By: N. Selvakumar
Email: anselva@yahoo.com
Web: www.nselva.com

SQL

SQL stands for Structured Query Language. SQL is used to communicate with a database. SQL statements are used to perform tasks such as add, edit, delete data in a table or retrieve data from a table.

Some common relational database management systems that use SQL statements are MySQL, MS Access, MS SQL Server, Oracle, and Paradox also have their own additional features. The standard SQL commands such as "Select", "Insert", "Update", "Delete", "Create", "Alter" and "Drop" can be used to accomplish almost everything that one needs to do with a database.

Features of SQL statements:

- SQL statement provides an interface between relational database such as MySQL, MS SQL Server, Access, Oracle, and Paradox and User (Application Programs). All SQL statements are instructions to the database.

- All programs written in SQL are portable, means they can often be moved from one database to another with little modification.

In database, stores information in the form of tables, consisting of rows and columns. A row of a table represents a record and a column of a table represents an attribute or field. We will learn about how to create the tables to store the information, insert the information into the table, and retrieve the information from the table use by SQL statements.

SQL statements are broadly divided into two categories. Here we are giving a brief overview of each of these categories.

1. Data Manipulation Language (DML) statements
2. Data Definition Language (DDL) statements

1. Data Manipulation Language (DML) statements:

Data Manipulation Language statements are used to manipulate the data from table. These are the most frequently used commands. These statements are used for insert/ delete records in the table, to perform queries on the table to see its contents, changing data values in the existing rows of the table.

Main DML statements are
SELECT (to retrieve data from an existing table)
INSERT (to insert rows in the existing table)
DELETE (to delete rows from an existing table)
UPDATE (to update records of an existing table)

(i) Select SQL statement:

The **select** statement is used to query the database and retrieve selected data that match the criteria that you specify. Here is the format of a simple select statement:

Select [distinct column1] [*] column1[, column2, etc] from TableName [where condition(s)] [order by column1 asc or desc] [group by column1] [having condition(s)]
Note : []- optional

Example: Table name – employee
Fields - name, empno ,salary and age

- SELECT * from employee
- SELECT name, salary from employee
- SELECT * from employee where age = 30
- SELECT * from employee where age in (35,50,60,25)
- SELECT * from employee where age not in (35,50,60,25)
- SELECT * from employee where age <> 30 ORDER BY RAND() limit 1
- SELECT * from employee where age >= 30
- SELECT * from employee where age = 30 or age = 40
- SELECT * from employee where age>=30 and age <= 40
- SELECT * from employee where salary between 15000 and 40000
- SELECT * from employee where salary not between 15000 and 40000
- SELECT * FROM sp WHERE qty IS NULL
- SELECT * FROM sp WHERE qty IS NOT NULL
- SELECT P.*, B.bname FROM Products P, Brands B WHERE P.BrandID=B.ID
- SELECT uni.names FROM uni WHERE uni.names IN (SELECT fac.names FROM fac)
- SELECT student.names FROM student UNION SELECT fac.names FROM fac
- SELECT Authors.Author, Titles.Title FROM Authors, Titles WHERE Authors.BookNo = Titles.BookNo
- SELECT Authors.Author, Titles.Title FROM Authors `INNER JOIN` Titles `ON` Authors.BookNo = Titles.BookNo

Sometimes we have to select data from two or more tables to make our result complete. We have to perform a join. Tables in a database can be related to each other with keys. A primary key is a column with a unique value for each row. Each primary key value must be unique within the table. The purpose is to bind data together, across tables, without repeating all of the data in every table.

The INNER JOIN returns all rows from both tables where there is a match. If there are rows in Employees that do not have matches in Orders, those rows will **not** be listed.

The LEFT JOIN returns all the rows from the first table (Employees), even if there are no matches in the second table (Orders). If there are rows in Employees that do not have matches in Orders, those rows **also** will be listed.

SELECT Employees.Name, Orders.Product FROM Employees LEFT JOIN Orders ON Employees.Employee_ID=Orders.Employee_ID

The RIGHT JOIN returns all the rows from the second table (Orders), even if there are no matches in the first table (Employees). If there had been any rows in Orders that did not have matches in Employees, those rows **also** would have been listed.

SELECT Employees.Name, Orders.Product FROM Employees RIGHT JOIN Orders ON Employees.Employee_ID=Orders.Employee_ID

Like is a very powerful operator that allows you to select only rows that are "like" what you specify. The percent sign "%" can be used as a wild card to match any possible character that might appear before or after the characters specified.

select first, last, city from Empinfo where first LIKE 'selva%'
This SQL statement will match any first names that start with 'selva'.

select first, last from Empinfo where last LIKE '%s'
This statement will match any last names that end in a 's'.

select * from Empinfo where first = 'selva'
This will only select rows where the first name equals 'selva' exactly.

