SQL FUNDAMENTALS

- Structured Query Language (SQL) is a database query language used create, access, and manipulate data and structure in Relational DBMS.
- SQL was the first commercial language introduced for **E.F Codd**'s Relational model of database.
- **Examples of RDBMS:** MySQL, Oracle, MS SQL Server, MS Access etc.,

SQL Sub languages:

1. **Data-Definition language (DDL)**
2. **Data-Manipulation Language (DML)**
3. **Transaction Control Language (TCL)**
4. **Data Control Language (DCL)**

Basic Datatypes in SQL

- **char(n)**: Fixed length character string, with user-specified length '*n'.*
- **varchar(n):** Variable length character strings, with user-specified maximum length *n.*
- **int**:
- **numeric(p,d):** Fixed point number, with user-specified precision of ***'p'*** digit number, with ***'d'*** digits to the right of decimal point.
- **float(n):** Floating point number, with user-specified precision of at least *n* digits.
- **date :** represents date in the format of 'yyyy-mm-dd'
- **time:** represents time in the format of 'hh:mm:ss'
- **timestamp:** represents a combination of date and time in the format of 'yyyy-mm-dd hh:mm:ss'

DATA-DEFINITION LANGUAGE (DDL)

Data Definition Language (DDL) statements are used to define and modify the database structure or schema. In **MYSQL and ORACLE,** the DDL Commands are auto committed.
Some examples:

- CREATE - to create database objects
- ALTER - alters the structure of the database
- DROP - delete objects from the database
- TRUNCATE - remove all records from a table ,but, the table still exists
- RENAME - rename an object

DML

Data Manipulation Language (DML) statements are used for managing data within schema objects. In **MYSQL**, DML commands are auto committed. In **ORACLE**, DML commands are not auto committed.
Some examples:

- SELECT - retrieve data from the a database
- INSERT - insert data into a table
- UPDATE - updates existing data within a table
- DELETE - deletes all records from a table, the space for the records remain

DCL

Data Control Language (DCL) statements. Some examples:

- GRANT - gives user's access privileges to database
- REVOKE - withdraw access privileges given with the GRANT command

TCL

Transaction Control (TCL) statements are used to manage the changes made by DML statements. It allows statements to be grouped together into logical transactions.

- COMMIT - save work done
- SAVEPOINT - identify a point in a transaction to which you can later roll back
- ROLLBACK - restore database to original since the last COMMIT
- SET TRANSACTION - Change transaction options like isolation level and what rollback segment to use

---

DATA DEFINITION IN SQL

Creating Tables:

The SQL command for creating an empty table has the following form:

Syntax:

create table <table-name> (<column 1> <data type> ,<column2t> <data type>, . . . , .<column n> <data type>, <integrity constraint1>,…., <integrity constraint n>);

Example:

The create table statement for EMP table has the form

create table EMP (**empno** int, **ename** varchar(30), **salary** float(7,2));

Deleting tables:

The following command will drop the table from the database.

Syntax:

```
drop table <table-name>;
```

Example:

```
drop table emp;
```

Alter table:

Alter table command is used to add, modify or drop attributes or columns from the table.

Syntax:

```
alter table <table-name> add/modify <col-name1> <datat ype>,….,<col-namen> <data type>;

alter table <table-name> drop <col-nmae>;
```

Example:

```
alter table emp add year varchar(3);          //adds a column to the table

alter table emp modify year varchar(4);       // modify the column name

alter table emp drop year                     //delete/drop a column from the table
```

Truncate table:

The truncate table command deletes the rows in the table but retains the table structure.

Syntax:

```
truncate table <table-name>;
```

Example:

truncate table emp;

DATA MANIPULATION LANGUAGE

Insert command:

Insert command is used to insert single or multiple rows in the table.

Syntax:

Single row:

insert into <table-name> values (list of values);

Example:

Single row:

insert into emp values(1000,’anitha’,10000);

Multiple rows:

insert into emp values(1001,'banu',' 20000),(1002,'anu', 25000),(1003, 'hari', 10000);

Insert values for specific columns:

insert into emp (empno, ename) values(1004,'nithya'), (1005, 'john');

Delete command:

The delete command removes tuples from a relation. It includes the where clause to select the tuples to be deleted.

Syntax:

delete from <table-name> where attribute-name = ‘value’ or value;

Example:

```
delete from emp where ename = ‘anitha’;
```

Update command:

The update command is used to modify attribute values of one or more selected tuples. It includes a where clause to select the tuples to be modified from a single relation.

Syntax:

update <table-name> **set** col1=val1,…,coln = valn **where** condition;

Example:

```
update emp

set salary = 12000

where empno =1000;
```

Select clause:

The select clause is used to select a particular or all attributes from relations.

