SQL- Structured query language

Row is called as “record”

Each block are called as “cells” or “fields”

Unique value: can be more than one

Primary value: only one

**DDL** Data Definition Language used to create modify struture of database object

Create Alter Truncate Drop Rename

**DML** Data Manipuulation Language used to retrive store modify delete data in database

insert update delete select

**DCL** Data Coontrol Language used to create roles permission as well it is used to control access to database by securiing it

Grant Revoke, Revoke

**TCL** Transactoin Control Language used to manage different transaction occuting

Commit, Rollback, savepoint

**Mysql commandline:**

mysql –u root –p

show databases;

use databasename;

show tables;

**Mysql Engines:**

InnoDB

mysam

heap ex.csv,archieve

**Queries:**

**DataBase:**

CREATE DATABASE <database\_name>;

CREATE DATABASE rajesh\_database;

DROP DATABASE <database\_name>;

**Table:**

CREATE TABLE <table\_name>(<column 1><data\_type1>,<column 2><data\_type2>,);

CREATE TABLE employee\_tbl (emp\_id integer not null primary key, emp\_name varchar(20));

CREATE VIEW <view\_name> as <query>

CREATE VIEW v\_name as select\* from employee\_tbl;

//it will call directly employee\_tbl

ALTER VIEW <viename> AS ...query...;

DROP VIEW <viewname>;

CREATE INDEX <index\_name> on <table\_name>;

CREATE INDEX i\_name on employee\_tbl (emp\_id);

CREATE CLUSTERED INDEX <index\_name> on <table\_name>;

CREATE CLUSTERED INDEX i\_name on employee\_tbl (emp\_id)

ALTER TABLE <table\_name> ADD <name> <datatype>;

ALTER TABLE employee\_tbl ADD emp\_phone integer;

ALTER TABLE <table\_name> MODIFY <name> <datatype> AFTER <name2>;

ALTER TABLE employee\_tbl MODIFY emp\_phone integer AFTER id;

ALTER TABLE <table\_name> ALTER COLUMN <name> <datatype>;

ALTER TABLE employee\_tbl ALTER COLUMN emp\_phone varchar(10);

ALTER TABLE <table\_name> CHANGE COLUMN <old\_name> <newname><datatype>;

ALTER TABLE employee\_tbl CHANGE COLUMN emp\_phone empp\_phone varchar(10);

ALTER TABLE <table\_name> DROP COLUMN <name> <datatype>;

ALTER TABLE employee\_tbl DROP COLUMN emp\_phone integer;

ALTER TABLE <table\_name> RENAME TO <newtablename>;

ALTER TABLE employee\_tbl RENAME TO employee\_table;

TRUNCATE TABLE <tablename>;

DROP TABLE <tablename>;

DELETE (FROM)<tablename>;

RENAME TABLE <oldtablename> TO <newtablename>;

Truncate will remove data from table, Drop will remove tables and databse

**Inserting data:**

INSERT INTO <table\_name> (<column1,column2,…>) VALUES (<values1>,<values2>)

INSERT INTO employee\_tbl (emp\_id,emp\_name) VALUES (01,’rajesh’);

INSERT INTO <table\_name> <column1,column2,…> SELECT \*FROM <another\_table>

INSERT INTO employee\_tabl2 (emp\_id,emp\_name) SELECT \*FROM employee\_tbl;

INSERT INTO <table\_name> (value1, value2)

INSERT INTO employee\_tbl (02,’ ’);

INSERT INTO employee\_tbl (02,NULL); //these two are called null only

**Updating data:**

UPDATE table\_name SET column value WHERE condition

UPDATE employee\_tbl SET emp\_name ‘ganesh’,emp\_name=”vignesh” WHERE emp\_id=02,emp\_id=03;

**Deleting:**

DELETE FROM <table\_name>

DELETE FROM employee\_tbl //delete all records in table

DELETE FROM <table\_name> WHERE condition

DELETE FROM employee\_tbl WHERE id=01;

**Selecting:**

SELECT ALL <coloum\_name> FROM <table\_name>

SELECT ALL emp\_name FROM employee\_table;

