A **database schema** describes the organization of the database in terms of its constituent objects [tables, views, indices, sequences], their interrelationships and constraints imposed on them.
The structure of constituent tables are referred as **relation schema**, prescribing the inherent constraints [entity integrity, referential integrity, domain] and data types of each constituent attribute of the relation.

SQL stands for Structured Query Language, is a Fourth Generation, Non-Procedural Programming Language. It is an efficient tool to query and update the relational database [say Oracle 11g R2, MySQL 5.5, MS SQL Server 2000].

As a default install, Oracle 11g has **system** as an administrating user with password as **system**. You must remember that password [as well as data] is case sensitive. The object and/or attribute names are all stored as Uppecases.

It is not advisable to query and update database from **system** userspace [unless you are required to perform administrative tasks of user management, schema updation, delegation of authorities, and other].
Therefore, we shall create a normal, non-administrator user, say CS6XX with password same as username [but in lowercase]. Here XX denotes your class roll number.

Firstly, activate SQL Plus console and login to Oracle using

Username : system
Password : system

You will be logged in as **system** user to Oracle, and the SQL prompt will appear.

To know the current user of the database instance, at SQL prompt issue the statement

```
SHOW USER
```

Now, Let us create a user, as mentioned earlier [the current user must be administrator user, say **system**].

```
CREATE USER CS600 IDENTIFIED BY cs600;
```

To verify, whether the user has been created, query the ALL_USERS view.

```
SELECT USERNAME FROM ALL_USERS
    WHERE USERNAME LIKE 'CS6%';
```

Once a user is created, it has to be assigned with set of privileges and permissions to operate and manage the database instance.
Set of permissions and privileges may be incorporated into a database role.

Oracle database has three default roles namely – CONNECT, RESOURCE, DBA. The privileges included in these roles are -

Role	Privileges and Permissions
CONNECT	CREATE SESSION
RESOURCE	CREATE DATABASE LINK, CREATE ROLE, CREATE SYNONYM, CREATE TABLE, CREATE VIEW, CREATE MATERIALIZED VIEW, CREATE TYPE, CREATE PROCEDURE, CREATE TRIGGER, CREATE SEQUENCE
DBA	ALL system privileges with ADMIN option

The CONNECT role enables a user to connect to a database, whereas the RESOURCE role enables the user to create database objects and manipulate them.

Let us assign the role(s) – CONNECT and RESOURCE to a user,

```
GRANT CONNECT, RESOURCE TO CS600;
```

Now log off the system user and log in as CS600 user.

```
DISCONNECT SYSTEM;
CONNECT CS600/cs600
```

You are in a new userspace. Use SHOW USER to verify the logged-in user.

SQL statements can be categorized as following language subset -

SQL Subset	Basic Statements included
DATA DEFINITION LANGUAGE [**DDL**]	ALTER, CREATE, DROP, TRUNCATE
DATA MANIPULATION LANGUAGE [**DML**]	DELETE, INSERT, UPDATE
DATA QUERY LANGUAGE [**DQL**]	SELECT with variuos clauses/options
DATA CONTROL LANGUAGE [**DCL**]	GRANT, REVOKE
TRANSACTION CONTROL LANGUAGE [**TCL**]	COMMIT, ROLLBACK, SAVEPOINT

The Scenario:

it is required to create a database for managing Online Courses. A participant can take any one of the available courses. Each course is uniquely identified by course number and include particulars like course name, credits. Each participant is identified uniquely by participant ID and include particulars like participant name, gender [M/F], course taken.

Considerations:

The database will have two tables (or relations) namely, PARTICIPANT and COURSE. Follwing are the relation schemata -

PARTICIPANT (PID, PNAME, GENDER, CID)

COURSE (CID, CNAME, CREDIT)

PID and CID are primary key attributes for respective tables. The PARTICIPANT table is related to COURSE table in a many-to-one association. Hence CID will be a foreign key in PARTICIPANT referencing COURSE.

As for any PARTICIPANT record to be added, will require the presence of a COURSE record, the COURSE table will be created first.