Syntax:

select <attribute list> or (*) from <table list> where <condition>;

Example:

```
select * from emp where empno =1000;

select empno, ename from emp;
```

<u>TRANSACTION CONTROL LANGUAGE</u>

There are following commands used to control transactions:

- COMMIT: to save the changes.
- ROLLBACK: to rollback the changes.
- SAVEPOINT: creates points within groups of transactions in which to ROLLBACK
- SET TRANSACTION: Places a name on a transaction.
- Transactional control commands are only used with the DML commands INSERT, UPDATE and DELETE only.
- They cannot be used while creating tables or dropping them because these operations are automatically committed in the database.

THE COMMIT COMMAND

The COMMIT command is the transactional command used to save changes invoked by a transaction to the database.

The COMMIT command saves all transactions to the database since the last COMMIT or ROLLBACK command.

SYNTAX:COMMIT;

THE ROLLBACK COMMAND

The ROLLBACK command is the transactional command used to undo transactions that have not already been saved to the database.

The ROLLBACK command can only be used to undo transactions since the last COMMIT or ROLLBACK command was issued.

SYNTAX: ROLLBACK

THE SAVEPOINT COMMAND

A SAVEPOINT is a point in a transaction when you can roll the transaction back to a certain point without rolling back the entire transaction.

SYNTAX: SAVEPOINT SAVEPOINT_NAME;

This command serves only in the creation of a SAVEPOINT among transactional statements. The ROLLBACK command is used to undo a group of transactions.

SYNTAX: ROLLBACK TO SAVEPOINT_NAME;

The RELEASE SAVEPOINT Command

The RELEASE SAVEPOINT command is used to remove a SAVEPOINT that you have created.

Syntax: RELEASE SAVEPOINT SAVEPOINT_NAME;

Once a SAVEPOINT has been released, you can no longer use the ROLLBACK command to undo transactions performed since the SAVEPOINT.

Example

create table student(id int, name varchar(30), city varchar(20));

start transaction;

insert into student values(101,'arun','chennai'), (102, 'adhi','nellore');

select * from student;

```
id    name     city
101   arun     chennai
102   adhi     nellore
```

savepoint s;

insert into student values(103,'aron','chennai'), (104, 'john','trichy');

select * from student;

```
id    name     city
101   arun     chennai
102   adhi     nellore
```

103	aron	chennai
104	john	trichy

rollback to savepoint s;

select * from student;

id	name	city
101	arun	chennai
102	adhi	nellore

<u>DATA CONTROL LANGUAGE:</u>

1.Command: GRANT

Syntax:

GRANT permissions ON objects TO account

Example Query 1: GRANT INSERT ON employee TO PUBLIC;

Example Query 2: GRANT SELECT,UPDATE ON employee to username;

2.Command: REVOKE

Syntax:

REVOKE permissions ON object FROM account

Example Query:

REVOKE SELECT ON student FROM username;

BUILT IN FUNCTIONS:

- DUAL is a table automatically created by **Oracle Database** along with the data dictionary.

Character functions

mysql> select initcap(‘xyz’) from dual;

mysql> select lower(‘xyZ’) from dual;l8io

mysql> select upper(‘xyz’) from dual;

mysql> select ascii(‘a’) from dual;

mysql> select rtrim(‘xyzabc’,’abc’) from dual;

mysql> select ltrim(‘xyzabc’,’xyz’) from dual;

mysql> select translate(‘rmk’,’k’,’d’) from dual;

mysql> select lpad(‘rmd’,8,’*’) from dual;

mysql> select rpad(‘rmd’,8,’*’) from dual;

mysql> select length(‘rmd’) from dual;

Numeric Functions

mysql> select abs(34) from dual;

mysql> select abs(-28) from dual;

mysql> select ceil(44.56) from dual;

mysql> select floor(33.48) from dual;

mysql> select power(4,2) from dual;

```
mysql> select mod(10,3) from dual;

mysql> select sqrt(25) from dual;

mysql> select trunc(100.43) from dual;

mysql> select round(100.256,1) from dual;

mysql> select sign(-34) from dual;

mysql> select sign(34) from dual;
```

Date functions

```
mysql> select to_char(sysdate,’dd-mm-yyyy’) from dual;

mysql> select add_months(sysdate,9) from dual;

mysql> select months_between(’07-aug-05’,sysdate) from dual;

mysql> select greatest(’17-jun-00’,sysdate) from dual;

mysql> select next_day(’03-jun-08’,’wed’) from dual;

mysql> Select last_day(sysdate) from dual;

mysql> select trunc(to_date(‘03-jun-08’),’year’) from dual;

mysql> select last_day(sysdate) from dual;
```

CREATING TABLES TO SET VARIOUS CONSTRAINTS

CONSTRAINTS

Constraints are part of the table definition that limits and restriction on the value entered into its columns.

