Introduction to SQL

SQL – The query language

- SQL stands for "Structured Query Language".
- Used for accessing and modifying information in the database.

- It can define the structure of the data, modify the data and specify the constraints in a database.

Overview of SQL

- IBM developed the original version of SQL, originally called ‘Sequel’ in 1970s.
- SQL established itself as the standard relational database language.
- In 1986, the ANSI and ISO published an SQL standard called ‘SQL-86’.
- Most recently SQL:2008.

Overview of SQL

The SQL language has several parts:

- **Data Definition Language(DDL)**
- **Data Manipulation Language(DML)**
- **Integrity**
- **View definition**
- **Transaction Control**
- **Embedded SQL and Dynamic SQL**
- **Authorization**

DDL

- The DDL provides commands for defining relation schema, deleting relations, and modifying relation schemas.
- They are:
  - CREATE
  - ALTER
  - DROP
  - RENAME
  - TRUNCATE

DML

- The DML provides the ability to query information from the database and to insert tuples into, delete tuples from, and modify tuples in the database.
- They are :
- SELECT(DQL)
- INSERT
- DELETE
- UPDATE

Integrity:

- **The SQL DDL include commands for specifying integrity constraints.**

View definition:

- **The SQL DDL includes commands for defining views.**

Transaction control:

- Specifying commands for beginning and ending transactions.
- TCL commands
- COMMIT
- ROLLBACK
- SAVEPOINT

Embedded SQL and Dynamic SQL:

- **This specify how SQL statements can be embedded within general-purpose programming languages, such as C, C++ etc.**

Authorization:

- Specify commands for access rights to relations and views.
- DCL Commands
- GRANT
- REVOKE

SQL Data-Definition

- The SQL DDL allows specification of not only a set of relations, but also information about each relation, including:

1. The schema for each relation.
2. The types of values associated with each attribute.
3. The integrity constraints.
4. The set of indices to be maintained for each relation.
5. The security and authorization information for each relation.
6. The physical storage structure of each relation on disk.

SQL Data-Definition

Basic Data Types

- **Char(n) : A fixed length character string with user-specified length n.** The full form, **character**
- **Varchar(n) : A variable-length character string with user-specified maximum length n.** The full form, **character varying**
- **Int : An integer**
- **Smallint : A small integer**
- **Numeric(p,d) : A fixed point number with user-specified precision.** The number consists of *p* digits (plus a sign), and *d* of the *p* digits are to the right of the decimal point.
- Thus, **numeric**(3,1) allows 44.5 to be stored exactly, but neither 444.5 or 0.32 can be stored exactly in a field of this type.

SQL Data-Definition

- **Real, double precision : Floating point and double precision floating point numbers.**
- **Float(n) : A floating point number with precision of at least n digits.**

SQL Data-Definition

- **Basic Schema Definition**

1.DDL COMMANDS

- **CREATE**
- **Synatx**
- **CREATE TABLE tablename(columname1 datatype, columname2 datatype…column datatype);**
- **Eg:**
- **CREATE TABLE department (dept_name varchar(20), building varchar(15),budget numeric(12,2), primary key(dept_name));**

- **The CREATE TABLE AS Statement**
- **create a table from an existing table by copying the existing table's columns.**
- **SYNTAX - COPYING ALL COLUMNS FROM ANOTHER TABLE**
- **CREATE TABLE new_table AS (SELECT * FROM old_table);**

If there were records in the OLD table, then the NEW table would also contain the records selected by the SELECT statement.

- **The CREATE TABLE AS Statement**
- **SYNTAX - COPYING SELECTED COLUMNS FROM ANOTHER TABLE**

- **CREATE TABLE new_table AS (SELECT column_1, column2, ... column_n FROM old_table);**

Eg;

CREATE TABLE TEMP1 AS (SELECT NAME FROM TEST1);

- The CREATE TABLE AS Statement
- SYNTAX - COPYING SELECTED COLUMNS FROM MULTIPLE TABLES
- CREATE TABLE new_table AS (SELECT TABLE1.column_1, .. TABLE2.colum1,.. FROM old_table_1, old_table_2, ... old_table_n);
- Eg:
- CREATE TABLE TEMP3 AS (SELECT TEST1.NAME, TEST2.SALARY FROM TEST1, TEST2 WHERE TEST1.NAME=TEST2.NAME_S);

- The ALTER TABLE Statement
- ALTER TABLE statement is used to add, delete, or modify columns in an existing table.
- To add a column in a table
- syntax:
- ALTER TABLE table_name ADD column_name datatype;

