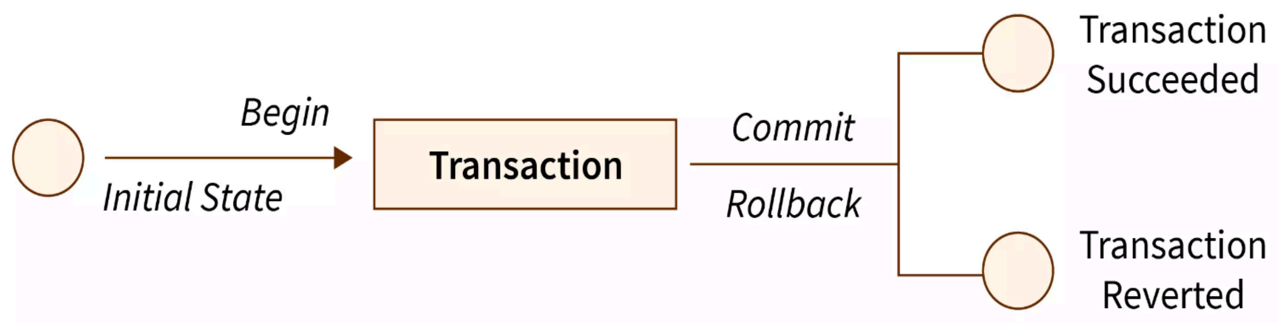
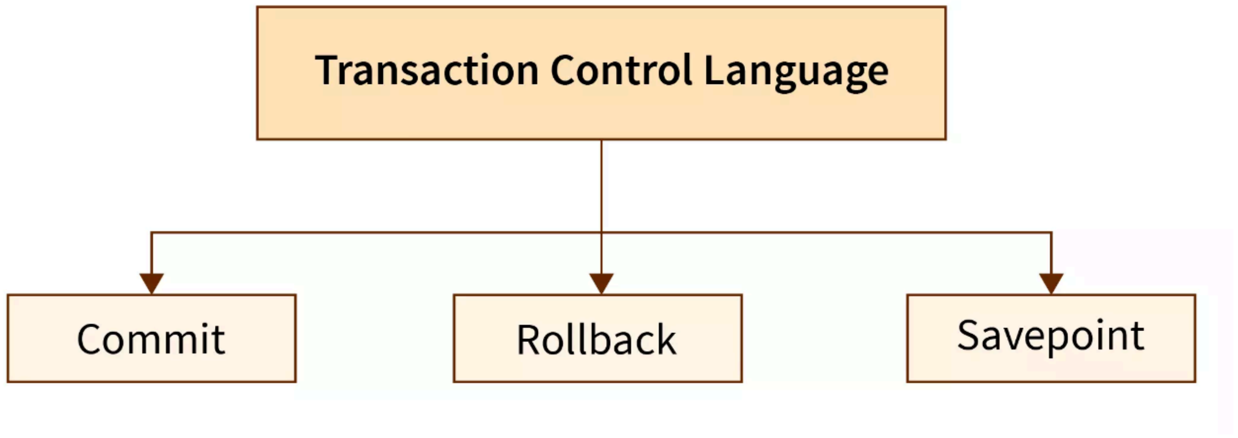
Transaction Control Language (TCL)

- TCL is a set of special commands that deal with the transactions within the database.
- Transaction control in SQL ensures data integrity and consistency in a database by managing changes through transactions.
- A transaction is a sequence of one or more SQL statements that are executed as a single unit of work.
- A transaction is a single, indivisible database action.
- If the transaction contains multiple statements, **it is called a multi-statement transaction (MST).** By default, all transactions are multi-statement transactions.
- For example, suppose we are creating a new record or updating or deleting any record from a table. In that case, we are performing a transaction on the table.
- In SQL, each transaction begins with a particular set of task and ends only when all the tasks in the set is completed successfully. However, if any (or a single) task fails, the transaction is said to fail.



Types of TCL Commands



COMMIT

The **COMMIT** command is used to **permanently save** all changes made during the current transaction. Once a COMMIT is executed, the changes become permanent and visible to other users.

Key Features of COMMIT:

- Saves all modifications made by `INSERT`, `UPDATE`, and `DELETE` statements.
- Releases any **locks** held by the transaction.
- Marks the end of a transaction.
- **Before COMMIT:** The changes are only visible to the current session.
- **After COMMIT:** The changes become permanent and visible to all users.
- **Cannot ROLLBACK after COMMIT:** Once committed, you **cannot undo** the changes.

Example:

```
UPDATE accounts SET balance = balance - 1000 WHERE
account_id = 101;
UPDATE accounts SET balance = balance + 1000 WHERE
account_id = 102;
COMMIT; -- Save changes permanently
```

SAVEPOINT

- The SAVEPOINT command in TCL is basically used to temporarily save a transaction so that we can roll back to that point (saved point) whenever required.
- SAVEPOINT allows you to partially rollback changes instead of rolling back the entire transaction.

Key Features of SAVEPOINT:

- Allows rolling back to a specific point in a transaction.
- Helps in error handling by undoing only specific changes.
- Does **not** commit the transaction, so changes can still be rolled back or committed.
- **You can create multiple SAVEPOINTs** in a transaction.
- **ROLLBACK TO SAVEPOINT only undoes changes after the savepoint**; earlier changes remain.
- **If you ROLLBACK without specifying a SAVEPOINT, the entire transaction is undone.**

- **COMMIT removes all SAVEPOINTs**, making rollback impossible.

