Transaction Management And Concurrency Control

As more networks and databases are connected together, the importance of a solid database management system becomes apparent. Transaction and Concurrency Control, Recovery and Backup, and Security are major functions that should be examined when choosing the correct system. Databases which contain your company's valuable information must be properly protected, backed up, and secure from data loss and unauthorized access.

In response to this requirement, both Oracle and Microsoft have implemented strong features into their database products. This paper compares the offerings of the two databases in terms of features, functionality, and ease of management.

Table of Contents

Security

a) Overview

b) SQL Server Security

c) Oracle Security

d) Comparison

Conclusion

Introduction

This paper will provide a comparative review of three database management system functions: transaction and concurrency control, recovery and backup, and security, between Microsoft SQL Server and Oracle. The purpose is to enhance understanding of database functionality and, through comparison, provide insight into the commonalities and differences between two different systems.

Overview of Database Management Systems

Microsoft SQL Server is a relational database server, with its primary languages being T-SQL and ANSI SQL. ANSI SQL is the American National Standards Institute standardized SQL and is used as the base for several different SQL languages, including T-SQL. T-SQL is a proprietary extension that uses keywords for the various operations that can be performed, such as creating and altering database schemas, entering and editing data, and managing and monitoring the server. Any application that works through SQL Server will communicate via T-SQL statements. T-SQL has some differences/extensions to basic SQL, including local variables, control of flow language, changes to delete and update statements, and support functions for date and string processing, and mathematics.

Version 1.0 of SQL Server was released in 1989 and originated in Sybase SQL Server. Microsoft later ended the co-licensing agreement with Sybase and went on to develop their own version of SQL Server. The latest version is SQL Server 2008, released on August 6, 2008, and includes many improvements to speed and functionality, which will be discussed in further detail below.

Sample SQL Server Architecture Diagram

1

Oracle Database is a relational database management system produced by Oracle Corporation. Users can utilize the proprietary language extension to SQL, PL/SQL, or

the object-oriented language Java to store and execute functions and stored procedures.

Oracle V2 was first released in November 1979 and did not support transactions, but had basic query and join functionality. The latest version is Oracle Database 11g, released in 2007, and includes many enhancements to functionality, which will be discussed in further detail below.

Sample Oracle 11g Architecture Diagram

2

Transaction Management and Concurrency Control

Overview

A transaction, a single logical unit of work, is an action or series of actions that are performed by a user or application which can access or change the database contents. A transaction results in database transformation from one consistent state to another, and can either result in success or failure. A failed transaction is aborted and the database restores to the previous consistent state. The Database Management System is responsible for making sure all updates related to the transaction are carried out, or that stability is maintained in the case of a failed transaction. Transactions have four basic properties: Atomicity, Consistency, Independence, and Durability (ACID). Atomicity means that it is a single unit of work. Consistency ensures that data is always held firmly together in a coherent state, even after a failed transaction or crash. Independence ensures that the effects of an incomplete transaction are contained and not visible to other transactions. Durability ensures that successful transactions result in permanent changes to the state of the database.

Concurrency control is the process of managing and controlling simultaneous database operations. This is required because actions from different users and operations must not interfere with functionality, or the database could be left in an inconsistent state. Potential problems that concurrency control can solve are lost updates, inconsistent analysis, and uncommitted dependencies. The two main concurrency control techniques are locking and timestamping.

SQL Server TM and CC

SQL Server fulfills the ACID requirements by using transaction management, locking, and logging. An explicit transaction is created in SQL Server by using the BEGIN TRANSACTION and COMMIT TRANSACTION commands. ROLLBACK TRANSACTION rolls back a transaction to the beginning or another save point within the transaction. SAVE TRANSACTION sets a savepoint within the transaction by

dividing the transaction into logical units that can be returned to if part of the transaction is conditionally cancelled.