5 important aggregate functions: SUM, AVG, MAX, MIN, and COUNT. They are called aggregate functions because they summarize the results of a query, rather than listing all of the rows.

Select Sum(Salary), Avg(Salary) From Employees
Select Min(Benefits) From Employees Where Position = 'Manager'
Select Max(Benefits) From Employees Where Position = 'Manager'
Select Count(*) From Employees Where Position = 'Staff'

One special use of GROUP BY is to associate an aggregate function (especially COUNT; counting the number of rows in each group) with groups of rows.

Example, Assume that the "Orders" table has the Price column and each row has a value for that column. We want to see the price of the most expensive item bought by each owner. We have to tell SQL to group each owner's purchases, and tell us the maximum purchase price:

Select BuyerID, Max(Price) From Orders Group By BuyerID;

Now, say we only want to see the maximum purchase price if the purchase is over Rs.1000/=, so we use the HAVING clause:

Select BuyerID, Max(Price) From Orders Group By BuyerID Having Price > 1000

SQL duplicate records: select email, count(*) from users group by email order by 2 desc

ii) Insert statement: The insert statement is used to add a row of data into the table.

insert into TableName (first_column,...last_column) values (first_value,...last_value)

Example:
Insert into employees(EmpNo, Name, Age, DOB, Salary) values ('E16','Selva',28,'1980-05-25', 85000.00)

In the example above, the column name EmpNo will match up with the value 'E16', and the column name Salary will match up with the value 85000.00.

Note: All strings and dates should be enclosed between **single** quotes.

Insert into TempEmployees select * from Employees

- **INSERT INTO** Books (Author, Title) **VALUES** ('Cole, T', 'SQL')
- **INSERT INTO** Books (Author, Title) **SELECT** Authors.Author, Titles.Title **FROM** Authors, Titles **WHERE** Authors.BookNo = Titles.BookNo and Authors **LIKE** 'A%'

iii) Update statement: The update statement is used to edit record(s) that match a specified criteria. This is accomplished by carefully constructing a where clause.

UPDATE TableName set columnname1 = newvalue1 [,columnname2" = "newvalue2"...]
where columnname1 OPERATOR value [and|or column OPERATOR value];

- **UPDATE** price **SET** retail = 22.50
- **UPDATE** price **SET** retail = retail + 2.50
- **UPDATE** Books **SET** retail = retail + 10.00 **WHERE** Author = 'Cole, T'

- **UPDATE** Books, Titles **SET** Books.Titles = Titles.Title **WHERE** Books.BookNo = Titles.BookNo **AND** Books.PubDate < '1 Jan 1997'

iv) Delete statement: The delete statement is used to delete record(s) from the table. To delete an entire record/row from a table, enter "delete from" followed by the table name, followed by the where clause which contains the conditions to delete. If you leave off the where clause, all records will be deleted.

Delete from tablename where columnname OPERATOR value [and|or column OPERATOR value]

- DELETE FROM employee WHERE lastname = 'Kumar'
- DELETE FROM employee WHERE firstname = 'Siva' or lastname = 'Kumar'
- DELETE FROM Books WHERE PubDate < '1 Jan 1980'
- DELETE FROM Authors WHERE Authors.BookNo IN (SELECT Title.BookNo FROM Titles WHERE Titles.Title = 'For Whom The Bell Tolls')

Note: if you leave off the where clause, all records will be deleted. Eg: delete from Employees

2. Data Definition Language (DDL) statements:

Data Definition Language (DDL) for creating, altering, and dropping tables or fields. Main DDL statements are CREATE, ALTER and DROP.

Create statement:
For example, the following statements create a table:

```
CREATE TABLE tblproducts (
  id int(6) NOT NULL auto_increment  PRIMARY KEY,
  productname varchar(200) default '',
  categoryid int(4) default 0,
  description TEXT default '',
  price  decimal(9,2) default 0,
  ishome char(1) default '0',
  status char(1) default 'A',
  updateddate timestamp NOT NULL default CURRENT_TIMESTAMP
);
```

ALTER:
You can add new columns to an existing table using this ALTER TABLE syntax:

ALTER TABLE employee ADD column_name data_type [, ADD column_name data_type ...]
You can delete existing columns from a table using the following ALTER TABLE syntax:
ALTER TABLE table DROP column_name [, DROP column_name ...]

ADD and DROP operations can be combined in a single statement.

For example, the following statement drops two columns and adds one:
ALTER TABLE employee DROP LAST_NAME, DROP FIRST_NAME, ADD FULL_NAME varchar[50]

ALTER TABLE employee AUTO_INCREMENT = 10000;

DROP:
DROP TABLE Employee- delete a table Employee (data & table structure)

ALTER TABLE Employee DROP LAST_NAME, DROP FIRST_NAME
ALTER TABLE adminusers CHANGE firstname fname VARCHAR(80);