Transaction management

Introduction to Transactions

- 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.

Statement Execution and Transaction Control

- A SQL statement that executes successfully is different from a committed transaction.
- Executing successfully means that a single statement was:
  - Parsed
  - Found to be a valid SQL construction
  - Executed without error as an atomic unit.
- However, until the transaction that contains the statement is committed, the transaction can be rolled back, and all of the changes of the statement can be undone.

Statement-Level Rollback

- If at any time during execution a SQL statement causes an error, all effects of the statement are rolled back.
- The effect of the rollback is as if that statement had never been executed. This operation is a **statement-level rollback**.
- Errors discovered during SQL statement **execution** cause statement-level rollbacks.
- Single SQL statements involved in a **deadlock** (competition for the same data) can also cause a statement-level rollback.
- A SQL statement that fails causes the loss only of any work it would have performed itself. *It does not cause the loss of any work that preceded it in the current transaction.*

Resumable Space Allocation

- Oracle9*i* provides a means for suspending, and later resuming, the execution of large database operations in the event of space allocation failures.
- This enables an administrator to take corrective action, instead of the Oracle database server returning an error to the user.
- After the error condition is corrected, the suspended operation automatically resumes. This feature is called **resumable space allocation** and the statements that are affected are called **resumable statements**.
- A statement executes in a resumable mode only when the client explicitly enables resumable semantics for the session using the `ALTER SESSION` statement.

Resumable Space Allocation

- A resumable statement is suspended when one of the following conditions occur:
  - Out of space condition
  - Maximum extents reached condition
  - Space quota exceeded condition
- For a nonresumable statement, these conditions result in errors and the statement is rolled back.
- Suspending a statement automatically results in suspending the transaction. ***Thus all transactional resources are held through a statement suspend and resume.***
- When the error condition disappears (for example, as a result of user intervention or perhaps sort space released by other queries), the suspended statement automatically resumes execution.

Oracle and Transaction Management

- A transaction in Oracle begins when the first executable SQL statement is encountered.
- When a transaction begins, Oracle assigns the transaction to an available rollback segment to record the rollback entries for the new transaction.
  - A transaction ends when any of the following occurs:
  - You issue a `COMMIT` or `ROLLBACK` statement without a `SAVEPOINT` clause.
  - You execute 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 executes 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.

Commit Transactions

- **Committing** a transaction means making permanent the changes performed by the SQL statements within the transaction.
- Before a transaction that modifies data is committed, the following has occurred:
  - Oracle has generated rollback segment records in rollback segment buffers of the system global area (SGA). The rollback information contains the old data values changed by the SQL statements of the transaction.
  - Oracle has generated redo log entries in the redo log buffer of the SGA. The redo log record contains the change to the data block and the change to the rollback block. These changes may go to disk before a transaction is committed.
  - The changes have been made to the database buffers of the SGA. ***These changes may go to disk before a transaction is committed.***
- The data changes for a committed transaction, stored in the database buffers of the SGA, are not necessarily written immediately to the datafiles by the database writer (DBW*n*) background process. This writing takes place when it is most efficient for the database to do so.

Commit Transactions

- **Committing** means that a user has explicitly or implicitly requested that the changes in the transaction be made permanent.
- An explicit request means that the user issued a `COMMIT` statement.
- An implicit request can be made through normal termination of an application or in data definition language,
- When a transaction is committed, the following occurs:
  - The internal transaction table for the associated rollback segment records that the transaction has committed, and the corresponding unique system change number (SCN) of the transaction is assigned and recorded in the table.
  - The log writer process (LGWR) writes redo log entries in the SGA's redo log buffers to the online redo log file. It also writes the transaction's SCN to the online redo log file.
  - Oracle releases locks held on rows and tables.
  - Oracle marks the transaction complete.

Rollback of Transactions

- **Rolling back** means undoing any changes to data that have been performed by SQL statements within an uncommitted transaction.
- Oracle uses rollback segments to store old values.
- The redo log contains a record of changes.
- Oracle allows you to roll back an entire uncommitted transaction.
- Alternatively, you can roll back the trailing portion of an uncommitted transaction to a marker called a savepoint.
- In rolling back **an entire transaction**, without referencing any savepoints, the following occurs:
  - Oracle undoes all changes made by all the SQL statements in the transaction by using the corresponding rollback segments.
  - Oracle releases all the transaction's locks of data.
  - The transaction ends.