**Special Selecting:**

DISTINCT

ORDER BY

GROUP BY

LIKE

NOT LIKE

IN

NOT IN

IS NULL

IS NOT NULL

BETWEEN

LIMIT

SELECT DISTINCT <column\_name> FROM <table\_name>

SELECT DISTINCT emp\_name FROM employee\_tbl; //does not show repeated values

SELECT \*FROM <table\_name> WHERE <column\_name> ORDER BY <value> ASC

SELECT \*FROM employee\_tbl WHERE emp\_id=01 ORDER BY emp\_name ASC

SELECT \*FROM <table\_name> WHERE <column\_name> ORDER BY <value> DESC

SELECT \*FROM employee\_tbl WHERE emp\_id=01 ORDER BY emp\_name DESC

SELECT <column\_name> FROM <table\_name> GROUP BY <value> //ascending sorting

SELECT salary FROM employee\_tbl GROUP BY salary,emp\_name; //not display identical

SELECT \* FROM <tablename> WHERE <column\_name> LIKE “%rajesh%”;

SELECT \* FROM <tablename> WHERE <column\_name> NOT LIKE “%rajesh%”;

SELECT \* FROM <tablename> WHERE <column\_name> IN (‘muthu’,’rajesh’);

SELECT \* FROM <tablename> WHERE <column\_name> NOT IN (‘muthu’,’rajesh’);

SELECT \* FROM <tablename> WHERE <column\_name> IS NULL;

SELECT \* FROM <tablename> WHERE <column\_name> IS NOT NULL;

SELECT \* FROM <tablename> WHERE <column\_name> BETWEEN 1 AND 3;

SELECT \* FROM <tablename> LIMIT 2;

**Datatype:**

|  |  |  |
| --- | --- | --- |
| Integer | BigInt  Int  SmallInt  TinyInt  MediumInt | 9\*1019  2\*109  -32768 to 32768  0 to 255  -8388608 to -8388608 |
| Real | Bit  Decimel  Float  Real  Double  Boolean  Serial | 0,1  1.0000  1.0000000  1  1.0000000000000  TRUE FALSE |
| String | Char  VarChar  Text  MediumText  LongText  Binary  VarBinary | 65,535 chars  16777215 chars  4,294,967,295 chars |
| Date | Date  DateTime  TimeStamp  Time  Year | YYYY-mm-dd  YYYY-mm-dd hh:mm:ss  YYYY-MM=DD HH:MM:SS  hh:mm:ss  YY or YYYY |
| Large object | TinyBlob  Blob  MediumBlob  LongText |  |
| Spactial | Geometry  Point  LinString  Polygon  MultiPoint  MultiLineString  MultiPolygon  Geometry colleation |  |

Int, VarChar,Text, Date

Numeric : TinyInt, SmallInt, MediumInt, Int, BigInt, Decimel, Float, Double, Real, Bit, Boolean, Serial

DataTime: Date, DateTime, TimeStamp, Date, Year

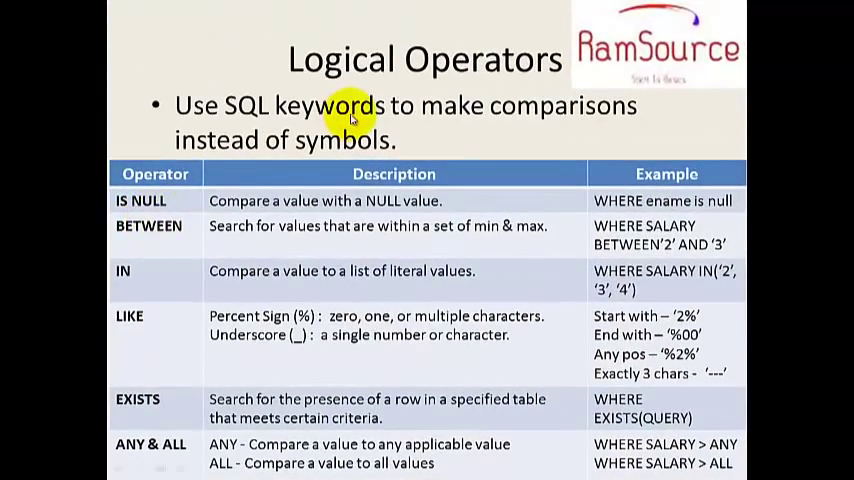
String:Char, VarChar, TinyText, Text, MediumText,LongText,Binary,VarBinary, TinyBlob, MediumBlob, TinyBlob,Blob, Enum, Set

