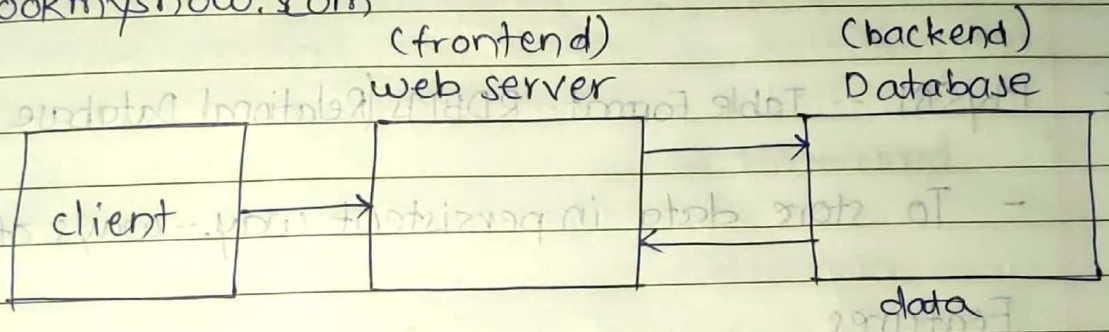
# Database Technologies

Day 1 29-6-21

flipkart.com
railway reservation
bookmyshow.com




* Types of Databases

- SQL Data (Tabular, Structured)
  - Oracle --- Licensed --- Java is managed by Orcle
  - mysql --- freeware
  - postgreSQL --- freeware
  - SQL server --- Microsoft --- Licensed --- .NET Technology
- NoSQL data (Unstructured, JSON) (javascript object notation
  - MongoDB
  - Cassendra
  - Hbase
- GraphDB
  - Neo4j
- Memory Database (HQL, Memdb)
  - Research type project
  - memory database --- data is less --- faster --- volatile.

Server
- Those companies are going to store data, programs, backups.
To store these data multiple servers are required.

* MySQL -- Table Format -- RDBMS (Relational Database management system)
- To store data in persistent way. -- ways of storing files

Features
- retrieval of data is very easy.
- Sharing data becomes very easy.
- redundancy of data decreases (repeatation).
- Transaction control --- ACID... every SQL supports ACID property.
- transfer of funds
source a/c money gets deducated but does not get deposited in destination a/c.

* ACID Property
Atomicity -- every transaction gets exeecutated as a single unit.
Consistency -- after every transaction data should be in correct state.
Isolation - any user when login to database should read same data.
Durability - longer period of time consistency will be maintained by application.

- Secure - financial application, voting, census -- RDBMS

* Difference between SQL and NoSQL.

| SQL | NoSQL |
|---|---|
| 1) In all secure applications ...financial app | 1) Less security... social media |
| 2) structured | 2) Unstructured |
| 3) Table | 3) Collection |
| 4) Transaction control | 4) Less transaction control. |

SQL Database ... MySQL

Tabular ... Rules

| Cust id | Cname | Addr | email | A/C no | Balance | Type | relmgr |
|---|---|---|---|---|---|---|---|
| 1 | kishori | Aundh | ab@gmail. | 1 | 12345 | Saving | AA |
| 1 | kishori | Aundh | a.b@ | 2 | 123451 | current | AA |
| 1 | kishori | Aundh | a.b@ | 3 | 12345 | demat | AA |
| 2 | Rajan | Baner | r.b@ | 4 | 111111 | saving | BB |

- every row should be uniquely identifiable.

1) primary key ... (account no.) :- minimal subset of attributes which identifies the row uniquely.

2) Composite key :- Many times your primary key is formed by more than one column, then it called as Composite key.

3) Candidate key (A/C no. ,passport no., adhar no.)
- Minimal subset of attributes that identifies the row uniquely & which may become primary key.

