SQL SERVER

www.ukiinfotech.co.in

Santhanathapuram 7th street,

, Pudukkottai-600119

✉ukinfotechpdk@gmail.com

✆+91 909596967455

1.0.SQL SERVER

1.1.INTROOUCTION

SQL is a language which is used to interact with relational database management system.

1.2. DATABASE

***Database** is a organized collection of data. For example a database of a college would be having a collection of data*

1.3 Database Management System

A database management system is a software application which is used for managing different databases. It helps us to create and manage database. With the help of DBMS we take care following tasks

- *Data Security*
- *Data Backup*
- *Manages huge amount of data*
- *Data export & import*
- *Serving multiple concurrent database requests*
- *Gives us a way to manage the data using programming languages.*

1.4 Types of Databases

There are two types of databases –

- *Relational Database*
- *Non-relational Database*

1.5 Non-relational databases:

Data is **not** organized in form of tables. Data is stored in form of key & value pairs. The examples of non-relational databases are: JSON & XML.

We cannot interact with non-relational databases using SQL.

1.6 Relational Databases:

In relational database, data is organized in form of tables. A table contains rows and columns of data. Table has a unique key to identify each row of the table.

SQL is used to interact with relational databases. We often refer relational database as SQL database

2. SQL

SQL stands for Structured Query Language, which is a standardised language for interacting with RDBMS (Relational Database Management System). Some of the popular relational database example are: MySQL, Oracle, mariaDB, postgreSQL etc.

SQL is used to perform C.R.U.D (Create, Retrieve, Update & Delete) operations on relational databases.

SQL can also perform administrative tasks on database such as database security, backup, user management etc

We can create databases and tables using SQL.

2.1Types of Structured Query Language

2.2.1 DDL (Data Definition Language)

DDL is used to define table schemas. DDL commands are: CREATE, DROP and ALTER.

2.2.2DML (Data Manipulation Language)

DML is used for inserting, updating and deleting data from the database. DML commands are: INSERT, DELETE and UPDATE.

2.2.3 DQL (Data query Language)

DQL is a concept to fetch the data from the database in a table by using select statement .

2.2 What is a Query? & its protocols

A Query is a set of instruction given to the database management system, which tells RDBMS what information you would like to get, insert, update or delete from the database

Note: Do not worry about learning the queries now, this is just an introduction. We will learn how to write queries in detail in the upcoming notes .

2.3SQL Statements

SQL statement tells the database that what information you would like to retrieve or what operation you want to perform on the data.

For example:
Consider this table: STUDENT

```
STUDENT_NAME    STU_AGE    STU_ADDRESS
------------    -------    ------------
Ajeet          30        Chennai
Chaitanya        31        Noida
Steve          29        Agra
Rahul           30         Gurgaon
```

SQL statement to fetch STUDENT_NAME from the table STUDENT:

```
SELECT STUDENT_NAME FROM STUDENT;
```

To fetch the complete table:

```
SELECT * FROM STUDENT;
```

2.4 SQL is NOT case sensitive

SQL is not a case sensitive language. For example: Both the following statements would work fine and produce the same output:

```
select * from student;
```

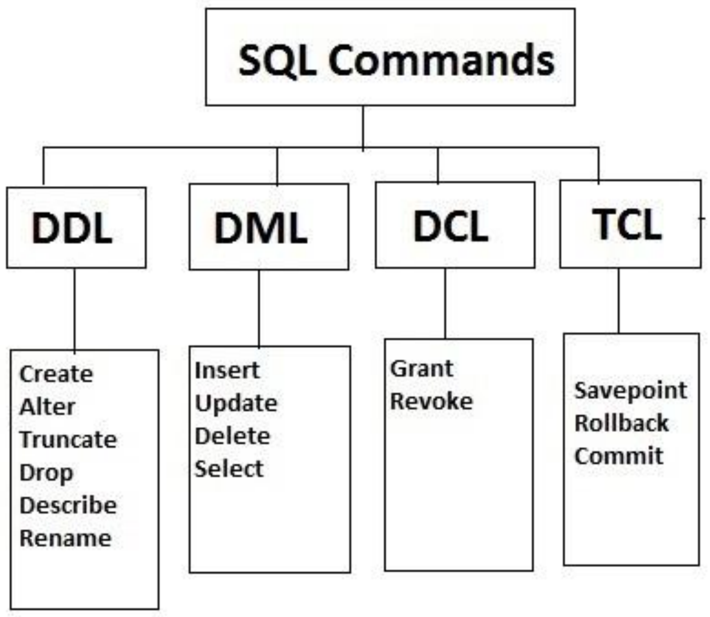
&

```
SELECT * FROM STUDENT;
```

2.5 Semicolon

As we have seen above, we have to end the SQL statement with a semi colon, it tells the RDBMS that this is a complete SQL statement. We can write more than one SQL statements together but we have to end each one of them with semicolon so that the database management system knows that they are different SQL statements. This way RDBMS can serve more than one SQL queries in a single database call.

