Introduction to SQL

Sangeeta Bhandari

PG department of computer Science & IT

HMV, Jalandhar

Outline

- SQL
- SQL Featurtes
- DDL, DML, DCL, DTL
- RDBMS, Table etc.
- Constraints
- SQL Commands with Examples

What is SQL ?

- SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in a relational database.
- SQL is the standard language for Relational Database System. All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their standard database language.

SQL is Popular because..

- Allows users to access data in the relational database management systems.
- Allows users to describe the data.
- Allows users to define the data in a database and manipulate that data.
- Allows to embed within other languages using SQL modules, libraries & pre-compilers.
- Allows users to create and drop databases and tables.
- Allows users to create view, stored procedure, functions in a database.
- Allows users to set permissions on tables, procedures and views.

SQL Commands

- SQL commands can be classified into the following groups based on their nature –
- Data definition language
- Data manipulation language
- Data control language
- Data transaction language

Data definition language

Sr.No.	Command & Description
1	**CREATE** Creates a new table, a view of a table, or other object in the database.
2	**ALTER** Modifies an existing database object, such as a table.
3	**DROP** Deletes an entire table, a view of a table or other objects in the database.
4.	**RENAME** Renames existing object

Data manipulation language

Sr.No.	Command & Description
1	**SELECT** Retrieves certain records from one or more tables.
2	**INSERT** Creates a record.
3	**UPDATE** Modifies records.
4	**DELETE** Deletes records.

Data Control Language

Sr.No.	Command & Description
1	**GRANT** Gives a privilege to user.
2	**REVOKE** Takes back privileges granted from user.

Data Transaction Language

Sr.No.	Command & Description
1.	•**COMMIT** commits a Transaction
2.	•**ROLLBACK** rollback a transaction in case of any error occurs
3.	**SAVEPOINT** to rollback the transaction making points within groups
4.	**SET TRANSACTION**

RDBMS

- **R**elational **D**atabase **M**anagement **S**ystem RDBMS is the basis for SQL, and for all modern database systems like MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.
- A Relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model as introduced by E. F. Codd.

Table

- The data in an RDBMS is stored in database objects which are called as **tables**. This table is basically a collection of related data entries and it consists of numerous columns and rows.

TABLE: EXAMPLE

ID	Name	Age	Address	Salary
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Rahul	40	jalandhar	2500.00
4	Jatin	35	amritsar	1800.00
5	Sufi	22	Kota	4500.00
6.	Balbir	28	mumbai	8500.00
7.	Ankur	27	Banglore	10000.00
8.	Amit	42	Chandigarh	6500.00
9.	Pankaj	32	Patiala	6500.00

Field, Record and Column in Table

- Every table is broken up into smaller entities called fields. The fields in the CUSTOMERS table consist of ID, NAME, AGE, ADDRESS and SALARY.
- A record is also called as a row of data is each individual entry that exists in a table.
- A column is a vertical entity in a table that contains all information associated with a specific field in a table.

SQL Constraints

- Constraints are the rules enforced on data columns on a table. These are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database.

SQL Constraints

- Following are some of the most commonly used constraints available in SQL
- NOT NULL Constraint – Ensures that a column cannot have a NULL value.
- DEFAULT Constraint – Provides a default value for a column when none is specified.
- UNIQUE Constraint – Ensures that all the values in a column are different.
- PRIMARY Key – Uniquely identifies each row/record in a database table.
- FOREIGN Key – Uniquely identifies a row/record in any another database table.
- CHECK Constraint – The CHECK constraint ensures that all values in a column satisfy certain conditions.
- INDEX – Used to create and retrieve data from the database very quickly.

Create Table Command

- SQL CREATE TABLE Statement

  CREATE TABLE table_name

  (column1 datatype,

  column2 datatype,

  column3 datatype,

  ..... columnN datatype,

  PRIMARY KEY(one or more columns));

Create Table: Example

- creates a CUSTOMERS table with an ID as a primary key and NOT NULL are the constraints showing that these fields cannot be NULL while creating records in this table –
- SQL> CREATE TABLE CUSTOMERS

  (ID INT NOT NULL, NAME VARCHAR (20) NOT NULL, AGE INT NOT NULL,

  ADDRESS CHAR (25) , SALARY DECIMAL (18, 2),

  PRIMARY KEY (ID));

Insert command

- SQL INSERT INTO Statement

  INSERT INTO table_name

  (column1, column2....columnN)

  VALUES (value1, value2....valueN);

Insert Into: Example

- INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00);
  INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (2, 'Khilan', 25, 'Delhi', 1500.00);
- INSERT INTO CUSTOMERS VALUES (7, 'Muffy', 24, 'Indore', 10000.00);

Select Command

SQL SELECT Statement

- SELECT column1, column2....columnN FROM table_name;

SQL DISTINCT Clause

- SELECT DISTINCT column1, column2....columnN FROM table_name;

Select Command

SQL WHERE Clause

- SELECT column1, column2....columnN FROM table_name WHERE CONDITION;

SQL AND/OR Clause

- SELECT column1, column2....columnN FROM table_name WHERE CONDITION-1 {AND|OR} CONDITION-2;

Select Command

SQL IN Clause

- SELECT column1, column2....columnN FROM table_name WHERE column_name IN (val-1, val-2,...val-N);

SQL BETWEEN Clause

- SELECT column1, column2....columnN FROM table_name WHERE column_name BETWEEN val-1 AND val-2;

Select Command

SQL LIKE Clause

- SELECT column1, column2….columnN FROM table_name WHERE column_name LIKE { PATTERN };

