of roles, and is also always in the public role.

Database Users and the guest Account

In Microsoft SQL Server, a user login account must be authorized to use a database and its objects. One of the following methods can be used by a login account to access a database:

- The login account can be specified as a database user.
- The login account can use a guest account in the database.
- A Windows group login can be mapped to a database role. Individual Windows accounts that are members of that group can then connect to the database.

Members of the db_owner or db_accessadmin roles, or the sysadmin fixed server role create the database user account roles. An account can include several parameters: the SQL Server login ID, database user name (optional), and up to one optional role name. The database user name does not have to be the same as the user's login ID. If a database user name is not provided, the user's login ID and database user name are identical. If a role name is not provided, the database user is only a member of the public role. After creating the database user, the user can be assigned to as many roles as necessary. However, it is strongly recommended that users be assigned to the minimum number of roles required to perform their tasks inline with the principle of least privileges for secure systems.

Members of the db_owner or db_accessadmin roles can also create a guest account. The guest account allows any valid SQL Server login account to access a database even without a database user account. By default, the guest account inherits any privileges that have been assigned to the public role; however, these privileges can be changed to be greater or less than that of the public role. The guest user account exists by default in every database but is disabled. You need to explicitly enable the guest account to allow connections.

A Windows user account or group account can be granted access to a database, just as a SQL Server login can. When a Windows user who is a member in a group connects to the database, the user receives the permissions assigned to the Windows group. If a member of more than one Windows group that has been granted access to the database, the user receives the combined rights of all of the groups to which he belongs.

The sysadmin Role

Members of the Microsoft SQL Server sysadmin fixed server role have similar permissions to that of an Oracle DBA. In SQL Server, the sa SQL Server Authentication Mode login account is a member of this role by default, as are members of the local Administrators group if SQL Server is installed on a Windows –based computer. A member of the sysadmin role can add or remove Windows users and groups, as well as SQL Server logins. Members of this role typically have the following responsibilities:

- Installing SQL Server.
- Configuring servers and clients.
- Creating databases.*
- Establishing login rights and user permissions.*
- Transferring data in and out of SQL Server databases.*
- Backing up and restoring databases.*
- Implementing and maintaining replication.
- Scheduling unattended operations.*
- Monitoring and tuning SQL Server performance.*
- Diagnosing system problems.*

The tasks designated with an asterisk (*) can be delegated to other security roles or users.

A member of the sysadmin fixed server role can access any database and all of the objects (including data) on a particular instance of SQL Server. As in Oracle, there are several commands and system procedures that can only be issued by members of the sysadmin role.

The db_owner Role

Although a Microsoft SQL Server database is similar to an Oracle tablespace in use, it is administered differently. Each SQL Server database is a self-contained administrative domain. Each database is assigned a database owner (dbo). This user is always a member of the db_owner fixed database role. Other users can also be members of the db_owner role. Any user who is a member of this role can manage the administrative tasks related to her database (unlike Oracle, in which one DBA manages the administrative tasks for all tablespaces). These administrative tasks include:

- Managing database access.
- Changing database options (read-only, single user, and so on).
- Backing up and restoring the database contents.
- Granting and revoking database permissions.

- Creating and dropping database objects.

Members of the db_owner role have permissions to do anything within their database. Most rights assigned to this role are separated into several fixed database roles, or can be granted to database users. It is not necessary to have sysadmin server-wide privileges to have db_owner privileges in a database.

Schemas

SQL Server 2005 supports a new feature called schemas. A schema is a collection of database entities that form a single namespace. A namespace is a set in which every element has a unique name. This helps in creating a logical separation between the application user and the database objects.

The abstraction of database users from schemas provides several benefits to administrators and developers as listed below:

- Multiple users can own a single schema through membership in roles or Windows groups. This extends the familiar functionality allowing roles and groups to own objects.
- Dropping database users is greatly simplified as it does not require the renaming of objects contained within that user's schema. Schema owners can also be changed with no impact to existing application code. Thus it is no longer necessary to revise and test applications that refer explicitly to schema-contained objects after dropping the user that created them.
- Multiple users can share a single default schema for uniform name resolution.
- Shared default schemas allow developers to store shared objects in a schema created specifically for a specific application, rather than in the DBO schema.
- Permissions on schemas and schema-contained objects can be managed with a higher degree of granularity than in earlier releases.
- Schemas increases the level of granularity at which permissions are managed. Permissions can be given for a specific user to manage and own objects within a schema as compared to the entire database. For instance, User 1 can be given permissions to modify the objects in the “Sales" schema within a database but the user will be restricted from modifying objects in the “inventory” schema within the same database.