4) Foreign key :- Before we enter data in column, if we refer primary key of same table or different table it is called as foreign key. (To maintain correctness of data)

5) Super key :-
Any combination that identifies the row uniquely.
a/c no, a/c no + cus id, a/c no + cname, a/c no + cname + cus id + pass.

| Student id | Subject id | Marks |
|---|---|---|
| 1 | 100 | 98 |
| 1 | 200 | 98 |
| 2 | 100 | 95 |
| 2 | 200 | 99 |

subject id + student id -- primary key -- composite key

Account Table

| Cust id | A/c no | Balance | Type | rel mgr |
|---|---|---|---|---|
| 1 | 1 | 12345 | saving | AA |
| 1 | 2 | 123451 | current | AA |
| 2 | 3 | 12345 | demat | AA |
| | 4 | 111111 | saving | BB |

cust id - foreign key

Customer

| Cus id | Cname | Addre | email | Adhar no. | Passport no. |
|---|---|---|---|---|---|
| 1 | Kishori | Aundh | a.b@ | 2222222 | 1234555 |
| 2 | Rajan | Baner | r.b@ | | |
| 3 | Revati | Aundh | zdxn | 2333 | 111111 |

6) surrogate key :- if you want to identify the row uniquely with database, then we need to add some external or virtual column in table which id not part of data is called surrogate key.

7) Unique key :- It does not allow duplicate values in the column, but it allow to store multiple null values.

SQL (Structured query language)

PLSQL (Procedural language - structured query language)

- loops, if statement, variable declaration is not there

4GL (4 generation language)

- loops, if statement, variable declaration -- PLSQL

employee

| Emp id | Ename | Sal | mgr no | dept no. | Pass port | Adhar |
|---|---|---|---|---|---|---|
| 1 | Rajan | 11111 | 3 | 10 | | |
| 2 | Revati | 22222 | 3 | 20 | | |
| 3 | Anil | 44444 | | 10 | | |
| 4 | Rajan | | | 40 | | |

Primary - emp id Candidate - emp id, adhar, passport

unique - adhar, passport foreign - ~~dept no~~ mgr no -- reference emp no, dept no -- reference dept id of dept

Department

| Dept id | Dname | Location |
|---|---|---|
| 10 | HR | Mumbai |
| 20 | Accounts | Pune |

primary - Dept id. candidate - dept id, dname
super - dept id + dname, dept id, dname, dept id + dname

Types of statement

| | |
|---|---|
| DQL | Data query language |
| DDL | Data definition language |
| DML | Data Manipulation language |
| DCL | Data Control language |
| TCL | Transaction control language |

| | |
|---|---|
| DQL | select |
| DDL | create, table, Alter, table, truncate, drop |
| DML | Insert, update, delete |
| DCL | Grant, revoke |
| TCL | commit, Rollback, savepoint. |

step 1:-
open mysql client window
step 2:-
To create database

```
create database iacsdedacmay21
```

step 3:-
To switch to the database

```
use iacsdedacmay21
```

step 4:-
download demobldmysql.sql
and store it in C:\mydata

mysql> source c:\mydata\demobldmysql.sql.

show tables;
It will show in tabular format.
-To see all fields and all rows within the tables.
select * from emp;

select * from dept;
- To see the columns within tables.
desc emp;

desc dept;

desc salgrade;

Day 2 30-6-21

- To see particular ~~columns~~ rows i.e. empno, ename.

```
select empno, ename
from emp;
```

- To display rows in which sal > 2000.

```
select empno, ename, sal, deptno
from emp
where sal > 2000;
```

-To display all emp with sal >2000 and ename = 'BLAKE'

```
select empno, ename, sal, deptno
  from emp
where sal >2000 and ename = 'BLAKE';
```

sal >2000 will get checked 14 time it is true for 14
ename = 'BLAKE' will get checked 6 time.
total conditions 14+6 = 20

```
where ename = 'BLAKE' and sal >2000;
```

ename = 'BLAKE' will get checked 14 times.
sal >2000 will get checked 1 time
total condition 14+1 = 15
hence this is more time efficient.

-To display emp sal >2000 or ename = 'BLAKE'

```
select
from empno, ename, sal, deptno.
where ~~BLAKE~~ename = 'BLAKE' or sal >2000.
```

ename = 'BLAKE'
sal >2000

where sal >2000 or ename = 'BLAKE'
sal >2000 get checked 14 times ... 8 times false
ename = 'BLAKE' ... 8 times
total condition 14+8 = 22

Rules to improve efficiency of the query

- in 'and' operator the condition which is false more times should be first condition.

- in 'or' operator the condition which is true should be first condition.

```
select *
from product
where price > 2000 and pname = 'bag'
```

```
select *
from product
where pname = 'bag' and price > 2000.
```

- In this the 1st query is less efficient than 2nd.