3. SQL Commands



3.1 DDL COMMANDS

DDL statements are used to alter/modify a database or table structure and schema. These statements handle the design and storage of database objects.

***CREATE** – create a new Table, database, schema*

***ALTER** – alter existing table, column description*

***DROP** – delete existing objects from database*

```
Create database sample
Use sample
Create table stud_info(id int,name varchar(20),addr varchar(100))

Alter table stud_info add phno bigint, dept varchar(20),salary int, pin int

Alter table stud_info drop column pin

Drop table stud_info
Drop database sample
```

3.3 DML

DML statements affect records in a table. These are basic operations we perform on data such as selecting a few records from a table, inserting new records, deleting unnecessary records, and updating/modifying existing records. DML statements include the following:

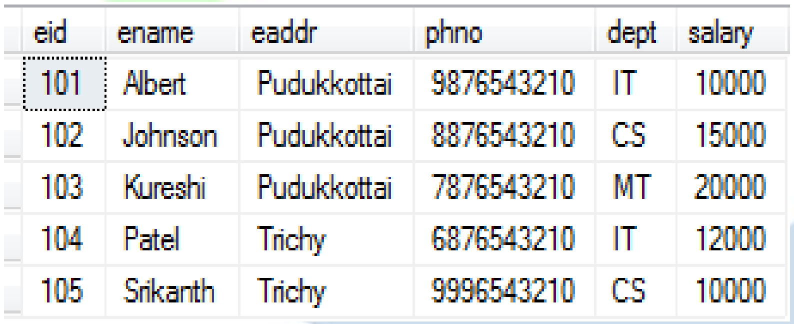
```
create database jjc
use jjc
create table emp_info(eid int,ename varchar(20),eaddr varchar(100),phno bigint, dept varchar(20),salary int)

insert into emp_info values(101,'Albert','Pudukkottai',9876543210,'IT',10000)

insert into emp_info values(102,'Johnson','Pudukkottai',8876543210,'CS',15000)

insert into emp_info values(103,'Kureshi','Pudukkottai',7876543210,'MT',20000)

insert into emp_info values(104,'Patel','Trichy',6876543210,'IT',12000)

insert into emp_info values(105,'Srikanth','Trichy',9996543210,'CS',10000)
update emp_info set ename='singh' where eid=104
delete from emp_info where eid=105
```

3.4 DQL

SELECT *– select records from a table*

```
select * from emp_info
```

OUTPUT :



eid	ename	eaddr	phno	dept	salary
101	Albert	Pudukkottai	9876543210	IT	10000
102	Johnson	Pudukkottai	8876543210	CS	15000
103	Kureshi	Pudukkottai	7876543210	MT	20000
104	Patel	Trichy	6876543210	IT	12000
105	Srikanth	Trichy	9996543210	CS	10000

4. Data Types

4.1 SQL Server String Data Type

char(n)	It is a fixed width character string data type. Its size can be up to 8000 characters.
varchar(n)	It is a variable width character string data type. Its size can be up to 8000 characters.
varchar(max)	It is a variable width character string data types. Its size can be up to 1,073,741,824 characters.
text	It is a variable width character string data type. Its size can be up to 2GB of text data.
nchar	It is a fixed width Unicode string data type. Its size can be up to 4000 characters.
nvarchar	It is a variable width Unicode string data type. Its size can be up to 4000 characters.
ntext	It is a variable width Unicode string data type. Its size can be up to 2GB of text data.
binary(n)	It is a fixed width Binary string data type. Its size can be up to 8000 bytes.
varbinary	It is a variable width Binary string data type. Its size can be up to 8000 bytes.
image	It is also a variable width Binary string data type. Its size can be up to 2GB.