- Eg) ALTER TABLE customer ADD Gender char(1);

- To delete a column in a table,
- syntax:
- ALTER TABLE table_name DROP COLUMN column_name;
- Eg)ALTER table customer drop column Birth_Date;
- To change the data type of a column in a table,
- syntax:
- ALTER TABLE table_name MODIFY column_name datatype
- Eg) ALTER TABLE customer MODIFY Address char(100);

- To rename a column in a table,
- syntax:
- ALTER TABLE table_name RENAME COLUMN column 1 TO column 2;
- Eg) ALTER table customer RENAME COLUMN Address TO Addr;

- **The DROP TABLE Statement**
- **The DROP TABLE statement is used to delete a table.**
- **DROP TABLE table_name;**
- **Eg;**
- **DROP TABLE customer;**

- **The TRUNCATE TABLE Statement**
- **If we only want to delete the data inside the table, and not the table itself then, use the TRUNCATE TABLE statement:**
- **TRUNCATE TABLE table_name**

- **The RENAME Statement**
- **The SQL RENAME command is used to change the name of the table.**
- **Syntax to rename a table**
- **RENAME old_table_name To new_table_name;**
- **Rename customer TO Cust;**

DESC

- Used to describe the structure of the table
- Synatx:
- DESC tablename; or DESCRIBE tablename;
- Eg; DESC student;

2. DML Commands

- The INSERT INTO Statement
- INSERT INTO statement is used to insert a new row in a table.
- It is possible to write the INSERT INTO statement in two forms.
- The first form doesn't specify the column names where the data will be inserted, only their values:
- INSERT INTO table_name VALUES (value1, value2, value3,...)
- The second form specifies both the column names and the values to be inserted:
- INSERT INTO table_name (column1, column2, column3,...) VALUES (value1, value2, value3,...)

- The UPDATE statement
- Used to update existing records in a table.
- UPDATE table_name SET column1=value, column2=value2,... WHERE some_column=some_value
- Eg;
- UPDATE Store_Info SET Sales = 500 WHERE store_name = "Los Angeles"
  AND Date = "Jan-08-1999"

- The DELETE Statement
- The DELETE statement is used to delete rows in a table.
- DELETE FROM table_name WHERE some_column=some_value
- Eg)DELETE FROM Store_Info WHERE store_name = "Los Angeles"
- It is possible to delete all rows in a table without deleting the table.
- DELETE FROM table_name
  or
  DELETE * FROM table_name

- **The SELECT Statement**
- **The SELECT statement is used to select data from a database.**
- **SELECT column_name(s) FROM table_name**
- **SELECT * FROM table_name**

- **WHERE clause**
- **The WHERE clause is used to extract only those records that fulfill a specified criterion.**
- **SELECT column_name(s) FROM table_name WHERE column_name operator value**

- Used to specify a condition while fetching the data from single table or joining with multiple tables.
- Used to filter the records and fetching only necessary records.
- It can used in SELECT, UPDATE, DELETE statement, etc.,
- Syntax:
- SELECT column1, column2, columnN FROM table_name WHERE [condition]
- Eg
- SELECT ID, NAME, SALARY FROM CUSTOMERS WHERE SALARY > 2000;

- **SQL allows the use of the logical connectives AND, OR, NOT in the WHERE clause.**
- **Logical connectives can be expressions involving comparison operators <,<=,>,>=,=,<>.**

Basic Structure of SQL Queries

The basic structure of an SQL query consists of 3 clauses:

- **SELECT**
- **FROM**
- **WHERE**
- **The query takes as its input the relations listed in the from clause, operates on them as specified in the where and select clauses, and then produces a relation as the result.**

1.Queries on single relation

- In cases where we want to eliminate duplicates, we insert the keyword DISTINCT after SELECT.
- SELECT DISTINCT columnname FROM tablename;
- The result of the query contains each column at most once.

- **DISTINCT keyword is used with SELECT statement to eliminate all the duplicate records and fetching only unique records.**
- **Syntax:**
- **SELECT DISTINCT column1, column2,.....columnN FROM table_name WHERE [condition]**
- **Eg;**
- **SELECT DISTINCT SALARY FROM TAB1 ORDER BY SALARY;**

2. Queries on Multiple relations

- Queries often need to access information from MULTIPLE relations.
- Eg;
- SELECT name, instructor.dept_name, building FROM instructor, department WHERE instructor.deptname=department.dept_name;

- **The naming convention requires that the relations that are present in the FROM clause have distinct names.**
- **The FROM clause by itself is a Cartesian product of the relation.**

3. Natural join

- The natural join operation operates on two relations and produces a relation as the result.
- Eg;
- Select name,course_id from instructor NATURAL JOIN teaches.