- Find all emp who joined on particular date i.e. 28 sept 1981.

```
select *
from emp
where hiredate = '1981-09-28';
```

- To find all emp who do not earn commission.

```
select *
from emp
where comm is null or comm=0;
```

- To find all emp who earn commission

```
select *
from emp
where comm is not null or comm!=0;
```

- To find all emp with sal >1500 and sal<3000

```
select *
from emp
where sal >=1500 and sal <=3000;
```

To check range of data

```
select *
from emp
where sal between 1500 and 3000;
```
(greater than equal to)

```
where sal between 1501 and 2999;
```
(to check in between)

- To find all emp with sal not >=1500 and sal not <=3000

```
select *
from emp
where sal not between 1500 and 3000;
```

- To find all emp with sal either 1600 or 2000 or

```
select *
from emp
where sal =1250 or sal=1600 or sal=2000;
```

use in operator

```
select * from emp
where sal in (1250, 1600, 2000);
```

- To display names of all emp who work in dept no 10, 20

```
select * from emp
where deptno in (10, 20);
```

where deptno not in(10, 20); ... do not work in 10 & 20.

- To find all emps with name starts with A.
  To check pattern use like operator.
  % -- matches with 0 or more char.
  _ - matches with 1 char

(under score)

```
select * from emp
where ename like 'A%';
```

- To find all emps with name ends with N.

```
select * from emp
where ename like '%N';
```

- To find all emps with name has A at 2nd part ends with N.

```
select * from emp
where ename like '_A%N';
```

- To find all emps with name A at 2nd pos.

```
select * from emp
where ename like '_A%';
```

- To find all emps with L at 3rd pos or starts with M or I at second last pos.

```
select * from emp
where ename like '__L' or ename like 'M%'
or ename like '%I_';
```

- A at 2nd pos or does not starts with M or not having I at second last position.

```
select * from emp
where ename like '_A%' or ename not like '%I_'
or ename not like 'M%';
```

- Starts with M or A or B.

```
select * from emp
where ename like 'A%' or ename like 'B%' or
ename like 'M%';
```

REGEXP

^ ... matches pattern at beginning
$ ... matches pattern at the end
* ... matches with 0 or more char.
+ ... matches with 1 or more char.

^ - begin with
[^] - In bracket other than.

_ - matches with only 1 char.
? - matches with 0 or 1 char.
[a-z A-Z 0-9] - matches with any one char or num.
[aeiou] matches with any vowel.
[^aeiou] Anything other than vowel.
[^0-9] Except num every char.
{m} - exactly m occurance
{m,n} - minimum m & maximum n occurance
(a|b|c) - multiple patterns

- starts with M, A or B.

```
select * from emp
where ename REGEXP '^[MAB]';
```

- A at 2nd pos or does not starts with M or not having I at second last pos.

```
select * from emp
where ename REGEXP '(^.A|^[^M]|[^I].$)';
```

- Starts with A & ends with N & 3rd pos is L.

```
select * from emp
where ename like 'A_L%N';          (1)
where ename REGEXP '^A.L.*N';      (2)
```
o/p of (1) and (2) is same.

- LL anywhere in the name.

```
select * from emp
where ename like '%LL%';           (1)
where ename REGEXP 'L{2}';         (2)
where ename REGEXP 'LL';           (3)
```

- starts with any char betn range A to H.

```
select * from emp
where ename REGEXP '^[A-Ha-h]';
```

- ifnull

```
select empno,ename,sal,comm sal+ifnull(comm,0)
from emp;
```

- To calculate net sal = sal+comm & display col names

```
select empno,ename,sal,comm, ifnull(comm,0),
sal+ifnull(comm,0) "net sal"
from emp;
```

- Display empno, ename, sal & increased sal by 10%.

sal + 0.10 * sal

```
select empno, ename, sal, sal*1.10 "newsal";
from emp;
```

- Display all emps arranged on name.

```
select empno, ename, sal, sal*1.10 "new sal"
from emp
order by ename; (in ascending order)
order by ename desc; (in desending order)
```

- Display all emps arranged by ~~name~~ job if job is same then arrange on ename in descending order

```
select empno, ename, sal, sal*1.10 "new sal"
from emp
order by job, ename desc;
```

(in order by clause we can use max 255

- Built in functions are available.