Locking ensures transactional integrity and database consistency. In SQL Server, locking is automatically implemented, and provides both optimistic and pessimistic concurrency controls. Optimistic concurrency control assumes that resource conflicts are unlikely but not impossible, and allows transactions to execute without locking resources. Pessimistic concurrency control locks resources for the duration of a transaction. SQL Server can lock the following resources: RIDs, keys, pages, extents, tables, and databases. It utilizes several lock modes, including shared, update, exclusive, intent, and schema locks. Shared locks allow for concurrent read operations that do not change or update data, such as a SELECT statement. Update locks prevent a common form of deadlock that occurs when multiple sessions are reading, locking, and potentially updating resources later. Exclusive locks are used for data modification operations, such as INSERT, UPDATE, or DELETE, and ensure that multiple updates can't be made on the same resource at the same time. Intent locks are used to establish a lock hierarchy, and include intent shared, intent exclusive, and shared with intent exclusive locks. Schema locks are used when a schema dependent operation of a table is executed, and include schema modification and schema stability locks.

A deadlock occurs when two transactions have locks on separate objects and each user is waiting for a lock on the other object. SQL Server can set deadlock priority by scanning for sessions that are waiting for a lock request, and the SET DEADLOCK_PRIORITY command to customize deadlocking. The SETLOCK_TIMEOUT command can set the maximum time that a statement waits on a blocked resource, because the timeout period is not enforced by default.

Oracle TM and CC

Oracle Database offers two isolation levels, providing developers with operational modes that preserve consistency and provide high performance. Statement level read consistency automatically provides read consistency to a query so that all the data the query sees comes from a single point in time when the query began. The query never sees any dirty data or changes made during query execution. Transaction level read consistency extends read consistency to all queries in a transaction. Oracle uses rollback segments, containing old values of data that have been changed by uncommitted or recently committed transactions, to provide consistent views and does not expose a query to phantoms.

Oracle Real Application Clusters (RACs) use cache-to-cache block transfer to transfer read-consistent images of blocks between instances. It uses high speed, low latency interconnects to answer remote data block requests.

Isolation levels provided by Oracle Database are read committed, serializable, and read-only. Users can choose the appropriate isolation levels for transactions depending

on the type of application and workload, using these statements: SET TRANSACTION ISOLATION LEVEL READ COMMITTED; SET TRANSACTION ISOLATION LEVEL SERIALIZABLE; and SET TRANSACTION READ ONLY. The ALTER SESSION function can then be used to change isolation level for different transactions.

Read committed is the default transaction isolation level. Each query executed by a transaction sees data committed before the query began. Oracle Database does not prevent other transactions from modifying the data read by a query, so that data can be changed by other transactions between two query executions. This can lead to non-repeatable reads and phantoms in cases where the transaction runs the same query twice. This isolation level is good for when few transactions are likely to conflict, and can provide higher potential throughput.

Serializable transactions see only changes made at the beginning of the transaction, plus changes in the transaction itself through INSERT, UPDATE, and DELETE statements. These transactions do not experience non-repeatable reads or phantoms. This isolation level is suitable for large databases and short transactions that update few rows, when there is a low chance that two concurrent transactions will modify the same rows, or where long-running transactions are primarily read-only. A serializable transaction can modify a data row only if it can determine that prior changes were committed before the current transaction began. Oracle Database uses control information in the data block to indicate which rows have committed and uncommitted changes. The amount of history that is retained is determined by the INITRANS parameter of CREATE and ALTER TABLE. To avoid having insufficient recent history information, higher values can be set for INITRANS for tables that will have many transactions updating the same blocks. If a serializable transaction fails with the CANNOT SERIALIZE ACCESS error, the application can either commit the work executed to that point, execute additional statements with ROLLBACK, or undo the entire transaction.

Read-only transactions see only changes made at the time the transaction began and don’t allow INSERT, UPDATE, or DELETE statements.

Oracle Database uses locks to control simultaneous access to data resources. Low-level serialization mechanisms called latches are used to protect shared data structures in the System Global Area. Oracle automatically gets the necessary locks when executing SQL statements, using the lowest applicable level of restrictiveness to provide the highest possible data concurrency and data integrity. The user may also lock data manually. There are two modes of locking: exclusive and share lock modes. Exclusive lock mode prevents the associated resource from being shared, and is obtained to modify data. The first transaction to lock the data is the only one which can modify it until the lock is released. Share lock mode allows the associated resource to be shared, depending on the operations. Users reading data can hold share locks to prevent a writer access. Multiple transactions can have share locks on the same

resource. All locks created by statements within a transaction last until the transaction is completed or undone.