Additional Basic Operations

1. The Rename operation

- Using AS clause
- To rename a relation is to replace a long relation_name with a shortened.
- Eg;
- Select name as nme,courseid, from instructor, teaches where instructor.id=teaches.id;
- Select T.name, S.courseid, from instructor as T, teaches AS S where T.id=S.id;

The identifiers T and S is referred to as correlation name in the SQL std., commonly referred to as a table alias or a correlation variable or a tuple variable.

2. String operations

- The % character matches any substring
- The _ matches any character.
- Use LIKE to match patterns.
- Select deptname from dept where building LIKE ‘%Watson%’

- LIKE clause is used to compare a value to similar values using wildcard operators.
- There are two wildcards used in conjunction with the LIKE operator:
- The percent sign (%)
- The underscore (_)
- The percent sign represents zero, one, or multiple characters.
- The underscore represents a single number or character.

- **Eg;**
- **WHERE SALARY LIKE '200%‘ Finds any values that start with 200**
- **WHERE SALARY LIKE '%200%‘ Finds any values that have 200 in any position**
- **WHERE SALARY LIKE '_00%‘ Finds any values that have 00 in the second and third positions**
- **WHERE SALARY LIKE '%2‘ Finds any values that end with 2**

3. Attribute specification in Select Clause

- Eg;
- SELECT instructor.* FROM instructor,teaches WHERE instructor.id=teaches.id;

ORDER BY

- **ORDER BY clause is used to sort the data in ascending or descending order, based on one or more columns.**
- **Syntax:**
- **SELECT column-list FROM table_name [WHERE condition] [ORDER BY column1, column2, .. columnN] [ASC | DESC];**
- **Following is an example, which would sort the result in ascending order by NAME**
- **SELECT * FROM Tab1 ORDER BY Name;**
- **Following is an example, which would sort the result in descending order by NAME:**
- **SELECT * FROM Tab1 ORDER BY Name DESC;**

4. Ordering the display of Tuples

- ORDER BY clause
- SELECT name FROM instructor WHERE dept_name='Physics' ORDER BY name;
- SELECT * FROM instructor ORDER BY salary DESC;

5. WHERE clause predicates

- **Use with BETWEEN,AND,OR etc..**

- **AND Operator:**
- **The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause.**
- **Syntax:**
- **SELECT column1, column2, columnN FROM table_name WHERE [condition1] AND [condition2]...AND [conditionN];**

- **SELECT Id, Name, Salary FROM CUSTOMERS WHERE Salary > 2000 AND age < 25;**

- **OR Operator:**
- **The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause.**
- **SELECT column1, column2, columnN FROM table_name WHERE [condition1] OR [condition2].**

- **SELECT Id, Name, SALARY FROM CUSTOMERS WHERE Salary > 2000 OR age < 25;**

- **"AND" & "OR Operator:**
- **SELECT * FROM suppliers WHERE (city = 'New York' AND name = 'IBM') OR (ranking >= 10);**

- IN
- SELECT * FROM STUDENT WHERE FIRST_NAME IN ('APPU', 'RAJI', 'SAJAN');
- This statement will select the all columns from the student table where the first_name is equal to either: APPU, RAJI or SAJAN. It will return the rows if it is ANY of these values.
- The IN conditional operator can be rewritten by using compound conditions using the equals operator and combining it with OR

```
SELECT * FROM student WHERE First_name='APPU' OR
First_name ='RAJI' OR First_name = 'SAJAN';
```

- **BETWEEN NUMERIC**
- **The BETWEEN conditional operator is used to test to see whether or not a value (stated before the keyword BETWEEN) is "between" the two values stated after the keyword BETWEEN.**
- **SELECT * FROM tablename WHERE column BETWEEN 20 AND 30;**

- **BETWEEN DATE**
- **SELECT * FROM Temp WHERE dob BETWEEN '02-JAN-14' AND '12-MAR-14';**

BETWEEN USING NOT OPERATOR

- SELECT * FROM Test WHERE Salary NOT BETWEEN 4000 AND 5000;

SET OPERATIONS

SET OPERATIONS

- **UNION**
- **INTERSECT**
- **EXCEPT**

SET OPERATIONS

- **UNION clause/operator is used to combine the result-set of two or more SELECT statements**
- **SELECT statement within the UNION must have the same number of columns. With same data type**
- **UNION operator selects only distinct values by default.**
- **syntax**
- **SELECT *column_name(s)* FROM *table1***
  UNION
  SELECT *column_name(s)* FROM *table2*;