4.2 SQL Server Numeric Data Types

bit	It is an integer that can be 0, 1 or null.
tinyint	It allows whole numbers from 0 to 255.
Smallint	It allows whole numbers between -32,768 and 32,767.
Int	It allows whole numbers between -2,147,483,648 and 2,147,483,647.
bigint	It allows whole numbers between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807.
float(n)	It is used to specify floating precision number data from -1.79E+308 to 1.79E+308. The n parameter indicates whether the field should hold the 4 or 8 bytes. Default value of n is 53.
real	It is a floating precision number data from -3.40E+38 to 3.40E+38.
money	It is used to specify monetary data from -922,337,233,685,477.5808 to

	922,337,203,685,477.5807.

4.3 SQL Server Date and Time Data Type

datetime	It is used to specify date and time combination. It supports range from January 1, 1753, to December 31, 9999 with an accuracy of 3.33 milliseconds.
datetime2	It is used to specify date and time combination. It supports range from January 1, 0001 to December 31, 9999 with an accuracy of 100 nanoseconds
date	It is used to store date only. It supports range from January 1, 0001 to December 31, 9999
time	It stores time only to an accuracy of 100 nanoseconds
timestamp	It stores a unique number when a new row gets created or modified. The time stamp value is based upon an internal clock and does not correspond to real time. Each table may contain only one-time stamp variable.

5. AGGREGATE FUNCTIONS

An aggregate function performs a mathematical calculation one or more values and returns a single value. The aggregate function is often used with the GROUP BY clause and HAVING clause of the SELECT statement.

```
select MAX(salary) from emp_info
select MIN(salary) from emp_info
select SUM(salary) from emp_info
select AVG(salary) from emp_info
select COUNT(eid) from emp_info
select COUNT(*) from emp_info
```

6. CONSTRAINTS

Constraints are the rules that we can apply on the type of data in a table. That is, we can specify the limit on the type of data that can be stored in a particular column in a table using constraints.

- ***NULL*** **–** It allows empty values
- ***NOT NULL****:-- doesn't allow empty values*
- ***UNIQUE –*** *Doesn't allow duplicate values but allow null values*
- ***PRIMARY KEY –*** *Doesn't allow duplicate values not null*
- ***FOREIGN KEY –*** *Reference of another table primary key*
- ***CHECK****:- Specify the condition to column*
- ***DEFAULT*** *– Set default value to the column*

CONSTRAINTS SYNTAX :

```
CREATE TABLE sample_table
(
column1 data_type(size) constraint_name,
column2 data_type(size) constraint_name,
column3 data_type(size) constraint_name,
....
);
```

6.1 Query example

NULL :

```
create table t1(rno int NULL,name varchar(20))
```

<u>NOT NULL</u>

```
create table t2(rno int NOT NULL,name varchar(20))
```

<u>Primary Key</u>

```
create table t3(rno int,name varchar(20) not null,CONSTRAINT t3c primary key(rno))
```

<u>Foreign Key</u>

```
create table t4(eid int,addr varchar(200) not null,CONSTRAINT t4c foreign key(eid) references t3(rno))
```

<u>Unique</u>

```
create table t5(rno int,name varchar(20) not null,CONSTRAINT t5c unique(rno))
```

Check

```
create table t6(rno int,name varchar(20) not null,age int,CONSTRAINT t6c check(age>18))
```

Default

```
create table t7(rno int,name varchar(20) not null,city varchar(20) default 'Pudukkottai')
```

6.2 KEYS IN SQL :

Alternate Key- *combination of unique & notnull*

```
create table t8(rno int not null,name varchar(20) not null,CONSTRAINT t8c unique(rno))
```

Candidate Key – *combination of primarykey & alternate key*

```
create table t9(rno int,regno int not null, name varchar(20) not null,CONSTRAINT t9c primary key(rno),constraint t9c2 unique(regno))
```

Composite Key- *more than one aalternate key*

```
create table t10(rno int not null,regno int not null, name varchar(20) not null,CONSTRAINT t10c unique(rno),constraint t10c2 unique(regno))
```

7. operators

Operators are used to specify conditions in an SQL statement and to serve as conjunctions for multiple conditions in a statement. Types of operators :

- *Arithmetic operators*
- *Comparison operators*
- *Logical operators*
- *Operators used to negate conditions*
- *Wild card operaators*

7.1 SQL Arithmetic Operators:

Operator	Meaning	Operates

		on
+ (Add)	Addition	Numeric value
- (Subtract)	Subtraction	Numeric value
*** (Multiply)**	Multiplication	Numeric value
/ (Divide)	Division	Numeric value
% (Modulos)	Returns the integer remainder of a division. For example, 17 % 5 = 2 because the remainder of 17 divided by 5 is 2.	Numeric value

7.2 SQL Comparison Operators:

Operator	***Description***
=	*Equal to*
>	*Greater than*
<	*Less than*
>=	*Greater than or equal to*
<=	*Less than or equal to*
<>	*Not equal to*

7.3 Logical Operator

Logical Operator	Function
AND	Both the conditions mentioned in the WHERE clause should be TRUE.
OR	At least one of the conditions mentioned in the WHERE clause should be TRUE.
NOT	The mentioned condition should be false in the WHERE clause.