Data Encryption

SQL Server 2005 natively supports all encryption infrastructure requirements natively within the database and are fully integrated with a key management infrastructure. 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, or symmetric keys. This feature works with Windows Server 2003 or later.

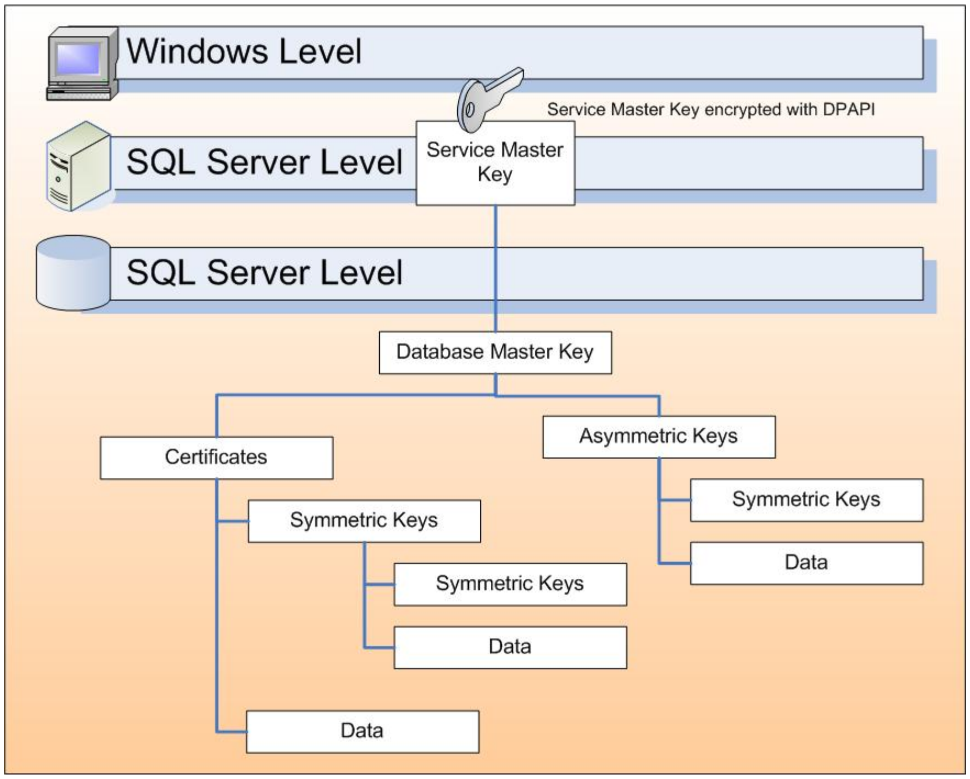

Figure 4: SQL Server data encryption

Each SQL Server instance has a Service Master Key, created automatically during setup, which is encrypted with a Data Protection API (using the credentials of the SQL Server service account) provided by the underlying Windows Server 2003 operating system. Its main purpose is to secure system data, such as passwords used in instance-level settings such as linked servers or connection strings).

The Service Master Key is also used to secure each of the Database Master Keys (protected additionally with a password supplied when a Database Master Key is created). Within each database, its master key serves as the basis for creating certificates or asymmetric keys, which subsequently can be applied to protect data directly or to further extend the encryption hierarchy (i.e., by creating symmetric keys). Creation, storage, and other certificate and key management tasks can be handled internally within SQL Server, without resorting to features of the operating system or third party products. Encryption and decryption is provided by pairs of functions complementing each other such as EncryptByCert() and DecryptByCert().

For more details on data encryption, refer to SQL Server Books Online.

High Availability Features

Oracle 10g includes Data Guard as a feature used in creating high availability solutions. Data Guard is the management, monitoring, and automation software infrastructure that creates, maintains, and monitors one or more standby databases to protect enterprise data from failures, disasters, errors and corruption.