Because row locks are acquired at the highest degree of restrictiveness, no lock conversion is needed or performed. Oracle automatically converts table lock restrictiveness from lower to higher as appropriate. Lock escalation is when multiple locks are held at one level of granularity, and a database raises the locks to a higher level of granularity. An example is converting many row locks into one table lock. Oracle Database never escalates locks, because this increases the chances of deadlocks. A deadlock occurs when two or more users are waiting on data locked by each other. This can prevent transactions from continuing to work. Oracle automatically detects deadlocks and solves them by rolling back one of the statements. User generated deadlocks can be avoided by locking tables in the same order for transactions accessing the same data.

Oracle Database locks fall into three general categories: DML locks (data locks), DDL locks (dictionary locks), and Internal locks and latches.

DML locks protect data (i.e. tables, rows). The purpose is to guarantee the integrity of data accessed by multiple users. Row locking is the finest granularity and has the best possible concurrency and throughput. A transaction always acquires an exclusive row lock for each individual row modified by INSERT, UPDATE, DELETE, and SELECT with the FOR UPDATE clause. If a transaction uses a row lock, it also uses a table lock for the corresponding table. Table locking is mainly used for concurrency control with DDL operations. Table locks are used when a table is modified by the INSERT, UPDATE, DELETE, SELECT with FOR UPDATE, and LOCK TABLE DML statements. These statements require table locks to reserve DML access to the table for the transaction and to prevent conflicting DDL operations. Table locks can be used at both table and subpartition level for partitioned tables. A table lock can be held in the following modes, from least to most restrictive: row share (RS), row exclusive (RX), share (S), share row exclusive (SRX), and exclusive (X).

A row share table lock is the least restrictive, and has the highest degree of concurrency for a table. It indicates the transaction has locked rows in the table and intends to update them. It is specified by the statement LOCK TABLE “” IN ROW SHARE MODE. A row exclusive table lock is slightly more restrictive, and indicates the transaction holding the lock has made one or more updates to rows in the table or issued a SELECT FOR UPDATE statement. It is specified by LOCK TABLE “” IN ROW EXCLUSIVE MODE;. A share table lock is made automatically for a table specified by the statement LOCK TABLE “” IN SHARE MODE;. A share row exclusive lock is more restrictive and is made for a table specified by the statement LOCK TABLE “” IN SHARE ROW EXCLUSIVE MODE;. Exclusive table locks are the most restrictive and are specified by the statement LOCK TABLE “” IN EXCLUSIVE MODE;.

DDL locks protect the structure of schema objects (i.e. table definitions). Internal locks and latches are automatic and protect internal data structures such as data files. Only individual schema objects that are modified or referenced are locked during DDL operations. The entire data dictionary is never locked. DDL locks have three categories: exclusive DDL locks, share DDL locks, and breakable parse locks. Exclusive and share DDL locks last until DDL statement execution and automatic commit is complete.

Most DDL operations require exclusive DDL locks for a resource to prevent interference with other DDL operations that might reference the same object. If another DDL lock is already held, then the operation must wait until the other lock is released to proceed. DDL operations also create DML locks on the modified schema object.

Some DDL operations require share DDL locks to allow data concurrency for similar DDL operations. A share DDL lock is created for the following statements: AUDIT, NOAUDIT, COMMENT, CREATE (OR REPLACE) VIEW/ PROCEDURE/ PACKAGE/ PACKAGE BODY/ FUNCTION/ TRIGGER, CREATE SYNONYM, and CREATE TABLE (if CLUSTER is not used).

Breakable parse locks are acquired is created for a SQL statement and each schema object it references. A parse lock does not restrict any DDL operation and can be broken to allow conflicting DDL operations. It is created in the parse phase of SQL statement execution and held as long as the shared SQL area for the statement is in the shared pool.

