**St. Xavier’s College**

****

**Database Management System**

**Lab assignment 10 #2**

**Submitted to:**

**..............................**

**Er. Sanjay Yadav**

**Department of Computer Science**

**Submitted by:**

**Pankaj Niroula**

**BSCCSIT 4th Sem**

**Roll number:25**

**Transaction Management**

**Introduction**

A transaction in Oracle begins when the first executable SQL statement is encountered. An executable SQL statement is a SQL statement that generates calls to an instance, including DML and DDL statements.

When a transaction begins, Oracle assigns the transaction to an available undo tablespace to record the rollback entries for the new transaction.

A transaction ends when any of the following occurs:

* A user issues a COMMIT or ROLLBACK statement without a SAVEPOINT clause.
* A user runs a DDL statement such as CREATE, DROP, RENAME, or ALTER. If the current transaction contains any DML statements, Oracle first commits the transaction, and then runs and commits the DDL statement as a new, single statement transaction.
* A user disconnects from Oracle. The current transaction is committed.
* A user process terminates abnormally. The current transaction is rolled back.

After one transaction ends, the next executable SQL statement automatically starts the following transaction.

**Transaction**

A transaction is a logical unit of work that contains one or more SQL statements. A transaction is an atomic unit. The effects of all the SQL statements in a transaction can be either all committed (applied to the database) or all rolled back (undone from the database).

A transaction begins with the first executable SQL statement. A transaction ends when it is committed or rolled back, either explicitly with a COMMIT or ROLLBACK statement or implicitly when a DDL statement is issued.

To illustrate the concept of a transaction, consider a banking database. When a bank customer transfers money from a savings account to a checking account, the transaction can consist of three separate operations:

* Decrement the savings account
* Increment the checking account
* Record the transaction in the transaction journal

Oracle must allow for two situations. If all three SQL statements can be performed to maintain the accounts in proper balance, the effects of the transaction can be applied to the database. However, if a problem such as insufficient funds, invalid account number, or a hardware failure prevents one or two of the statements in the transaction from completing, the entire transaction must be rolled back so that the balance of all accounts is correct.

Figure below illustrates the banking transaction example.

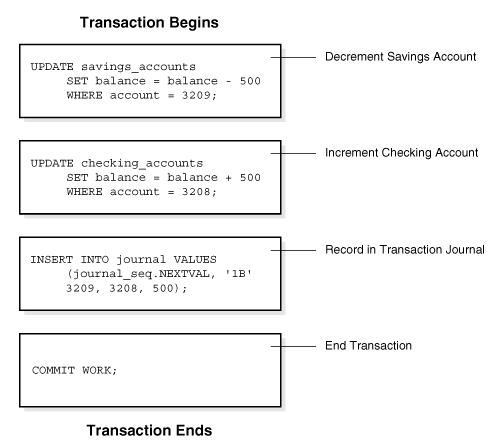


Figure 1 A Banking Transaction

**Transaction Recovery**

Every Microsoft® SQL Server™ 2000 database has a transaction log that records data modifications made in the database. The log records the start and end of every transaction and associates each modification with a transaction. An instance of SQL Server stores enough information in the log to either redo (roll forward) or undo (roll back) the data modifications that make up a transaction. Each record in the log is identified by a unique log sequence number (LSN). All of the log records for a transaction are chained together.

An instance of SQL Server records many different types of information in the transaction log. Instances of SQL Server 2000 primarily log the logical operations performed. The operation is reapplied to roll forward a modification, and the opposite of the logical operation is performed to roll back a modification.

Each instance of SQL Server controls when modifications are written from its data buffers to disk. An instance of SQL Server may cache modifications in buffers for a period of time to optimize disk writes. A buffer page that contains modifications that have not yet written to disk is known as a dirty page. Writing a dirty buffer page to disk is called flushing the page. When modifications are cached, care must be taken to ensure that no data modification is flushed before the corresponding log image is written to the log file. This could create a modification that could not be rolled back if necessary. To ensure that they can recover all modifications, instances of SQL Server use a write-ahead log, which means that all log images are written to disk before the corresponding data modification.

A commit operation forces all log records for a transaction to the log file so that the transaction is fully recoverable even if the server is shut down. A commit operation does not have to force all the modified data pages to disk as long as all the log records are flushed to disk. A system recovery can roll the transaction forward or backward using only the log records.

Periodically, each instance of SQL Server ensures that all dirty log and data pages are flushed. This is called a checkpoint. Checkpoints reduce the time and resources needed to recover when an instance of SQL Server is restarted. For more information on checkpoint processing, see Checkpoints and the Active Portion of the Log.

System Recovery

Media Recovery

Media recovery restores datafiles to a point in time before failure. The following is a list of media recovery options:

**Complete Media Recovery**

Complete media recovery is the recovery of all damaged or missing database files and the application of all redo information. The database may need to be opened with the RESETLOGS option if a backup control file or new control file was created for the recovery. There are three types of complete media recoveries.

* Closed database recovery
* Open-database, offline tablespace recovery
* Open database, offline tablespace datafile recovery

**Incomplete Media Recovery**

Incomplete media recovery is also called point-in-time recovery. Point in time recovery that is not continued to a complete recovery must be terminated by the OPEN RESETLOGS option. The database must be closed during incomplete recovery operations. There are three types of incomplete media recoveries.

* Cancel based recovery
* Time based recovery
* Change based recovery

**Two Phase Commit**

Unlike a transaction on a local database, a distributed transaction involves altering data on multiple databases. Consequently, distributed transaction processing is more complicated, because the database must coordinate the committing or rolling back of the changes in a transaction as a self-contained unit. In other words, the entire transaction commits, or the entire transaction rolls back.

The database ensures the integrity of data in a distributed transaction using the **two-phase commit mechanism**. In the **prepare phase**, the initiating node in the transaction asks the other participating nodes to promise to commit or roll back the transaction. During the **commit phase**, the initiating node asks all participating nodes to commit the transaction. If this outcome is not possible, then all nodes are asked to roll back.

All participating nodes in a distributed transaction should perform the same action: they should either all commit or all perform a rollback of the transaction. The database automatically controls and monitors the commit or rollback of a distributed transaction and maintains the integrity of the **global database** (the collection of databases participating in the transaction) using the two-phase commit mechanism. This mechanism is completely transparent, requiring no programming on the part of the user or application developer.

The commit mechanism has the following distinct phases, which the database performs automatically whenever a user commits a distributed transaction:

* Prepare Phase
* Commit Phase
* Forget Phase

**SQL Facilitites**