Example:

```
UPDATE employees SET salary = salary + 5000 WHERE
emp_id = 101;

SAVEPOINT sp1;  -- Create a savepoint

UPDATE employees SET salary = salary + 3000 WHERE
emp_id = 102;

SAVEPOINT sp2;  -- Create another savepoint

UPDATE employees SET salary = salary + 2000 WHERE
emp_id = 103;

ROLLBACK TO sp2; -- Rollback only to sp2, keeping
changes from sp1

COMMIT;    -- Commit the remaining changes
```

The update for emp_id = 103 is undone.
Updates for emp_id = 101 and 102 remain and are committed.

ROLLBACK

- The **ROLLBACK** command in Oracle SQL is used to **undo** changes made during the current transaction. It restores the database to the state before the transaction began or to a specific **SAVEPOINT**.
- **Undo Changes:** Cancels `INSERT`, `UPDATE`, or `DELETE` operations that have not been committed.
- **Rollback to SAVEPOINT:** Restores the database to a specific checkpoint without undoing the entire transaction.
- **Locks are Released:** Any locks held by the transaction are released after rollback.

Example : Rolling Back an Entire Transaction

```
UPDATE employees SET salary = salary + 5000 WHERE emp_id = 101;

UPDATE employees SET salary = salary + 3000 WHERE emp_id = 102;

-- Suppose an error occurs before committing

ROLLBACK;
```

All salary updates are undone, and the database remains unchanged.

Data Control Language (DCL)

Data Control Language (DCL) is a subset of SQL commands used to control access to data in a database. DCL is crucial for ensuring security and proper data management, especially in multi-user database environments.

The primary DCL commands in SQL include:

★ GRANT: This command is used to give users access privileges to the database. These privileges can include the ability to select, insert, update, delete, and so on, over database objects like tables and views.

- **System Privileges** – Allow users to perform specific actions in the database. Example: `CREATE TABLE`, `ALTER USER`, `DROP ANY TABLE`
- **Object Privileges** – Control access to specific database objects. Example: `SELECT`, `INSERT`, `UPDATE`, `DELETE` on tables

- **Syntax:** `GRANT privilege_name ON object_name TO user_name;`
- **For example,**
  - **GRANT SELECT ON employees TO user123;**
  - ***gives*** `user123` ***the permission to read data from the*** `employees` ***table.***

Example:

- ***Granting System Privileges***

```
GRANT CREATE TABLE TO user1;  -- Allows user1 to create tables

GRANT DROP ANY TABLE TO user2;  -- Allows user2 to drop any table

GRANT CREATE USER TO admin_user;  -- Allows admin_user to create new users

GRANT ALTER USER TO admin_user;  -- Allows admin_user to modify users

GRANT CREATE VIEW TO user3;  -- Allows user3 to create views

GRANT CREATE SEQUENCE TO user4;  -- Allows user4 to create sequences

GRANT CREATE PROCEDURE TO dev_user;  -- Allows dev_user to create stored procedures

GRANT EXECUTE ANY PROCEDURE TO user5;  -- Allows user5 to execute any procedure
GRANT GRANT ANY PRIVILEGE TO super_admin;  -- Allows super_admin to grant any privilege
```

- ***Granting Object Privileges***

```
GRANT SELECT ON employees TO user1;  -- Allows user1 to read data from employees table

GRANT INSERT ON employees TO user2;  -- Allows user2 to insert new rows in employees table

GRANT UPDATE ON employees TO user3;  -- Allows user3 to update records in employees table

GRANT DELETE ON employees TO user4;  -- Allows user4 to delete records from employees table

GRANT ALTER ON employees TO admin_user;  -- Allows admin_user to modify the table structure

GRANT INDEX ON employees TO user5;  -- Allows user5 to create indexes on the table

GRANT REFERENCES ON employees TO user6;  -- Allows user6 to create foreign keys referencing employees
```

GRANT EXECUTE ON my_procedure TO dev_user; -- Allows dev_user to execute my_procedure

- ★ **REVOKE:** This command is used to remove previously granted access privileges from a user.
  - **Syntax:** `REVOKE privilege_name ON object_name FROM user_name;`
  - For example,

REVOKE SELECT ON employees FROM user123;

would remove `user123`'s permission to read data from the `employees` table.

- Database administrators typically use DCL commands. When using these commands, it's important to carefully manage who has access to what data, especially in environments where data sensitivity and user roles vary significantly.
- In some systems, DCL functionality also encompasses commands like `DENY` (specific to certain database systems like Microsoft SQL Server), which explicitly denies specific permissions to a user, even if those permissions are granted through another role or user group.
- Remember, the application and syntax of DCL commands can vary slightly between different SQL database systems, so it's always good to refer to specific documentation for the database you are using.

Example:

```
REVOKE SELECT, INSERT ON employees FROM user1;  -- Removes SELECT, INSERT from user1

REVOKE CREATE TABLE FROM user2;  -- Removes CREATE TABLE privilege from user2

REVOKE manager_role FROM user7;   -- Removes manager_role from user7
```