SET OPERATIONS

- **To allow duplicate values, use the ALL keyword with UNION.**
- **Syntax**
- **SELECT *column_name(s)* FROM *table1***
  UNION ALL
  SELECT *column_name(s)* FROM *table2*;
- **Eg;**
- **SELECT Name FROM Test1 UNION SELECT Name FROM Test2;**
- **SELECT Name FROM Test1 UNION ALL SELECT Name FROM Test2;**

SET OPERATIONS

- **INTERSECT**
- **Returns any distinct values that are returned by both the query on the left and right sides of the INTERSECT operand.**
- **Syntax**
- **SELECT *column_name(s)* FROM *table1***
  INTERSECT
  SELECT *column_name(s)* FROM *table2*;
- **Eg;**
- **SELECT Name FROM Test1 INTERSECT SELECT Name FROM Test2;**

SET OPERATIONS

- **EXCEPT**
- **returns any distinct values from the left query that are not also found on the right query.**

- **SELECT Name FROM Test1 MINUS SELECT Name FROM Test2;**

Null Values

- IS NULL
- Select name from instructor where salary IS NULL;

AGGREGATE FUNCTIONS

AGGREGATE FUNCTIONS

- **AVG- compute average**
- **MIN- compute minimum**
- **MAX- compute maximum**
- **SUM- compute sum**
- **COUNT- counts number of rows**

- SQL COUNT Function - Used to count the number of rows in a database table.
- To count total number of rows in this table, then
- SELECT COUNT(*) FROM TAB1 ;
- want to count the number of records for AJIN, then
- SELECT COUNT(*) FROM TAB1 WHERE name=“AJIN";

- SQL MAX Function To select the highest (maximum) value for a certain column.
- to fetch maximum value of AGE,
- SELECT MAX(AGE) FROM TAB1;
- SQL MIN Function - to select the lowest (minimum) value for a certain column.
- to fetch minimum value of AGE
- SELECT MIN(AGE) FROM TAB1;

- SQL AVG Function - selects the average value for certain table column.
- SELECT AVG(AGE) FROM TAB1;
- SQL SUM Function - selecting the total for a numeric column.
- SELECT SUM(AGE) FROM TAB1;
- To take sum of various records set using GROUP BY clause.
- Following example will sum up all the records related to a single person and you will have total DEPOSIT by every person.
- SELECT NAME,SUM(DEPOSIT) FROM BANK GROUP BY NAME;

- SQL SQRT Functions - This is used to generate a square root of a given number.
- SELECT name, SQRT(AGE) FROM TAB1;

- SQL UPPER Functions - converts the value of a field to uppercase
- SELECT UPPER(FIRST_NAME) FROM STUDENT;

- SQL LOWER Functions - converts the value of a field to LOWERCASE
- SELECT LOWER(FIRST_NAME) FROM STUDENT;

1. Aggregation with GROUPING

- GROUP BY clause
- Used to arrange identical data into groups.
- Syntax:

- SELECT column1, column2 FROM table_name GROUP BY column1, column2;

- **GROUP BY** clause
- **Consider the CUSTOMERS table is having the following records:**

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Ramesh	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	kaushik	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

- **to know the total amount of salary on each customer, then GROUP BY query would be as follows:**
- **SELECT NAME, SUM(SALARY) FROM CUSTOMER GROUP BY NAME;**

NAME	SUM(SALARY)
Hardik	8500.00
kaushik	8500.00
Komal	4500.00
Muffy	10000.00
Ramesh	3500.00

- **Sql does not allow the use of DISTINCT with count(*).**
- **It is illegal to use DISTINCT with MAX and MIN.**

2. The HAVING clause

- The HAVING imposes a condition on the GROUP BY clause which further filters the groups created by the GROUP BY clause.
- Eg;
- SELECT SUM(salary) FROM employee GROUP BY dept_name HAVING dept_name='CS';
- Only GROUP functions can be used in the HAVING clause.