Example Comparison Operators:

```
select * from emp_info where salary=10000
select * from emp_info where salary>10000
select * from emp_info where salary<15000
select * from emp_info where salary<>10000
select * from emp_info where salary=10000 and dept='cs'
select * from emp_info where salary=10000 or dept='cs'
```

7.4 clause operator

ITS just use write specific expecting something in form of qquqestion in an query

WHERE CLAUSE

```
select * from emp_info where salary=10000 and dept='cs'
select * from emp_info where salary=10000 or dept='cs'
```

IN

```
select * from emp_info where eid in(101,103,105)
```

BETWEEN

```
select * from emp_info where eid between 103 and 105
```

7.5 WILD CARD OPERATOR

- Wildcards are special placeholder characters that can represent one or more other characters or values.
- Like many computer languages, SQL allows the use of various wildcard characters.
- Wildcards allow us to search the database for any data without knowing the exact values held within it

```
select * from emp_info where ename like 'a%'
select * from emp_info where ename like '%h'
select * from emp_info where ename like '[s-z]%'
select * from emp_info where ename like '%[a-j]'
select * from emp_info where ename like 's_n_h'
select * from emp_info where ename like '[^a-c]%'
select * from emp_info where ename not like '[a-c]%'
```

GROUP BY METHOD IN SQL SERVER

***G**ROUP BY Statement in SQL is used to arrange identical data into groups with the help of some functions. i.e if a particular column has same values in different rows then it will arrange these rows in a group.*

- GROUP BY clause is used with the SELECT statement.
- In the query, GROUP BY clause is placed after the WHERE clause.
- In the query, GROUP BY clause is placed before ORDER BY clause if used any.
- In the query , Group BY clause is placed before Having clause .
- Place condition in the having clause

Table creation :

```
create table Productsale (ProductId char(1) , customerID int , SaleQty int);
Insert Into Productsale values('A',101, 10 );
Insert Into Productsale values('A',44, 2 );
Insert Into Productsale values('A',3, 1 );
Insert Into Productsale values('B',44, 3 );
Insert Into Productsale values('C',3, 5) ;
Insert Into Productsale values('C',4, 1 );
Insert Into Productsale values('D',124, 6 );
Insert Into Productsale values('A',129, 98 );
```

Group by queries :

```
Select ProductId From ProductSale
Group By productId


Select ProductID , [Sale Quantity]=Sum(SaleQty)
 From ProductSale Group By ProductID
```

```
Select ProductID , [Sale Quantity]=Sum(SaleQty)
 From ProductSale Group By ProductID
```

```
Select ProductID , [Max Quantity]=MAX(SaleQty)
 From ProductSale Group By ProductID
```

8. SUBQUERIES & JOIN

8.1 SUBQUERIES :

Query with in a query is called sub query.

```
select * from emp_info where salary=(select MAX(salary) from emp_info)
select * from emp_info where salary=(select MAX(salary) from emp_info where dept='IT')
```

