SQL

- SQL is used to make a request to retrieve data from a Database.
- The DBMS processes the SQL request, retrieves the requested data from the Database, and returns it.
- This process of requesting data from a Database and receiving back the results is called a Database Query and hence the name Structured Query Language.
- SQL is a non-procedural language
  - You can’t write programs like the ones you would have done using C language
  - You can only write questions in English like language called queries which will fetch some data rows from the database.

SQL provides its users the following capabilities:

- Data Definition – lets you define the structure and organization of the stored data and the relationships among the stored data items.
- Data Retrieval – allows you or your application program to retrieve stored data from the database.
- Data Manipulation – allows you or your application program to update the database by adding new data, changing existing data or removing data from the database.
- Access Control – allows you to restrict a user’s ability to add, retrieve or delete data by protecting the data against unauthorized use.
- Data Sharing – allows you to coordinate data sharing by concurrent users’ ensuring that do not interfere with each other.
- Data Integrity - defines integrity constraints in the database protecting it from corruption due to inconsistent updates or system failures.

I. SQL Statements:

SQL has three kinds of statements, DDL, DML and DCL.

**DDL is Data Definition Language statements. Some examples:**

CREATE – to create objects in the database
ALTER – to alter the structure of the database

DROP – to delete objects from the database

TRUNCATE – to remove all records from a table. The space allocated for the records is also removed

COMMENT – to add comments to the data dictionary

**DML is Data Manipulation Language statements. Some examples:**

SELECT – to retrieve data from the database

INSERT – to insert data into a table

UPDATE – to update existing data within a table

DELETE – to delete all records from a table. The space allocated for the records remains intact

CALL – to call a PL/SQL or Java subprogram

EXPLAIN PLAN - explain access path to data

LOCK TABLE – to control concurrency

**DCL is Data Control Language statements. Some examples:**

GRANT – to give user access privileges to database objects

REVOKE – to withdraw access privileges given with the GRANT command

COMMIT – to save the work done

SAVEPOINT – to identify a point in a transaction to which you can later roll back

ROLLBACK – to restore database to original since the last COMMIT

SET TRANSACTION - Change transaction options like what rollback segment to use

**SUMMARY**

SELECT

INSERT, UPDATE, DELETE

CREATE, ALTER, DROP, RENAME, TRUNCATE

COMMIT, ROLLBACK, SAVEPOINT

GRANT, REVOKE

**DATA RETRIEVAL**

**DATA MANIPULATION LANGUAGE**

**DATA DEFINITION LANGUAGE**

**TRANSACTION CONTROL**

**DATA CONTROL LANGUAGE**
Explanation of entity relations with the help of an example:

branch (branch_name, branch_city, assets)
customer (customer_name, customer_street, customer_city)
loan (loan_number, branch_name, amount)
borrower (customer_name, loan_number)
account (account_number, branch_name, balance)
depositor (customer_name, account_number)

**Entity Relational Schema:**

![Entity Relational Schema Diagram]

1. **DDL (Data Definition Language):**

DDL is the subset of SQL commands used to create, modify or remove Oracle database structures.

Queries: Create, Alter and Rollback

Create: Creates a table.

Alter: Add/Modify a column, add a constraint.

Drop: Drop the entire table.
CREATE:

(a) Table

Syntax:

CREATE TABLE < table-name > (< column-name > < datatype (size) >,………) ;

Eg. Form the table branch and declare branch_name as the primary key

create table branch
(

branch_name char(15),
branch_city char(30),
assets integer,

constraint branchPK primary key (branch_name)
)

(b) User

CREATE USER Rajika IDENTIFIED BY password;
CREATE USER username IDENTIFIED how DEFAULT TABLESPACE tableSpaceName
TEMPORARY TABLESPACE tempTableSpaceName
QUOTA qSize ON tableSpaceName
PROFILE profileName
PASSWORD EXPIRE
ACCOUNT lockstatus;

Where:
username: This is the name of the user e.g. Rajika
identified by password where password is the user’s password
tableSpaceName: The tablespace (on the permanent storage media) that will be used to store the user’s database objects.
tempTableSpaceName The tablespace that will be used for temporary storage (for example – used for sorting)
qSize The quota size allowed for the user in the tablespace
tableSpaceName The tablespace for which the qSize applies
profileName  
User profile name – default profile assigned if there is no specific user profile

PASSWORD EXPIRE  
Use only when the password is set to automatically expire

lockstatus  
Lock or Unlock the user;

ALTER:

(a) Table

Syntax:
Alter table < table-name >
[ add ( < column- element > | < constraints > …..]  
[ modify < column- element > … ]
[ drop < options > ] ;

E.g.  Add column to the table loan which holds the information about the percentage interest to be applied on the loan amount.
Alter table loan add interest_rate varhar(2);

(b) User

ALTER command may be used to change a user’s permissions. The alter command would have the same parameters as that of CREATE. The format would be

ALTER USER Rajika <parameter to be changed>;

For example, to change the password for the user Rajika to a new password say “rajika123”, you will issue the following command:

ALTER USER Rajika IDENTIFIED BY rajika123;

DROP:

(a) Table

Syntax:  Drop table tablename;

E.g. Delete the table branch from your schema

Drop table branch;
(b) User

Syntax: Drop user username;

E.g. Drop user Rajika

Drop user Rajika;

2. DML (Data Manipulation Language):

DML statements are SQL commands that allow you to manipulate the data in the database.