Latches and internal locks protect internal database and memory structures. Users cannot access them. Latches are simple, low-level serialization mechanisms to protect shared data structures in the system global area. The use of latches is dependent on the operating system. Internal locks are higher-level, more complex mechanisms and include dictionary cache locks, file and log management locks, and tablespace and rollback segment locks. Dictionary cache locks are very short and are on dictionary caches while the entries are being modified or used. They make sure that parsed statements don’t have inconsistent object definitions. They can be shared or exclusive; shared last until the parse is finished and exclusive last until the DDL operation is finished.

File and log management locks protect different files. They are held for a long time because they indicate the status of files.

Tablespace and rollback segment files protect tablespaces and rollback segments. All instances must agree whether a tablespace is online or offline. Rollback segments are locked to make sure that only one instance can write to a segment.

Comparison

Microsoft SQL Server is enabled to lock smaller amounts of data at a time, which is a big improvement. There is row-level locking, so now SQL Server locks only the rows that are actually being changed. However, SQL Server has no multi-version consistency model, which means that reads and writes can block each other to ensure data integrity. The difference with Oracle is that the database maintains a snapshot of the data, which prevents queries from hanging without performing "dirty reads."

Backup and Recovery

Overview

Database backup and recovery mechanisms ensure that organizations have prepared a copy of their data, or have the tools necessary to recover from a failure. A failure is a state where inconsistency prevents transactions from reaching the desired results. Some types of failures are transaction failure, system failure, media failure, and communications failure. Transaction failure may be caused by deadlocks, time-outs, protection violations, or system errors. Transaction failures can be solved with either a partial or total rollback, depending on the extent of the failure. System failures can be recovered with a restart, or rollback to the last consistent state. Restore/roll forward functions help with restoring the database after a media failure.

SQL Server B and R

SQL Server databases consist of two physical hard drive files, the MDF and LDF files. MDF files contain all of the data being stored. LDF files contain a record of every data change. Logging data changes make undo operations and backups possible. The log file is cleared, or truncated, after a certain amount of time, which is determined by the database recovery model. SQL Server can maintain multiple databases, with different recovery model settings. The recovery model can be either simple, full, or bulk-logged.

With simple recovery, log files are not kept permanently, so when this setting is activated, a full backup must be done. Full backups restore all of the data and cannot be set to a specific time.

The full recovery setting refers to a database with a transaction log file history. The log files keep track of every data change operation. The database will stop working if the log file runs out of space, so the auto grow function can be enabled.

When running in full recovery, differential and transaction log backups become available. Differential backups copy all data changes since the last full backup. Every time a full backup is run, the differential backup is reset. Transaction log backups copy all data changes since the last full or transaction log backup. They are usually very small and fast. The disadvantage is the level of recovery; if any log backup is damaged or unusable, the data is not recoverable past the last good backup.

Oracle B and R

Oracle databases can be backed up using export/import, cold or off-line backups, hot or on-line backups, or RMAN backups. Exports extract logical definitions and data from the database to a file. Cold or off-line backups shut down the database and backup all data, log, and control files. Hot or on-line backups set the tablespaces into backup mode and backup the files. The control files and archived redo log files must also be backed up. RMAN backups use the “rman” utility to backup the database. More than one of these methods can and should be used and tested to make sure the database is securely backed up.

On-line backups can only be done when the system is open and the database is in ARCHIVELOG mode. Off-line backups are performed when the system is off-line; the database doesn’t have to be in ARCHIVELOG mode. It is easier to restore from off-line backups because no recovery is required, but on-line backups are not as disruptive and don’t require database downtime. Point-in-time recovery is available in ARCHIVELOG mode only.

Comparison

Starting with version 10g, Oracle Database adopted the Automatic Storage Management (ASM) feature, which automates storage management after a certain point. The DBA allocates storage devices to a database instance and it automates the placement and storage of the files. SQL Server storage management must be done manually, using the Share and Store Management Console in SQL Server 2008, or must purchase a separate tool. Oracle’s Flash Recovery feature automates the management of all backup files. The Flash Recovery area is a unified storage location for all recovery related files in the Oracle database. The DBA can also change the storage configuration without having to take the database offline. SQL Server also provides the ability to manage backup files, using a backup wizard to manage the relevant files, but does not do it automatically. SQL Server 2008 introduced improvements in backup compression. With compression, less disk I/O and storage is required to keep backups online, resulting in increased speed. Tradeoffs seem to be between SQL Server’s speed and Oracle’s increased functionality.