Savepoints In Transactions

- You can declare intermediate markers called **savepoints** within the context of a transaction.
- Savepoints divide a long transaction into smaller parts.
- Using savepoints, you can arbitrarily mark your work at any point within a long transaction. You then have the option later of rolling back work performed before the current point in the transaction but after a declared savepoint within the transaction.
- After a rollback to a savepoint, Oracle releases the data locks obtained by rolled back statements.

Savepoints In Transactions

- When a transaction is rolled back to a savepoint, the following occurs:
  - Oracle rolls back only the statements executed after the savepoint.
  - Oracle preserves the specified savepoint, but all savepoints that were established after the specified one are lost.
  - Oracle releases all table and row locks acquired since that savepoint but retains all data locks acquired previous to the savepoint.
- The transaction remains active and can be continued.

Transaction Naming

- Oracle9*i*, Release 1 (9.0.1), lets you name a transaction, using a simple and memorable text string.
- Advantages:
  - It is easier to monitor long-running transactions and to resolve in-doubt distributed transactions.
  - Log Miner can use transaction names to search for a specific transaction from transaction auditing records in the redo log.
  - You can use transaction names to find a specific transaction in data dictionary tables, such as `V$TRANSACTION.`
- **How Transactions Are Named**
  - You name a transaction by using the `SET TRANSACTION ... NAME` command before you start the transaction.
  - When you name a transaction, you associate the transaction's name with its ID.
  - Transaction names do not have to be unique;

Discrete Transaction Management

- *Application developers can improve the performance of short, nondistributed transactions by using the* ***`BEGIN_DISCRETE_TRANSACTION`*** *procedure.*
- *This procedure streamlines transaction processing so that short transactions can execute more rapidly.*
- *During a discrete transaction, all changes made to any data are deferred until the transaction commits.*
- *The following events occur during a discrete transaction:*
  - *Oracle generates redo information, but stores it in a separate location in memory.*
  - *When the transaction issues a commit request, Oracle writes the redo information to the redo log file along with other group commits.*
  - *Oracle applies the changes to the database block directly to the block.*
  - *Oracle returns control to the application after the commit completes.*

Autonomous Transactions

- Autonomous transactions are independent transactions that can be called from within another transaction.
- An autonomous transaction lets you leave the context of the calling transaction, perform some SQL operations, commit or roll back those operations, and then return to the calling transaction's context and continue with that transaction.
- Once invoked, an autonomous transaction is totally independent of the main transaction that called it.
- It does not see any of the uncommitted changes made by the main transaction and does not share any locks or resources with the main transaction.
- One autonomous transaction can call another.

Autonomous PL/SQL Blocks

- Use the pragma `AUTONOMOUS_TRANSACTION.` A **pragma** is a compiler directive.
- When an autonomous PL/SQL block is entered, the transaction context of the caller is suspended. This operation ensures that SQL operations performed in this block (or other blocks called from it) have no dependence or effect on the state of the caller's transaction context.
- When an autonomous block invokes another autonomous block or itself, the called block does not share any transaction context with the calling block.
- However, when an autonomous block invokes a non-autonomous block (that is, one that is not declared to be autonomous), the called block inherits the transaction context of the calling autonomous block.

Transaction Control Statements in Autonomous Blocks

- Transaction control statements in an autonomous PL/SQL block apply only to the currently active autonomous transaction.
- Examples of such statements are:
  - `SET TRANSACTION`
  - `COMMIT`
  - `ROLLBACK`
  - `SAVEPOINT`
  - `ROLLBACK TO SAVEPOINT`
- Similarly, transaction control statements in the main transaction apply only to that transaction and not to any autonomous transaction that it calls.
- For example, rolling back the main transaction to a savepoint taken before the beginning of an autonomous transaction does not roll back the autonomous transaction.