Spactial: Geometry, Point, LinString, Polygon, MultiPoint, MultiLineString, MultiPolygon, GeometryCollection

**Operator:**

Comparision operator

|  |  |  |
| --- | --- | --- |
| = | Equality | SELECT \* FROM employee\_tbl WHERE id=1000 |
| <> | Non Equality | SELECT \* FROM employee\_tbl WHERE i<>1000 |
| < | Less than | SELECT \* FROM employee\_tbl WHERE id >1000 |
| > | Greater than | SELECT \* FROM employee\_tbl WHERE id <1000 |
| <= | Less than or Equal to | SELECT \* FROM employee\_tbl WHERE id <= 1000 |
| >= | Greater than or Equal to | SELECT \* FROM employee\_tbl WHERE id >= 1000 |
| AND |  | WHERE id<1000 AND salary > 2000 |
| OR |  | WHERE id<1000 AND salary > 2000 |

**Logical Operator**

SELECT \* FROM employee\_tbl WHERE emp\_name IS NULL;

SELECT \*FROM employee\_tbl WHERE salary BETWEEN 10 AND 20;

SELECT \*FROM employee\_tbl WHERE salary IN (10,20);

SELECT \*FROM employee\_tbl WHERE emp\_name LIKE‘r%’,’%r’,’%r%’

SELECT \*FROM employee\_tbl WHERE emp\_name LIKE ‘\_\_\_’;

SELECT \*FROM employee\_tbl WHERE emp\_name LIKE ‘\_\_\_%’,’%\_\_\_’;

SELECT emp\_name FROM employee\_tbl obj1 WHERE EXISTS

(SELECT \* FROM employee\_tabl2 obj2 WHERE OBJ2.emp\_id=03 AND obj1.id=obj2.id)

SELECT \* FROM employee\_tbl WHERE salary >ALL

(SELECT \* FROM employee\_tbl WHERE salary< 15000) //compare to that <15000 condit

SELECT \* FROM employee\_tbl WHERE salary >ANY

(SELECT \* FROM employee\_tbl WHERE salary< 15000) //compare to lowest value

Negative Operator- NOT BETWEEN, NOT IN, NOT LIKE, IS NOT NULL, NOT EXISTS

Arithmatic Operator- +, -, \*, /

SELECT salary+20000 FROM employeee\_tbl WHERE emp\_name=’rajesh’;

SELECT salary-200 FROM employeee\_tbl WHERE emp\_name=’rajesh’;

SELECT salary\*2 FROM employeee\_tbl WHERE emp\_name=’rajesh’;

SELECT salary/2 FROM employeee\_tbl WHERE emp\_name=’rajesh’;

|  |
| --- |
| SELECT COUNT (\*) AS newcolumn\_name,  SUM (salary) AS sum\_name,  MAX (salary) as maxcolumn\_name;  MIN (salary) as mincolumn\_name;  AVG (salary) as avgcolumn\_name; |

**Aggregate Function:**

SUM- return total of values in column

COUNT- count rows donot contain null

MAX-return max value from group of rows

MIN-return min value from group of rows

AVG- return average value

Group by:

SELECT <column\_name> FROM <table\_name> GROUP BY <value> //ascending sorting

SELECT salary FROM employee\_tbl GROUP BY salary,emp\_name; //not display identical

Order by:

SELECT <column\_name> FROM <table\_name> ORDER BY <value> //ascending sorting

SELECT salary FROM employee\_tbl ORDER BY salary,emp\_name; //not display identical

Having:

HAVING <value> <condition>;

Ex. SELECT SUM(salary) AS sum\_name FROM emp\_tbl GROUP BY salary

HAVING salary>15000 ORDER BY sum\_name //orderby do same as groupby

**Join**:

1.cross join : without condition

2.Inner join (or) Equi join

3.Non equi join – compare each row in first table with all row in 2nd table

4.Outer join – a)Left outer join b)right outer join

5.Self join

6.Union,Union All

7.Intersect, Expect

**Cross Join:**without condition

SELECT e.ename s.sname FROM employee\_tbl AS e CROSS JOIN employe\_tbl2 AS s

//4 rows in E and 3 rows in S so output is 28 rows

**Inner Join:** join with condition

SELECT E. emp\_id,E.emp\_name,S.emp\_id,S.emp\_name

FROM employee\_tbl E INNER JOIN employee\_tbl2 S ON E.emp\_id=S.emp\_id //AS no need