Database Constraints

Oracle SQL facilitates enforcing constraints on column(s) and/or table(s). These constraints are categorized as -

Constraint Type	Enforcement Mechanism	SQL Statement
Entity Integrity	Set of attribute(s) uniquely identifying each tuple conating no NULL values. A **primary key** [PK] constraint.	PRIMARY KEY
Referential Integrity	Set of attribute(s) for which there exists a record in some other table. A **foreign key** [FK] constraint. The FK must be either a PK or a key with unique constraint in the referenced table.	FOREIGN KEY, REFERENCES
Uniqueness	The column must hold non-duplicating or unique values [NULL is allowed]. A **unique** [UQ] constraint.	UNIQUE
Domain Constraint	The column must always hold a value as per defined range(s). A **check** [CK] constraint.	CHECK
Non Null Constraint	The column must never be NULL. It must always require a value.	NOT NULL

Let us write the SQL schema definitions …

Assumptions for COURSE table -

- CID can take values between 101 and 149.
- CREDIT can take values between 3 and 5.
- CNAME must always have a unique value.

```
CREATE TABLE COURSE (
        CID NUMBER(3) NOT NULL,
        CNAME VARCHAR2(30) NOT NULL,
        CREDIT NUMBER(1),
        CONSTRAINT COURSE_PK_CID PRIMARY KEY (CID),
        CONSTRAINT COURSE_CK_CID CHECK (CID BETWEEN 101 AND 149),
        CONSTRAINT COURSE_CK_CREDIT CHECK (CREDIT BETWEEN 1 AND 5),
        CONSTRAINT COURSE_UQ_CNAME UNIQUE (CNAME)
);
```

Assumptions for PARTICIPANT table -

- PID can take values between 1001 and 9999.
- PNAME must always have a value.
- PGENDER can take values among ['M', 'F'].

```
CREATE TABLE PARTICIPANT (
        PID NUMBER(4) NOT NULL,
        PNAME VARCHAR2(25) NOT NULL,
        GENDER CHAR(1) NOT NULL,
        CID NUMBER(3),
        CONSTRAINT PARTICIPANT_PK_PID PRIMARY KEY (PID),
        CONSTRAINT PARTICIPANT_CK_PID CHECK (PID BETWEEN 1001 AND 9999),
        CONSTRAINT PARTICIPANT_CK_GENDER CHECK (GENDER IN ( 'M', 'F' )),
        CONSTRAINT PARTICIPANT_FK_COURSE_CID FOREIGN KEY
                (CID) REFERENCES COURSE(CID)
);
```

Check the structure of PARTICIPANT and COURSE tables.

```
DESCRIBE COURSE
DESC PARTICIPANT
```

Now, let us add 3 records to TEST.
Observe and analyze the output of last insertion.

```
INSERT INTO COURSE (CID, CNAME, CREDIT)
    VALUES (101, 'Database Management Systems', 5);

INSERT INTO COURSE (CNAME, CID, CREDIT)
    VALUES ('Object-Oriented Systems', 102, 4);

INSERT INTO COURSE (CREDIT, CNAME, CID)
    VALUES (3, 'Basic Operating System', 101);
```

Display the contents of COURSE table

```
SELECT * FROM COURSE;
```

Now, insert 6 records to PARTICIPANT.
Observe and report the effect in last 2 insertion.

```
INSERT INTO PARTICIPANT VALUES (1001, 'Albert DCosta', 'M', 101);
INSERT INTO PARTICIPANT VALUES (1002, 'Foster Silva', 'M', 102);
INSERT INTO PARTICIPANT VALUES (1003, 'Maria Anderson', 'F', 102);
INSERT INTO PARTICIPANT VALUES (1004, 'Pamela Smith', 'F', 101);
INSERT INTO PARTICIPANT VALUES (1005, 'Indiana Jones', 'M', NULL);
INSERT INTO PARTICIPANT VALUES (1006, 'Martinez Wales', 'F', 103);
```

To display all constraints on the PARTICIPANT table and the COURSE table, use the database view USER_CONSTRAINTS as below -

```
SELECT TABLE_NAME, CONSTRAINT_NAME, CONSTRAINT_TYPE
    FROM USER_CONSTRAINTS
    WHERE TABLE_NAME IN ('PARTICIPANT', 'COURSE');
```

The CONSTRAINT_TYPE column will have a single-character type specifier among ['C', 'P', 'R', 'U']. Here,

- 'C' represent a CHECK or a NOT NULL constraints
- 'P' represent a PRIMARY KEY constraint
- 'R' represent a FOREIGN KEY constraint
- 'U' represent a UNIQUE constraint

Store the populated data permanently on the physical storage, the disk.

```
COMMIT;
```

Querying the Database

View all object in the current tablespace

```
SELECT *
    FROM TAB;
```

Display the contents of PARTICIPANT table

```
SELECT *
    FROM PARTICIPANT;
```