In Oracle, backups are fully self-contained, but in SQL Server the DBA must manually recreate the system database using the install CD. Oracle also uses the Data Recovery Advisor (DRA) tool to automatically diagnose data failures, show repair options, and execute repairs at the user’s request. Oracle’s Flashback technology allows for instant recovery of dropped tables and logical data corruptions. SQL Server provides for data recovery by rebuilding the transaction log, running repair to fix any corruptions, and ensure the logical integrity of data is not broken.

Security

Overview

Security is an important part of any organization's database management system. According to Dr. Osei-Bryson's lecture notes, security breaches are typically categorized as unauthorized data observation, incorrect data modification, or data unavailability. Unauthorized data observation discloses confidential information to users without the proper permissions. Incorrect data modification can be either intentional or unintentional, but can be devastating to database consistency and can result in unreliable data. Unavailable data can be very costly to an organization, depending on how the data is used.

Three requirements for a data security plan include secrecy and confidentiality, integrity, and availability. Secrecy and confidentiality protects data from being accessed by unauthorized parties. Database integrity is important to protect the data from incorrect or improper modification. Availability means preventing and minimizing the damage from unavailable data.

Database management systems include some form of access control mechanism to make sure each user has access to only the data they require to perform their jobs. Users are granted certain authorizations by a security administrator to determine which actions can be performed on each object. The database administrator is responsible for account creation, assigning security levels, and granting/revoking privileges.

SQL Server Security

Security is an integral part of SQL Server's package, according to a recent White Paper commissioned by Microsoft. Security features for Microsoft SQL Server 2008 include policy-based management to apply policies to database objects. These policies contain a collection of conditions that can be used to enforce business and security rules.

Oracle Security

Oracle 11g uses supports strong authentication through KPI, Kerberos, and Radius for all connections to the database except connections made as SYSDBA or SYSOPER. Tablespace encryption provides an alternative to transparent data encryption column encryption by enabling the encryption of the entire tablespace. This is best used with large amounts of data. The transparent data encryption master key can be stored in an external hardware security module for stronger security. 11g also provides increased password protection, secure file permissions, optional default audit settings, and controls on the network callouts from the database.

Comparison

In SQL Server, transparent data encryption encrypts and decrypts data in the database engine and doesn't require more application programming. The functionality is included

in SQL Server 2008, but requires a $10,000 per processor additional charge with Oracle Database 11g. SQL Server 2008 allows Extensible Key Management and Hardware Security Module vendors to register in SQL Server and provide management that is separated from the database. This separation of keys from the data provides an additional layer of defense. SQL Server 2008 also has auditing support through an Auditing object, which allows administrators to capture and log all database server activity.

The National Vulnerability Database, provided by the National Institute of Science and Technology, reported over 250 security vulnerabilities with Oracle products over a four year period, and none with SQL Server. The report did not list the type and severity of the vulnerabilities, or which specific products were affected, but there seems to be a trend toward vulnerability.

Microsoft Update is a fairly straightforward and easy to use patching solution for SQL Server. Computerworld called Oracle's patch management system involved "excruciating pain" and "two-thirds of Oracle DBAs don't apply security patches." Oracle seems to be behind in patch management at this time.

SQL Server can also prevent highly privileged users from accessing sensitive data through use of the auditing object, assigning individual permissions, module signing, Policy-based management, and additional functionality. Oracle uses Database Vault to control privileged access, but costs 20k per processor.

Conclusion

The comparative review of Transaction Management and Concurrency, Recovery and Backup, and Security functions on Microsoft SQL Server and Oracle 11g database has shown that there are many similarities in the functionality between the two companies, but also key differences in database management philosophy. I learned that SQL Server seems to have the edge on speed and better security, but Oracle is making many advances in high level functionality and is starting to automate many features than in previous years. I was also able to improve my understanding of the DBMS functions by examining their practical application in separate systems.