//output is printing same equal name in both S and E

**Non Equi Inner Join**

SELECT E. emp\_id,E.emp\_name,S.emp\_id,S.emp\_name

FROM employee\_tbl E INNER JOIN employee\_tbl2 S ON E.emp\_id<>S.emp\_id

//ouput is 27 rows not display that one equal cond row

**Left Outer Join:**



SELECT E.emp\_id, E.emp\_name, S.dname

FROM employee\_tbl E LEFT OUTER JOIN employee\_tbl2 S

ON E.emp\_id=S.emp\_id

//match left table with right and display null if right table not have left table values

**Right Outer Join:**



just opposite to left outer join

**Self Join:**

SELECT E.ename, E1.ename as mname //mname for not to confuse while output

FROM employee\_tbl E, employee\_tbl E2 //same table diff alieas name

WHERE E.eid=E1.mid

**Union and Union All:**

SELECT emp\_name FROM employee\_tbl UNION SELECT emp\_id FROM employee\_tbl2

/display name and id shuffled

SELECT emp\_name FROM employee\_tbl UNION SELECT emp\_name FROM employee\_tbl2

**Intersect:**

SELECT e\_name FROM employe\_tbl INTERSECT SELECT m\_name FROM employe\_tbl2

//display only values which are same

**Except:**

SELECT e\_name FROM employe\_tbl EXCEPT SELECT m\_name FROM employe\_tbl2

//display only values in first table

**SUB Query:** ANY, ALL, IF, IF NULL, NULL IF

first inner query will execute and then outer will execute

SELECT E.ename E.city E.salary FROM emp\_tbl E

WHERE E.salary >(SELECT salary FROM emp\_tbl2 WHERE saalry<15000)

INSERT INTO emp\_tbl SELECT E.eid, E.ename, E.salary

WHERE E.salary >(SELECT salary FROM emp\_tbl2 WHERE salary <15000 )

UPDATE emp\_tbl SET salary =salary+5000

WHERE eid IN (SELECT eid FROM emp\_tbl2 WHERE city=’chennai’)

DELETE FROM emp\_tbl

WHERE eid=(SELECT eid FROM emp\_tbl2 WHERE ename=’rajesh’)

**Embedded Sub Query:**

SELECT E.eid ,E.ename, E.salary FROM emp\_tbl E

WHERE E.eid IN (SELECT ED.eid FROM emp\_tbl2 ED

WHERE eid=(SELECT EP.eid FROM emp\_tbl3 EP

WHERE salary=15000 ))

**Foriegn Key:**

create table stu\_table

(r\_emp\_id int(2),

FOREIGN KEY (r\_emp\_id) REFERENCES emp\_table (emp\_id),

stu\_id int(2) AUTO INCREMENT,

stu\_name (2) VARCHAR(30) NULL);

**Character Function:**  
Concatenation: SELECT ‘muthu’+’rajesh’

Replace: SELECT city REPLACE(city, ‘mum’,’bom’) FROM emp\_tbl; //bombai mumbai

Uppercase: SELECT city UPPER(cityname) FROM emp\_tbl; //mumbai MUMBAI

Lowercase SELECT LOWER(cityname) FROM emp\_tbl; //MUMBAI mumbai

Substring: SELECT SUBSTRING(ename,1,3) FROM emp\_tbl; //rajesh raj

Length: SELECT ename, LEN(ename) FROM emp\_tbl; //rajesh 5

Absloute: SELECT ABS(emp\_id) as absolutecolumn\_name FROM employee\_tbl;

Ceiling: SELECT CEILING(emp\_id) as absolutecolumn\_name FROM employee\_tbl;

Floor: SELECT FLOOR(emp\_id) as absolutecolumn\_name FROM employee\_tbl;

Squareroot: SELECT SQRT(emp\_id) as absolutecolumn\_name FROM employee\_tbl;

Power: SELECT POWER(emp\_id,2) as absolutecolumn\_name FROM employee\_tbl;

Date Functions:

SELECT GETDATE();