TYPES OF CONSTRAINTS:

1) Primary key
2) Foreign key/references(Integrity constraint)
3) Check
4) Unique
5) Not null
6) Default

OPERATION ON CONSTRAINT:

i) ENABLE
ii) DISABLE
iii) DROP

QUERIES TO INCLUDE CONSTRAINTS

PRIMARY KEY CONSTRAINTS:

- A primary key is a unique field of the table
- It **does not allow duplicate and null values**.

mysql>create table employee(eno int **primary key**, ename varchar(10), salary int, address varchar(35), deptno int);

Query OK, 0 rows affected

mysql>**insert into employee values(1001,’lalith’,50000,’chennai’,1),(1002,’lalitha’,55000,’chennai’,2);**

Query OK, 2 rows affected

mysql> select *from customer;

```
ENO      ENAME   SALARY CITY

--------- -----------------------------------

1001   lalith      50000          chennai

1002   lalitha     55000          chennai
```

2 rows in set

insert into employee values(1001,'Kavi',65000,'chennai',2);

ERROR 1062 (23000) at line 11: Duplicate entry '1001' for key 'employee.PRIMARY'

insert into employee values(null,'jahnavi',60000,'AP', 2);

ERROR 1048 (23000) at line 8: Column 'eno' cannot be null

UNIQUE KEY CONSTRAINTS

- A unique key does not allow duplicate values, but it accepts null values.
- A table can have more than one unique key

mysql> create table euniq(eno int **unique**, ename varchar(6));

Query OK, 0 rows affected

mysql> insert into euniq values(1001, ‘abitha’),(1002,’banu’);

Query OK, 2 rows affected

mysql> SELECT *FROM euniq;

```
ENO         ENAME
---------------------
1001            abitha
1002              banu
2 rows in set
```

insert into euniq values(1001, 'janani');

```
ERROR 1062 (23000) at line 9: Duplicate entry '1001' for key 'euniq.eno'
```

insert into euniq values(null, 'chitra');

select * from euniq;

```
eno    ename
----------------
1001   abitha
1002   banu
NULL   chitra
```

DEFAULT CONSTRAINTS

mysql> create table emp(num int, name varchar(20), bonus int **default** 5000);

mysql> **insert into emp values(101,'dev',10000);**

mysql> **insert into emp(num,name) values(102,'abi');**

mysql> select * from emp;

```
| num  | name  | bonus |
-----------------------------
101     dev      10000
102     abi       5000
```

CHECK CONSTRAINTS:

- Check constraint is used to check the values for specific attribute at the time of insertion. If check constraint is violated, the record is not allowed to be inserted.
- **MYSQL does not support CHECK constraints. They are just ignored.**

```
mysql> create table customercheck(cusno int,cusname varchar(30), sal int , check(sal>5000));

Query OK, 0 rows affected

mysql> insert into customercheck values(101,'karthick',45000),(102,'aruna',60000);

Query OK, 2 rows affected

mysql> select *from customercheck;

CUSNO CUSNAME          SAL
--------- -------------------------
101     karthick         45000
102     aruna            60000
```

NOT NULL CONSTRAINTS:

- Does not allow the attribute value to be null

```
mysql>create table customernull (cusno int, cusname varchar(5),cusal int not null, cusph int)
```

```
Query OK, 0 rows affected

mysql>insert into customernull values(1,'a' ,40000,123456),(2,’b’,60000,null);

Query OK, 2 rows affected

mysql>insert into customernull values(3,’c’,null,32456);

Error: cannot insert NULL into (3.”c”."CUSTOMERNULL"."CUSAL")

mysql> select *from customernull;

 CUSNO CUSNAME    CUSAL     CUSPH

---------- ----------------------------------------------

      1          a        40000          123456

      2          b        60000          32456
```

FOREIGN KEY CONSTRAINTS:

Creation of Parent table with primary key

```
Mysql > create table first(rno int primary key,name varchar(20), marks int);
      Query OK, 0 rows affected (0.03 sec)
```

Creation of child table with foreign key

```
Mysql > create table second(regno int, contact int, foreign key(regno) references
first(rno));

Query OK, 0 rows affected (0.07 sec)
```