SQL ORDER BY Clause

- SELECT column1, column2….columnN FROM table_name WHERE CONDITION ORDER BY column_name {ASC | DESC};

Select Command

SQL GROUP BY Clause

- SELECT SUM(column_name) FROM table_name WHERE CONDITION GROUP BY column_name;

SQL COUNT Clause

- SELECT COUNT(column_name) FROM table_name WHERE CONDITION;

SQL HAVING Clause

- SELECT SUM(column_name) FROM table_name WHERE CONDITION GROUP BY column_name HAVING (arithematic function condition);

Select Command: Example

- SQL> SELECT ID, NAME, SALARY FROM CUSTOMERS;
- SQL> SELECT * FROM CUSTOMERS;
- SQL> SELECT ID, NAME, SALARY FROM CUSTOMERS WHERE SALARY > 2000;

Select Command: Example

Fetch the ID, Name and Salary fields from the CUSTOMERS table, where the salary is greater than 2000 and the age is less than 25 years −

- SQL> SELECT ID, NAME, SALARY FROM CUSTOMERS WHERE SALARY > 2000 AND age < 25;

Fetch the ID, Name and Salary fields from the CUSTOMERS table, where the salary is greater than 2000 OR the age is less than 25 years.

- SQL> SELECT ID, NAME, SALARY FROM CUSTOMERS WHERE SALARY > 2000 OR age < 25;

Select Command, Order By

- The SQL **ORDER BY** clause is used to sort the data in ascending or descending order, based on one or more columns. Some databases sort the query results in an ascending order by default.
- SELECT column-list FROM table_name [WHERE condition] [ORDER BY column1, column2, .. columnN] [ASC | DESC];

Order By: Example

- SQL> SELECT * FROM CUSTOMERS ORDER BY NAME, SALARY;
- SQL> SELECT * FROM CUSTOMERS ORDER BY NAME DESC;

Group By

- The SQL **GROUP BY** clause is used in collaboration with the SELECT statement to arrange identical data into groups. This GROUP BY clause follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause.
- To know the total amount of the salary on each customer, then the GROUP BY query would be as follows.
- SQL> SELECT NAME, SUM(SALARY) FROM CUSTOMERS GROUP BY NAME;

Having Clause

- The **HAVING Clause** enables you to specify conditions that filter which group results appear in the results.
- The WHERE clause places conditions on the selected columns, whereas the HAVING clause places conditions on groups created by the GROUP BY clause.

Having Clause

- SELECT column1, column2 FROM table1, table2 WHERE [conditions] GROUP BY column1, column2 HAVING [conditions] ORDER BY column1, column2
- SQL > SELECT ID, NAME, AGE, ADDRESS, SALARY FROM CUSTOMERS GROUP BY age HAVING COUNT(age) >= 2;

Update Command

The SQL **UPDATE** Query is used to modify the existing records in a table. You can use the WHERE clause with the UPDATE query to update the selected rows, otherwise all the rows would be affected.

SQL UPDATE Statement

```
UPDATE table_name
SET column1 = value1, column2 =
value2....columnN=valueN
 [ WHERE CONDITION ];
```

Update Command

```
SQL> UPDATE CUSTOMERS SET ADDRESS =
   'Pune' WHERE ID = 6;
SQL> UPDATE CUSTOMERS SET ADDRESS =
   'Pune', SALARY = 1000.00;
```

Drop Table

SQL DROP TABLE Statement

SQL> DROP TABLE table_name;

Create Index Command

SQL CREATE INDEX Statement

CREATE UNIQUE INDEX index_name ON table_name (column1, column2,...columnN);

SQL DROP INDEX Statement

ALTER TABLE table_name DROP INDEX index_name;

Alter Table Command

The SQL **ALTER TABLE** command is used to add, delete or modify columns in an existing table. You should also use the ALTER TABLE command to add and drop various constraints on an existing table.

ALTER TABLE Statement

- ALTER TABLE table_name {ADD | DROP | MODIFY} column_name {data_ype};

Alter Table

- to add a **New Column** in an existing table
- ALTER TABLE table_name ADD column_name datatype;
- For example:
- ALTER TABLE CUSTOMERS ADD SEX char(1);

Alter Table

- to **DROP COLUMN** in an existing table
- ALTER TABLE table_name DROP COLUMN column_name;
- For Example:
- ALTER TABLE CUSTOMERS DROP SEX;

Alter Table

- to change the **DATA TYPE** of a column in a table
- ALTER TABLE table_name MODIFY COLUMN column_name datatype;
- to **ADD PRIMARY KEY** constraint to a table
- ALTER TABLE table_name ADD CONSTRAINT MyPrimaryKey PRIMARY KEY (column1, column2...);

Alter table Rename

SQL ALTER TABLE Statement (Rename)

- ALTER TABLE table_name RENAME TO new_table_name;

Delete Command

The SQL DELETE Query is used to delete the existing records from a table.You can use the WHERE clause with a DELETE query to delete the selected rows, otherwise all the records would be deleted.

SQL DELETE Statement

- DELETE FROM table_name
  WHERE {CONDITION};

Delete Command: Example

- SQL> DELETE FROM CUSTOMERS WHERE ID = 6;
- to DELETE all the records from the CUSTOMERS table –
- SQL> DELETE FROM CUSTOMERS;

Commit and Rollback

SQL COMMIT Statement

- COMMIT;

SQL ROLLBACK Statement

- ROLLBACK;

THANKS