SELECT DATEADD(DAY,1,DOJ) FROM emp\_tbl; date incremented by value 1

SELECT DATEADD(MONTH,2,DOJ) FROM emp\_tbl; month incremented by value 2

SELECT DATEADD(YEAR,3,DOJ) FROM emp\_tbl; year incremented by value 3

SELECT DATENAME (MONTH,DOJ) FROM emp\_tbl; display months in names

SELECT DATEPART(MONTH,1,DOJ) FROM emp\_tbl; display months in numbers

Data conversion:

SELECT emp\_id= (str)emp\_id FROM emp\_tbl; //strng typecasting

Cursor:

used to indicate which row, the operation is now. it should open before fetch

DECLARE cursor\_name CURSOR FOR SELECT \* FROM emp\_tbl;

OPEN cursor\_name

FETCH cursor\_name

CLOSE cursor\_name

DEALLOCATE cursro\_name

Store Procedure:

CREATE PROCEDURE procedures\_name AS SELECT ename,salary FROM emp\_tbl

EXEC procedure\_name //to execute

DROP PROCEDURE procedure\_name //to close

Trigger:

|  |  |
| --- | --- |
| CREATE TRIGGER trigger\_name  ON database\_name  FOR CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE  AS  print ‘DDL operation is not allowed, trigger is enable’  ROLLBACK | CREATE TRIGGER trigger\_name  ON database\_name  FOR insert, delete,update  AS  print ‘DDL operation is not allowed, trigger is enable’  ROLLBACK |
| DISABLE TRIGGER trigger\_name ON database\_name | DISABLE TRIGGER trigger\_name ON database\_name |
| DROP TRIGGER trigger\_name ON database\_name | DROP TRIGGER trigger\_name ON database\_name |

**Some Fucking extra theories:**

**Aggregate function:** we should use this function before groupby

AVG

COUNT

SUM

MIN

MAX

**Keys:**

1.super key - set attributes identify row

2.candidate key - minimal set of identity rwo

3.Primary key - an attribute identify row

4.Foreign key - column of one table points of onetable points

5.Unique - can have null

6.Secoundary key - might not be unique but we can strill use it identitfier

**Normalisation reduces dublication:**

1.First Normal form

2.Second Normal form

3.Third Normal form

4.Boyce-codd normal function

5.Fourth Normal form

6.Fifth Normal form

DataModels

1. Record Based a)Network model

b)Hierarical model

c)Relational model

2)Object based a)ER model

b)Object oriented model

c)sematic model

d)functional model

ER Model:

|  |  |
| --- | --- |
| Entity - |  |
| Relationship- |  |
| Attributes |  |
| Multivalued attributes |  |
| Derived aatribute |  |
| Composite attribute address |  |

**Cardinality:**

OneToOne

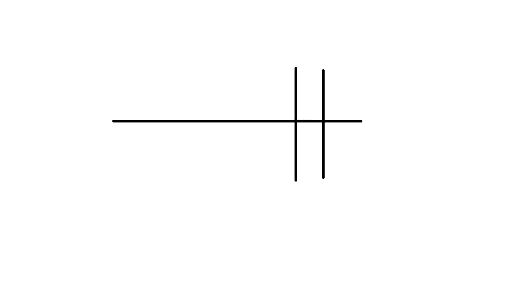
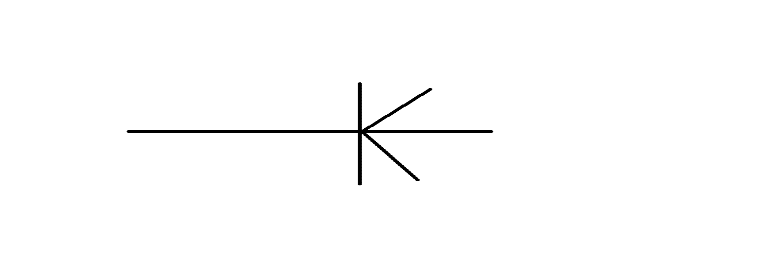
OneToMany

ManyToOne

ManyToMany

**Crow Foot Notation:**

|  |  |
| --- | --- |
| One |  |
| OneOrMore |  |
| Zero or one or more |  |
| Zero or one |  |
| many |  |



Many House

Rajesh

Rajesh

House

1 many