Queries: Insert, Update, Delete

- SELECT: Select is used to retrieve data from a table / tables.
- INSERT: Insert is used to add rows of data to a table.
- UPDATE: Update is used to change data in a table.
- DELETE: Delete is used to delete rows of data from a table.

SELECT

To retrieve data from a table you write a query by using the SELECT command. There are three basic syntaxes:

Syntax 1: Select all - To select all columns and all rows from a table you use the wildcard character, an asterisk (*), in place of the column names.

E.g. SELECT * from Customer;
This SELECT will retrieve all the columns from each row and all the rows from the Customer table.

Syntax 2: Select specific columns - To specifically select individual columns from a table you identify the individual column (field) name in the select statement as follows:

SELECT column_name1, column_name2, column_name3 from Tablename;

E.g. SELECT customer_name, customer_street, customer_city from Customer;

Syntax 3: Select specific rows - To select specific rows from a table you use the WHERE clause. This SELECT will retrieve just the rows that meet the specific criteria you identify as part of the WHERE clause:
E.g. SELECT * from Account WHERE account_number = ‘4539’;

The data you store may be case sensitive. If your database follows a case sensitive standard, and you populate a row with upper and lower case, you must always refer to it in the case it was stored.

E.g. If you stored the value “Tandon” in empLastName, you will not get a match if you use the following statement:
SELECT * from EMPLOYEE WHERE empLastName = ‘tandon’;

**INSERT**

**Syntax:** INSERT INTO <table-name> VALUES (<list-of-values>);

**E.g.** Add a new tuple to account

```
insert into account (branch_name, balance, account_number)
values ('Perryridge', 1200, 'A-9732')
```

**UPDATE**

**Syntax:** UPDATE <table-name> SET <column-name> = <value> [,column-name = value,……] [WHERE <condition>]

**E.g.** Increase all accounts with balances over $10,000 by 6%,

```
update account
set balance = balance * 1.06
where balance > 10000
```

**DELETE**

**Syntax:** DELETE FROM <table-name> [WHERE <condition>]

**E.g.** Delete all accounts at every branch located in the city ‘Syracuse’.

```
delete from account
where branch_name in (select branch_name
from branch
where branch_city = 'Syracuse')
```
More Examples of SELECT:

Syntax: select [all/distinct] attribute from table.
for e.g. select [all/distinct] branch_name from loan

Eg. i. Find the number of depositors for each branch.

select branch_name, avg(balance)
from account
group by branch_name
having avg(balance) > 1200

ii. Find all customers who have an account at all branches located in Brooklyn.

select distinct S.customer_name
from depositor as S
where not exists ( 
(select branch_name
from branch
where branch_city = 'Brooklyn')
except
(select R.branch_name
from depositor as T, account as R
where T.account_number = R.account_number and 
S.customer_name = T.customer_name ))

3. DCL (Data Control Language):

GRANT:

SQL GRANT is a command used to provide access or privileges on the database objects to the users.

The Syntax for the GRANT command is:

GRANT privilege_name
ON object_name
TO {user_name |PUBLIC |role_name}
[WITH GRANT OPTION];

- **privilege_name** is the access right or privilege granted to the user. Some of the access rights are ALL, EXECUTE, and SELECT.
- **object_name** is the name of an database object like TABLE, VIEW, STORED PROC and SEQUENCE.
• **user_name** is the name of the user to whom an access right is being granted.
• **PUBLIC** is used to grant access rights to all users.
• **ROLES** are a set of privileges grouped together.
• **WITH GRANT OPTION** - allows a user to grant access rights to other users.

For Example:

a. GRANT DBA TO Rajika;

This command will grant all the Database Administrative privileges to the user Rajika.

b. GRANT SELECT ON branch TO Rajika;

This command grants a SELECT permission on branch table to Rajika.

WITH GRANT option should be used carefully because for example if you GRANT SELECT privilege on branch table to a user1 using the WITH GRANT option, then user1 can GRANT SELECT privilege on branch table to another user, such as user2 etc. Later, if you REVOKE the SELECT privilege on branch from user1, still user2 will have SELECT privilege on employee table.

**REVOKE:**

The REVOKE command removes user access rights or privileges to the database objects.

The Syntax for the REVOKE command is:

```
REVOKE privilege_name
ON object_name
FROM {user_name |PUBLIC |role_name}
```

For Example:

REVOKE SELECT ON branch FROM Rajika;

This command will REVOKE a SELECT privilege on branch table from Rajika. When you REVOKE SELECT privilege on a table from a user, the user will not be able to SELECT data from that table anymore. However, if the user has received SELECT privileges on that table from more than one user, he/she can SELECT from that table until everyone who granted the permission revokes it. You cannot REVOKE privileges if they were not initially granted by you.