SQL Server 2005 has Data Mirroring as a feature which can be related as the equivalent to the Data Guard solution. Database mirroring is a primarily software solution for protecting a SQL Server database from both system and site failures. Database mirroring is implemented on a per database basis.

Database mirroring works by maintaining a hot standby server locally or at a remote location. During a typical database mirroring session, after a production server fails, client applications can recover quickly by reconnecting to the standby server either automatically or manually. The figure below illustrates a typical database mirroring environment.

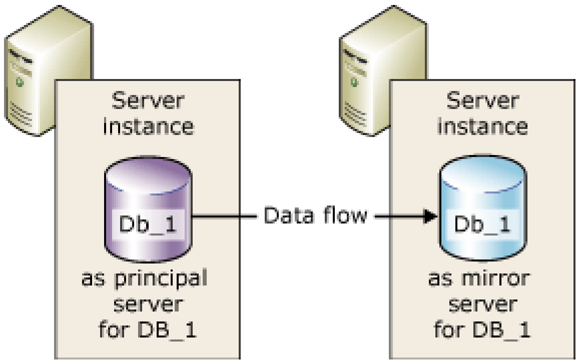

Figure 5: A typical database mirroring environment

Database mirroring involves two copies of a single database that typically reside on different computers. At any given time, only one copy of the database is currently available to clients. This copy is known as the principal database. Updates made by clients to the principal database are mirrored on the other copy of the database, known as the mirror database. Mirroring involves applying transactions from every insert, update, or delete transaction made on the principal database onto the mirror database.

The principal database and mirror database must reside on separate server instances (that is, instances of the Microsoft SQL Server database engine). The two server instances communicate and cooperate as partners in a database mirroring session. The two partners perform complementary roles in the session: the principal role and the mirror role. At any given time, one partner performs the principal role, and the other partner performs the mirror role. Each partner is described as owning its current role. The partner that owns the principal role is known as the principal server, and its copy of the database is the current principal database. The partner that owns the mirror role is known as the mirror server, and its copy of the database is the current mirror database.

The benefits of data mirroring include:

- Database mirroring provides complete or nearly complete redundancy of the data. For more information, see Database Mirroring Sessions in SQL Server Books Online

- Server instances communicate over a direct network connection and can be in separate locations. If necessary, they can be at considerable distances from each other to provide protection against major site disasters. No special hardware is required. This feature can also be implemented in conjunction with MSCS (Microsoft cluster service) in order to provide a disaster recovery solution over a hardware dependent Cluster solution.

- A database owner can use the database snapshots in the database on the mirror server to make the database available for limited reporting needs.

- Database mirroring can work as a non hardware dependent failover alternative over clustering.

An optional component called the Witness can be employed in a Database Mirroring setup to provide automatic failover. The purpose of the Witness is to monitor the status of the two servers in a mirrored pair and form a quorum with the surviving server in the event one server fails. This ensures that the system will not experience a “split-brain” condition.

Database Mirroring does not rely upon application code for client redirection in the case of a failover. The SNAC and ADO.NET data providers are fully database mirroring aware. The following figure illustrates a client connection to the initial partner, Partner_A, for a mirrored database named Db_1. This figure shows a case in which the initial partner name supplied by the client correctly identified the current principal server, Partner A. The initial connection attempt succeeded, and the data-access provider stores the name of the mirror server (currently Partner_B) as the failover partner name in its local cache. The following figure illustrates the manner in which the client name is stored in the provider. Should Partner_A fail, the client will attempt to re-connect to Partner_A to ensure the client’s inability to connect to Partner_A was not due to network congestion problems. If repeated attempts fail, the client will then attempt to connect to Partner_B and if successful, normal operations resume.