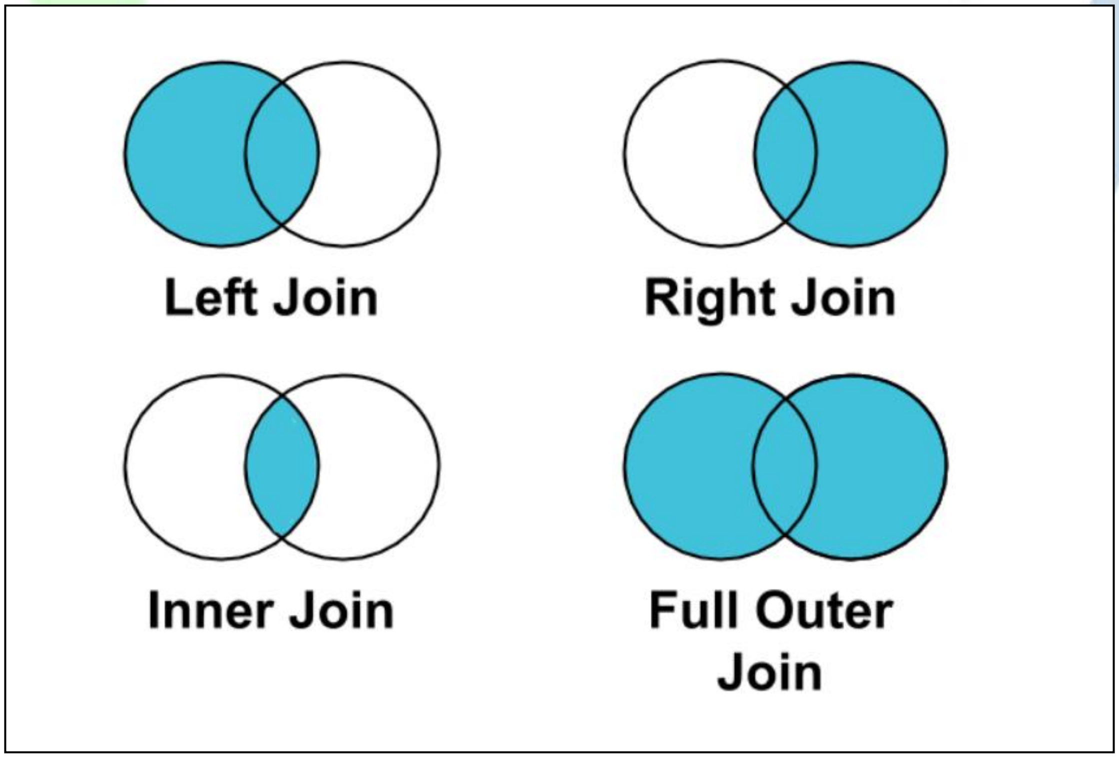
8.2 SQL Join Statement

In This topic we'll discuss the JOIN section of the SQL statement. We will cover its syntax and the different types of joins that SQL enables.

8.2.1 SQL syntax with focus on Join

```
SELECT col1, col2, col3, etc....
FROM  tableNameOne AS a
JOIN tableNameTwo AS b ON a.primeKey = b.primeKey
etc.
```

IN JOIN statement could be just JOIN or INNER JOIN, which are the same



8.2.2 Different Types of JOINs

- (INNER) JOIN - Return records that have matching values in both tables
    - LEFT (OUTER) JOIN
    - Return all records from the left table, and the matched records from the right table
    - RIGHT (OUTER) JOIN - Return all records from the right table, and the matched records from the left table
    - FULL (OUTER) JOIN - Return all records when there is a match in either left or right table

8.2.3 Discussion about join

In below topic we would join queries with example

```
create table emp_info(eid int,ename varchar(20),eaddr varchar(100),phno bigint, dept
varchar(20),salary int)

insert into emp_info values(101,'Albert','Pudukkottai',9876543210,'IT',10000)
insert into emp_info values(102,'Johnson','Pudukkottai',8876543210,'CS',15000)
insert into emp_info values(103,'Kureshi','Pudukkottai',7876543210,'MT',20000)
insert into emp_info values(104,'Patel','Trichy',6876543210,'IT',12000)
insert into emp_info values(105,'Srikanth','Trichy',9996543210,'CS',10000)
```

Table 2 :

```
create table emp_acc(eid int,designation varchar(20),w_hrs int)
insert into emp_acc values(101,'developer',8)
insert into emp_acc values(102,'executive',10)
insert into emp_acc values(103,'developer',8)
```

Lets we work join query by using above table

```
select * from emp_info inner join emp_acc on emp_info.eid=emp_acc.eid
select * from emp_info full outer join emp_acc on emp_info.eid=emp_acc.eid
select * from emp_info left outer join emp_acc on emp_info.eid=emp_acc.eid
select * from emp_info right outer join emp_acc on emp_info.eid=emp_acc.eid
select * from emp_info cross join emp_acc
```

VIEW

Views are the virtual tables consisting of columns and rows from the referenced table.

Note : Mirror form of table

SYNTAX for view :

```
Create view  viewname
As
 Select * from tablename
```

Query for view

```
Create view  v1
As
 Select * from emp_info
```

Here after we can use v1 as table name

***Note :** DDL Command not processed*
Only
DML Command can proceed