Inserting records in parent table

```
Mysql > insert into first values(101,'vini',99),(102,'arun', '89'),(103,'kabi',100);
       Query OK, 3 rows affected (0.02 sec)
   Records: 3  Duplicates: 0  Warnings: 0
```

```
Mysql > select * from first;

+------+------

| rno | name | marks |

+-----+------+------

| 101 | vini |    99 |

| 102 | arun |    89 |

 | 103 | kabi |   100 |

 +-----+------+-------

3 rows in set (0.02 sec)
```

Inserting records in child table

```
Mysql > insert into second values(105,895478);
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (``.`s
econd`, CONSTRAINT `second_ibfk_1` FOREIGN KEY (`regno`) REFERENCES `first` (`rno`))


Mysql > insert into second values(101,895478),(102,4743135);
    Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0


Mysql > select * from second;

+-------+---------

| regno | contact |

+-------+---------

|   101 |  895478 |
|   102 | 4743135 |

+-------+---------
```

Deleting a record form child table

```
Mysql > delete from second where regno=102;
Query OK, 1 row affected (0.00 sec)
```

Deleting a record form parent table

```
mysql > delete from first where rno=101;


ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (
``.`second`, CONSTRAINT `second_ibfk_1` FOREIGN KEY (`regno`) REFERENCES `first` (`rno`)
)       mysql >select * from first;
+-----+------+-------

| rno | name | marks |

+-----+------+-------
| 101 | vini |    99 |

| 102 | arun |    89 |

| 103 | kabi |   100 |

3 rows in set (0.00 sec)

mysql> select * from second;
  | regno | contact |

  |   101 |  895478 |
```

CREATION OF VIEWS

<u>VIEWS In SQL</u>

- A view is a virtual table based on the result-set of an SQL statement.
- A view contains rows and columns, just like a real table.
- The fields in a view are fields from one or more real tables in the database.

They are generally used to avoid duplication of data. Views are created for the following reasons,

- Data simplicity
- To provide data security
- Structural simplicity (because view contains only limited number of rows and columns)

SQL CREATE VIEW SYNTAX

CREATE VIEW view_name AS SELECT column_name(s) FROM table_name WHERE condition

EXAMPLE:

```
Mysql>select * from employee;
+------+-------+--------+
| eno  | name  | salary |
+------+-------+--------+
|  101 | kabil |  40000 |
|  102 | nitin |  38000 |
+------+-------+--------+
Mysql>create view empview as(select * from employee where salary>39000);

Query OK, 0 rows affected (0.02 sec)
```

```
Mysql>select * from empview;
+------+-------+------
| eno  | name  | salary |
+------+-------+--------
|  101 | kabil |  40000 |
+------+-------+--------
1 row in set (0.01 sec)
Mysql>insert into empview values(102,'pri',35000);

Query OK, 1 row affected (0.01 sec

Mysql>select * from employee;
+------+-------+--------
| eno  | name  | salary |
+------+-------+--------+
|  101 | kabil |  40000 |
|  102 | nitin |  38000 |
|  102 | pri   |  35000 |
+------+-------+--------+
3 rows in set (0.00 sec)
```

INSERT STATEMENT SYNTAX:

```
mysql> insert into (column name1, …) values(value1,….);
```

EXAMPLE:

```
insert into empview values ('sri', 120,'cse', 67,'16-nov-1981');
```

SQL UPDATING A VIEW SYNTAX

```
update view name set field name=value where condition
```

EXAMPLE:

update empview set employee_name='kavi' where employee_name='ravi';

SQL DROP VIEW SYNTAX

drop view view-name;

<u>INDEXES</u>

Indexes are special lookup tables that the database search engine can use to speed up data retrieval.

An index is a pointer to data in a table. An index in a database is very similar to an index in the back of a book.

When an index is created, it first sorts the data and then it assigns a ROWID for each row.

- An index can be created in a table to find data more quickly and efficiently.
- The users cannot see the indexes, they are just used to speed up searches/queries

SYNTAX TO CREATE INDEX

CREATE INDEX index_name ON table_name (column_name1,column_name2...);

EX:

mysql> create index myCompIndex on student_tbl(DOB,ADDRESS);

SYNTAX TO CREATE SQL UNIQUE INDEX

CREATE UNIQUE INDEX index_name ON table_name (column_name1, column_name2...);

EXAMPLE:

mysql> create unique index myIndex on student_tbl(name);

- index_name is the name of the INDEX.

- table_name is the name of the table to which the indexed column belongs.
- column_name1, column_name2..is the list of columns which make up the INDEX.

THE DROP INDEX COMMAND

An index can be dropped using SQL DROP command. Care should be taken when dropping an index because performance may be slowed or improved.

SYNTAX:

```
DROP INDEX index_name;
```

EXAMPLE:

```
drop index index_of_student_tbl;
```