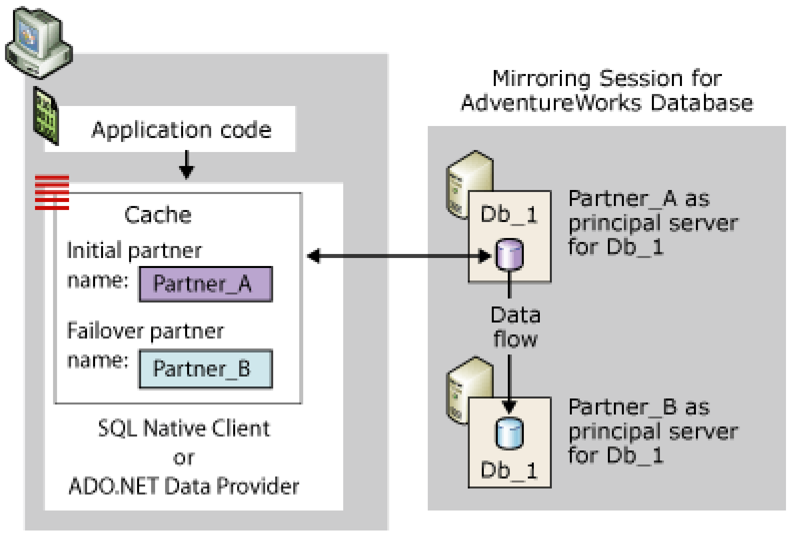
Application code
Cache
Initial partner name:
Partner_A
Failover partner name:
Partner_B
SQL Native Client or ADO.NET Data Provider
Mirroring Session for AdventureWorks Database
Db_1
Partner_A as principal server for Db_1
Data flow
Partner_B as principal server for Db_1
Db_1

Figure 6: Database mirroring

Database snapshots

A powerful feature combination is the use of database snapshots with database mirroring. Database snapshots are read-only databases providing an exact copy of a user database at a specific point in time. A source database can have many snapshots.

Database snapshots operate at the data-page level and do not have a 2X disk requirement. Before a page of the source database is modified for the first time, the original page is copied from the source database to the snapshot. This process is called a *copy-on-write* operation. The snapshot stores the original page, preserving the data records as they existed at the point in time when the snapshot was created. Subsequent updates to records in a modified page do not affect the contents of the snapshot. The same process is repeated for every page that is being modified for the first time. In this way, the snapshot preserves the original pages for all data records that have ever been modified since the snapshot was taken but does not take use up the same amount of disk space as the original database at the time of creation. Given the nature database snapshots and that they can be created and dropped in a matter of seconds, users typically create snapshots that are used for brief periods before dropping and creating a new database snapshot.

Apart from using database snapshots for reporting, it can also be used for rapid recovery from user errors. For example, a user that accidentally deleted all the data from a large table instead of deleting just a few rows, can have the table recovered rapidly using a valid database snapshot instead of performing a full restore from backup. Though it does not have the same level of flexibility and functionality, SQL Server 2005's Database Snapshots can provide some of the functionality found in Oracle's Flashback database feature.

Management Infrastructure and Tools

Building a large database is easy compared to managing the large database and ensuring it continues to deliver the performance, availability and security required. SQL Server has a robust set of features and tools to help DBAs and developers develop and manage both large complex applications and small simple applications. An important point to note is that both types of applications and everything in between can be developed and managed effectively from a single toolset and management framework.

SQL Server Management Studio

Oracle 10g uses the Enterprise Manager as the toolset to perform database administration tasks. The Enterprise Manager offers a unified view of the Oracle environment in the enterprise for database management, monitoring and tuning needs. The SQL Server Management Studio is the SQL Server equivalent to the Enterprise Manager in Oracle.

Completely revamped in Microsoft SQL Server 2005, the Microsoft SQL Server Management Studio is an integrated environment for accessing, configuring, managing, administering, and developing all components of SQL Server. SQL Server Management Studio combines a broad group of graphical tools with a number of rich script editors to provide access to SQL Server to administrators of all skill levels.

In addition to interfacing directly with the SQL Server relational engine, SQL Server Management Studio works with all components of SQL Server such as Analysis Services, Reporting Services, Integration Services, SQL Server Mobile, and Notification Services. DBAs use a single comprehensive utility that combines easy-to-use graphical tools with rich scripting capabilities.