1. Aggregate function - Functions those work on group of rows.

2. Single row function- Function which works on each row is called as single row function.
   a. number- abs, round, truncate, ceil, floor, sqrt.
   b. character
   c. dates

select empno, ename, sal, comm, ~~sal~~ abs(ifnull(comm,0)-sal) from emp;

select round (1234.6722, 2)
1234.67
select round (1234.6750, 2) 1234.68 (It will round off)
select truncate (1234.6722, 2)
1234.67 (It will consider only start 2 digits).

select ceil (12.56)
In ceil it will always remove digits after decimal and always show next number.

select floor (12.56)
In floor it will always remove digits after decimal and always show that no. only.

select sqrt (4)
2

```
select upper ('asdf')
       ASDF

select lower ('ASDF')
       asdf

select empno, upper(ename), sal
  from emp
where upper(ename) = 'ALLEN';

select empno, ename, substr (ename, 1, 3), sal
  from emp;

select substr ('welcome', 3, 3);

select empno, name, left (ename, 3), sal
 from emp;

select right ('welcome', 3);

select concat ("hello", "everyone");

select upper ('hello'), lower ('HELLO'),
concat(left (upper ("hello"), 1), right (lower ("Hello"), 4))
```

-Display ename & email of every emp.

email .. ename followed by @mycompany.com

```
select empno, ename, concat(ename, "@mycompany.com")
  from emp
```

-Display name, job and email.
email ename followed by . followed by 2,3 4 char of job followed by @mycompany.com

```
select empno, ename, job, concat (ename, '.', substr (job, 2,3), '@mycompany.com') "email"
 from emp;
```

- Find how many char are there in ename.

```
select ename, length (ename)
 from emp;
```

```
select ename, lpad (ename, 10, '*') (Add * in left side)
 from emp;
```

```
select ename, rpad (ename, 10, '*') (Add * in right side)
 from emp;
```

```
select rpad (1234, 7, 'X');
        1234XXX

select concat (lpad (1234, 10, 'X'), 'XXX');
        XXXXXX1234XXX

select ('    hello    ');

select ltrim ('  hello  '); (It will remove spaces from left side)
select rtrim ('  hello  '); ( from right side)
select trim ('  hello  '); ( from both side)
```

select reverse ('welcome');
emoclew

select ~~ascii~~ ascii ('A')
65

select empno, ename, ~~format (sal,~~ concat ('$',
format (sal, 1))
from emp;

select format (1234556, 2);
1,234,556.00

* Number and char related functions.

| Function | Use | Example | |
|---|---|---|---|
| abs() | To convert -ve to +ve | select abs(20-30) | 10 |
| round() | It will round the no. to specified no. of digits after decimal point | select round (20.5634, 2)<br>select round (20.5650, 2) | 20.51<br>20.57 |
| truncate() | It will truncate no. to specified no. of digits after decimal point | select truncate (20.5634, 2)<br>select truncate (20.5650, 2) | 20.56<br>20.51 |
| ceil() | It will remove all digits after decimal point & always show next no. | ceil (20.57)<br>ceil (20.13) | 21<br>21 |

| | | | |
|---|---|---|---|
| floor() | It will remove all digits after decimal point & always show same no. | floor (20.57) | 20 |
| | | floor (20.13) | 20 |
| sqrt() | It will display sqrt of the no. | sqrt (4) | 2 |
| ASCII('A') | Print ascii value of given char | ASCII ('A') | 65 |
| Upper() | convert given string to uppercase | Upper ("hEllo") | HELLO |
| lower() | convert given string to lowercase | lower ("hEllo") | hello |
| concat (exp1, exp2, ...) | "Concatenate the string | concat ('a', 'xx', 'c') | axxc |
| left(exp,n) | This will retrieve n char from leftmost side of string | left ("Hello", 3) | Hel |
| right(exp,n) | This will retrieve n char from rightmost side of string | right ("Hello", 3) | llo |
| substr(str, start, n) | This will retrieve n char from start pos. from given string | substr("welcome, 3, 4) | lcom |

reverse(str) Display string in reverse reverse("Hello") olleH

trim(str) Remove extra spaces from both sides. trim(' Hello ') Hello

ltrim(str) Remove extra spaces from left side. ltrim(' Hello ') Hello (without spaces from left)

rtrim(str) Remove extra spaces from right side. rtrim(' Hello ') Hello (without spaces from right)

length(str) Display no. of char. length('welcome') 7

lpad(str, length, char) It will add char on left side to make length no of chars. lpad("Hello", 10, "-") -----Hello