Modifications of database

- **DELETE**
- **INSERT INTO**
- **UPDATE**

- **DELETE**
- **Eg;**
- **DELETE FROM instructor WHERE dept_name=‘Finance’;**
- **DELETE FROM instructor WHERE dept_name IN (SELECT dept_name FROM department WHERE building=‘Watson’);**
- **This DELETE requests first finds all departments located in Watson, and then deletes all instructor tuples pertaining to those departments.**

Nested Queries

- A **nested query** is a **query that has another query embedded within it**; the embedded query is called a **subquery**.
- A subquery is usually added in the **WHERE** Clause of the sql statement.
- **The subquery (inner query) executes once before the main query (outer query) executes.**

```
SELECT    select_list
FROM      table
WHERE     expr operator
                (SELECT    select_list
                 FROM      table);
```

- Two tables ‘STUDENT' and ‘MARKS' with common field ‘ID'.

ID	NAME
1	aji
2	saji
3	raji
4	aa

4 rows returned

ID	MARK
1	80
2	70
3	40
4	50

4 rows returned

- To write a query to identify all students who get better marks than that of the student who's ID is ‘2‘.
- If we know the mark of ID ’2’ then
- SELECT A.ID,A.NAME, B.MARK FROM STUDENT A, MARK B WHERE A.ID=B.ID AND B.MARK >70;

ID	NAME	MARK
1	aji	80

1 rows returned in 0.03 sec

- But we do not know the marks of ‘2'.
- we require two queries (Nested query)
- One query returns the marks of '2' and
- Second query identifies the students who get better marks than the result of the first query
- SELECT A.ID,A.NAME, B.MARK FROM STUDENT A, MARKS B WHERE A.ID=B.ID AND B.MARK > (SELECT MARK FROM MARKS WHERE ID=2);

- **Subqueries must be enclosed within parentheses.**
- **Inner query can have only one column in the SELECT clause, unless multiple columns are in the main query.**
- **Inner Query cannot use An ORDER BY clause, although the main query can use an ORDER BY.**
- **The BETWEEN operator cannot be used with a inner query; however, the BETWEEN operator can be used within the main query.**

- **Subqueries can be used with**
- **SELECT**
- **INSERT**
- **UPDATE, and**
- **DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN etc.**

- **Subquerie in select statement** **Tab1** **Tab2**

ROLLNO	NAME
1	ABIN
2	BABY
3	SUJIN
4	DEVI

4 rows returned in 0.0(

MID	ROLLNO	MARK
1	3	55
2	3	50
3	3	56
4	3	50
5	3	43
6	4	23
7	4	34

7 rows returned in 0.00 seconds

- **List all details from tab2, where rollno =4 in tab1**
- **select * from tab2 where rollno in (select rollno from tab1 where rollno=4)**

MID	ROLLNO	MARK
6	4	23
7	4	34

2 rows returned in 0.00 seconds

- **Subquerie in insert statement**

Tab1

ROLLNO	NAME
1	ABIN
2	BABY
3	SUJIN
4	DEVI

4 rows returned in 0.00

Tab4

ROLLNO	NAME
1	ABIN
2	BABY
3	SUJIN
4	DEVI

4 rows returned in 0.00

- **Consider table tab1 & tab4 with similar structure.**
- **To copy the records from tab1 to tab4 where rollno=3**
- **insert into tab4 select * from tab1 where rollno in (select rollno from tab1 where rollno=3)**

ROLLNO	NAME
1	ABIN
2	BABY
3	SUJIN
4	DEVI
3	SUJIN

5 rows returned in 0.10

- **Subquerie in update statement** Tab1 Tab2

ROLLNO	NAME
1	ABIN
2	BABY
3	SUJIN
4	DEVI

4 rows returned in 0.0(

MID	ROLLNO	MARK
1	3	55
2	3	50
3	3	56
4	3	50
5	3	43
6	4	23
7	4	34

7 rows returned in 0.00 seconds

- **updating the marks with 10 in tab2, whose rollno = 4 in tab1**
- **update tab2 set mark=mark+10 where rollno in (select rollno from tab1 where rollno=4)**

MID	ROLLNO	MARK
1	3	55
2	3	50
3	3	56
4	3	50
5	3	43
6	4	33

- **Subquerie in delete statement**

Tab1

ROLLNO	NAME
1	ABIN
2	BABY
3	SUJIN
4	DEVI

4 rows returned in 0.0

Tab2

MID	ROLLNO	MARK
1	3	55
2	3	50
3	3	56
4	3	50
5	3	43
6	4	23
7	4	34

7 rows returned in 0.00 seconds

- **Delete all details from tab2, where rollno is 4 in tab1**
- **delete from tab2 where rollno in (select rollno from tab1 where rollno=4)**

MID	ROLLNO	MARK
1	3	55
2	3	50
3	3	56
4	3	50
5	3	43

5 rows returned in 0.00 seconds