The features of SQL Server Management Studio include:

- Supports most administrative tasks for SQL Server 2005 and SQL Server 2000 such as configuration of server parameters, backup and restore, and configuring replication.
- A single, integrated environment for SQL Server management and authoring.
- New management dialogs for managing objects in the SQL Server Database Engine, Analysis Services, Reporting Services, Notification Services, and SQL Server Mobile, that allows you to execute your actions immediately, send them to a code editor, or script them for later execution.
- Exporting and importing SQL Server Management Studio server registration from one management studio environment to another.
- Save or print XML Showplan or Deadlock files generated by SQL Server Profiler, review them later, or send them to administrators for analysis.
- New error and informational messages provide more information and allow you to send Microsoft a comment about the messages, or to copy messages to the clipboard, and allows you to easily e-mail the messages to your support team.
- A new activity monitor for collecting server process information which can be filtered and automatically refreshed at periodic intervals.
- There is also an integrated database mail interfaces for alert notifications.
- Code editor for T-SQL, MDX (multi-dimensional expressions), CLR (Common Language Runtime) and XML. The code editor is a very rich user interface for writing, debugging and tuning T-SQL, MDX and CLR scripts in a single interface. Some of the features of the code editors include
  - Color coding of syntax to improve the readability of complex statements.
  - A graphical query designer for drag and drop creation of queries.
  - Presentation of query windows as either tabs in the document window, or in separate documents.
  - Presentation of query results in either a grid or text window or redirected to a file.
  - Display of result grids as separate tabbed windows.

Dynamic Management views

Dynamic management views (DMV) and functions return server state information that can be used to monitor the health of a server instance, diagnose problems, and tune performance. The functionality is similar to Oracle V$ views though the coverage does vary.

There are two types of DMVs in SQL Server 2005; server scoped and database scoped. As the name implies, the information provided by the different views will be relevant to the scope it resides. All dynamic management views and functions exist in the sys schema and follow the naming convention dm_*.

When using a dynamic management view or function, you must prefix the name of the view or function by using the sys schema. For example, if you wanted to find out what are the current pending I/O requests in SQL Server, run the following query:

```
SELECT * FROM sys.dm_io_pending_io_requests
```

When used with Catalog view and Performance Monitor counters, DMVs can provide the same functionality as Oracle10g's Automatic Workload Repository (AWR). However, when working with SQL Server 2005, much of the intended purpose of Oracle's AWR is not needed because SQL Server automatically monitors and captures various statistics and dynamically re-configures various parameters to optimize performance. Oracle's AWR is intended to do the same but still requires explicit instruction to execute on configuration changes.

Monitoring and Tuning

SQL Server 2005 offers multiple features which can be used for performance monitoring and tuning of the database environment. The section below discusses these different tools and their features which can be used for monitoring and tuning the database.

SQL Server Profiler—SQL Server Profiler tracks engine process events, such as the start of a batch or a transaction, enabling you to monitor server and database activity (for example, deadlocks, fatal errors, or login activity). You can capture SQL Server Profiler data to a SQL Server table or a file for later analysis, and you can also replay the events captured on SQL Server step by step, to see exactly what happened. This tool is extremely useful to identify application impact on performance due to the activities performed by an application in the SQL Server Environment. This tool can be used by either the developer or administrator.

System Monitor—System monitor can be used to monitor the entire SQL Server subsystem in a local or a remote machine. System Monitor primarily tracks resource usage, such as the utilization of memory by SQL Server, enabling you to monitor server performance and activity using predefined objects and counters or user-defined counters to monitor events. System Monitor (Performance Monitor in Microsoft Windows NT 4.0) collects counts and rates rather than data about the events. You can set thresholds on specific counters to generate alerts that notify operators.

The key difference between SQL Server Profiler and System Monitor is that SQL Server Profiler monitors database engine events, whereas the System Monitor monitors resource usage associated with server processes.

SQL Server Management Studio (Activity Monitor) — the activity monitor in SQL Server 20005 helps in viewing a varied kind of information in an ad hoc manner. The activity monitor helps in viewing the processes running in an instance of SQL Server, the processes which are blocked, the locks in the system, and the current activity of users.

Data Movement and Tools

Data Export/Import tools

SQL Server 2005 also has a collection of data movement and loading tools. The BULK INSERT and bcp commands (similar to Oracle's SQL*Loader) have a long history of performing insert operations. DTS was often the tool of choice in SQL Server 2000 when transformations were required on the data despite is relatively basic set of features.