List all female participants.

```
SELECT *
    FROM PARTICIPANT
    WHERE GENDER = 'F';
```

List all female participants on increasing order of course ID.

```
SELECT *
    FROM PARTICIPANT
    WHERE GENDER = 'F'
    ORDER BY CID;
```

List all female participants on decreasing order of course ID.

```
SELECT *
    FROM PARTICIPANT
    WHERE GENDER = 'F'
    ORDER BY CID DESC;
```

For all participants list their name and course ID.

```
SELECT PNAME, CID
    FROM PARTICIPANT;
```

Using Aliases

For all participants list their name and gender. You must but display column headings as “ Participant_Name” and “ Sex” respectively.

```
SELECT PNAME AS PARTICIPANT_NAME, GENDER AS SEX
    FROM PARTICIPANT;
```

As GENDER is single-character field, the heading shows 'S' which earlier was 'G'. To get the desired heading we should use SQL *Plus facilities.

Use COLUMN and FORMAT options

```
COLUMN GENDER HEADING "SEX" FORMAT A3
COLUMN PNAME HEADING "PARTICIPANT_NAME" FORMAT A25
```

Now, re-execute the query in its usual form

```
SELECT PNAME, GENDER
    FROM PARTICIPANT;
```

Removing a Tuple

Remove a participant who registered for the course with ID = 101.

```
DELETE FROM PARTICIPANT
    WHERE CID = 101;
```

Remove all participant records.

```
DELETE FROM PARTICIPANT
    WHERE 1 = 1;
```

Recovering the deleted Tuples

Revert the effect of earlier DELETE statements.

```
ROLLBACK;
```

Changing values of fields

For the partcipant with PID = 1002, change its name to "HITMAN ROHIT".

```
UPDATE PARTICIPANT
    SET PNAME = 'HITMAN ROHIT'
    WHERE PID = 1002;
```

NOTE:

- The effect of DDL statements cannot be undone.
- The DDL statement before execution, saves the currently unsaved updates.
- The effect of DML statements can be undone till the point a DDL or COMMIT statement were executed [that resulted in a stable state of the database].
- TRUNCATE statement permanently removes all records from a table without any changes to its structure, whereas DROP statement removes the database object permanently.

The DEFAULT condition

The attributes in a table can be made to assume a default value when the same is not suppiled in the insert statement.

> Consider a table named COLLEGE with attributes college name and college location. The college names are unique and will serve to identify the entity. When college location is not supplied, a default location 'Nagpur' must be inserted.

Let us create this table in a different way...

[we shall practice ALTER, DROP and TRUNCATE commands]

```
DROP TABLE COLLEGE;
CREATE TABLE COLLEGE (
        CNAME VARCHAR2(50) NOT NULL,
        LOCATION VARCHAR2(15) DEFAULT 'Nagpur' NOT NULL
);
```

Before creating a table, it is a good practice to guarantee non-existence of such an object in the table space to avoid conflicting definitions.

Now, add a primary key to COLLEGE table.

```
ALTER TABLE COLLEGE
        ADD PRIMARY KEY (CNAME);
```

A better way is to add primary key as a named constraint.

```
ALTER TABLE COLLEGE
        DROP PRIMARY KEY;
ALTER TABLE COLLEGE
        ADD CONSTRAINT COLLEGE_PK_CNAME PRIMARY KEY (CNAME);
```

Now, add multiple records using anonymous PL/SQL scripts. [uses variables]

```
INSERT INTO COLLEGE (CNAME, LOCATION)
        VALUES ('&CNAME', '&LOCATION');
```

This will allow record insertion **interactively**. Add following records...

1. City College, Mumbai
2. Institute of Science, Nagpur
3. College of Engineering, Pune

To resexecute the anonymous script, use / at the SQL prompt.

Now, add a college as " Shri Ramdeobaba CoEM" to COLLEGE

```
INSERT INTO COLLEGE (CNAME)
    VALUES ('Shri Ramdeobaba CoEM');
```

Observe the contents of COLLEGE table to confirm the effect of DEFAULT option.

Bring the database table to a stable state.

```
COMMIT;
```

Now, remove all college records permanently.

```
SELECT * FROM COLLEGE;
TRUNCATE TABLE COLLEGE;
```

Try undoing the changes caused by TRUNCATE

```
ROLLBACK;
SELECT * FROM COLLEGE;
```

You can not recover to the earlier stable state.

Let us remove the COLLEGE table permanently.

```
DROP TABLE COLLEGE;
```