SQL Server 2005 now offers an exciting new tool called SQL Server Integration Services (SSIS). SSIS allows the integration of managed code in the data and control flow tasks to get the precise results needed for transformations and the ultimate destination of the data. It is a true ETL tool that efficiently gets data where it is needed by eliminating the need for multiple staging tables. SSIS uses a "pipeline" architecture and massages the data on its way from the source to its destination. The pipeline may also be a data source for Data Mining so that "real-time" data mining may be implemented.

With SSIS, users now have an industrial strength ETL tool bundled with SQL Server at no additional cost. Oracle does have an ETL tool but it is part of the Oracle Warehouse Builder which is sold as part of the Internet Development Suite.

Moving Databases

Similar to Oracle's transportable tablespaces feature, SQL Server provides a simple way to move databases from one instance to another. The move can be a logical and/or physical move and the difference in execution is minimal and the operations can be performed via SQL Server Management Studio or command line using T-SQL. To move a SQL Server database from one instance to another, you only need to detach the database from one instance and attach it to another. If physical movement is necessary, simply move the datafiles and log files associated with that database to the desired new location before attaching to the new instance.

Moving to different platforms is irrelevant in SQL Server since it only runs on the Windows platform. However, the process to move from one edition or version of Windows to another is done the same way as moving within the same edition or version. Do note that different versions/editions of Windows have different levels of functionality so please be aware of your database's requirements prior to moving.

Data Replication

Replication is a set of technologies for copying and distributing data and database objects from one database to another and then synchronizing between databases to maintain consistency. Using replication, you can distribute data to different locations and to remote or mobile users over local and wide area networks, dial-up connections, wireless connections, and the Internet. SQL Server 2005 supports all popular types of replication including transactional, peer-to-peer transactional (aka multi-master), merge and snapshot. SQL Server also supports replication with non SQL Server databases including Oracle.

Implementation methods and requirements will differ depending on the type of replication needed but for all practical purposes, these models deliver on most of the combined capabilities provided by Oracle replication and

Oracle Streams. Setting up, monitoring and managing replication can be done through SQL Server Management Studio or through T-SQL and some replication system stored procedures.

Transaction/Isolation Levels

In SQL Server 2005, the isolation level is the type of locking that is applied when working with sets of data and how other readers are isolated (or not) from data that is currently being read/modified. SQL Server has always supported the four ANSI standard isolation models – read uncommitted, read committed (default), repeatable read and serializeable but in SQL Server 2005, snapshot isolation support is introduced. Snapshot Isolation uses row versioning to provide transaction-level read consistency and only schema locks are acquired. This allows systems with mixed workloads to function effectively with large number users without having readers block writers and vice-versa. Row versioning is implemented in TEMPDB and the rows are persisted for as long as there are transactions that require the information. This is different from Oracle's implementation which uses the undo tablespace to implement the same functionality. SQL Server users will not encounter an equivalent of Oracle's "Snapshot Too Old" error.

Conclusion

Microsoft SQL Server 2005 is a comprehensive, integrated end-to-end data management solution that empowers users across the organization by providing them with a secure, reliable, and productive platform for enterprise data and business intelligence (BI) applications. SQL Server 2005 delivers powerful and familiar tools to information technology professionals and information workers.

SQL Server 2005 is designed with goals to increase business value through reduced TCO, ease of use and extensive integration capabilities for information management. SQL Server 2005 also offers various services and infrastructure to deploy Business Intelligence systems with significantly reduced development and integration effort. Further capabilities offered by features like Service Broker and Query Notification allows today's IT Professionals to design and deploy both classic client/server and web applications, and more advanced web services based applications with service oriented architectures.

Additional Resources

- For more information on Microsoft SQL Server 2005, visit www.microsoft.com/sql/2005
- For more information on migration tools and assistance, visit http://www.microsoft.com/sql/migration/default.mspx
- For a demo of SQL Server's High Availability features, visit http://www.microsoft.com/sql/2005/productinfo/demos/hademo.mspx
- For SQL Server hands-on lab materials, visit http://msdn.microsoft.com/sql/2005/2005labs/default.aspx
- For SQL Server e-Learning material, visit https://www.microsoftelearning.com/sqlserver2005/