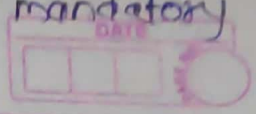
rpad(str, length, char) It will add char on right side to make length no of chars. rpad("Hello", 10, "-") Hello-----

## Day 3 1-7-21

- To display a code which is combination of first char of name & last 3 char of job whose sal is either 1250 or 2000 or 1600.

```
select ename, job, concat (left(ename,3), right(job,3))
from emp
where sal in (1250, 2000, 1600);
```

If alias name contains space then it is mandatory to write in double quotes (" ")



- Find length of job, & display name in reverse

```
select ename, job, length(job), reverse(ename)
from emp;
```

- Find all name which has A at 2nd pos. or 3rd pos or E at the end.

```
select empno, ename, job from emp
where ename like '__A%' or ename like '_A%' or
ename like '%E';
```

- To display all emp who do not earn commision & has A at 2nd or 3rd pos. & ends with E.

```
select ename, job from emp
where comm is null or comm=0 and ename like '_A%E' or
ename like '__A%E';
```

- Select all emp who works in dept no 10 or 20 and sal >= 1200 or sal <= 3000

```
select ename, empno, sal, dept no from emp
where deptno in (10,20) and sal between 1200 and 3000
```

* Date related function

```
select now();        (Today's date and time)
2021-07-01 08:53:53

select curdate();    (To show only date)
2021-07-01
```

select date_format(curdate(), '%m/%d/%Y')
07/01/2021
%M - To see in char %D - In char %Y - To see in 4 digits(year)
%m - To see in no. %d - In no. %y = To see year in 2 digits

select date_format(curdate(), '%M %D %Y')
July 1st 2021

- To calculate difference between two dates.

select datediff('2020-01-01', '2021-01-01');

select empno, ename, (datediff(curdate(), hiredate)/365)
from emp;

select empno, ename, (datediff(curdate(), hiredate)/365) experience
from emp
where floor(datediff(curdate(), hiredate)/365) > 38;

- To see date after 1 week.

select curdate, date_add(curdate(), interval 1 week);

- To see date after 1 month

select curdate, date_add(curdate(), interval 1 month);

- Find date after 2 years 4 months 2 days

select date_add(date_add(date_add(curdate(), interval 2 year), interval 4 month), interval 2 day);

- Find date before.

```
select date_sub(curdate(), interval 2 month);
```

```
select year(curdate());          2021
select month(curdate());         7
select day(curdate());           1
select quarter(curdate());       3
select weekday(curdate());       3
select weekofyear(curdate());    26
```

- To find all emp who joined 1981.

```
select empno, ename, hiredate, year(hiredate)
 from emp
where year(hiredate) = 1981
```

- Find all emp who joined on 1st august

```
// select empno, ename, hiredate
  select * from emp
where month(hiredate)=08 and day(hiredate)=01
```

- Find all emp who joined in either aug or sept.

(1)
```
select * from emp
where month(hiredate)=08 or month(hiredate)=09
```

(2)
```
select * from emp
where month(hiredate) in (8, 9);
```

- Find all emps who joined before 1 aug 1982.

```
select * from emp
where hiredate < '1982-08-01';
```

- To find the week

```
select weekday('2000-12-31'), week('2000-12-31'),
weekofyear('2000-12-31');
```

* To insert data in table

~~insert~~

```
select * from emp;
insert into emp values(123, 'Rajan', 'Designer', 7902,
'2020-12-10', 34567, 3456, 10);

insert into emp values(124, 'Revati', 'Designer', 7902,
'2019-12-10', 34567, 3456, 20);
```

To see emp joined 2 years before.

```
select empno, ename, hiredate, year(hiredate), year(curdate)
from emp
where year(hiredate) = year(curdate()) - 2;
```

- In mySQL autocommit is by default on

- To off auto commit

```
set autocommit = 0;
```

If autocommit is ON then rollback will not work.

- To rollback the entry

```
rollback;
```

- To store the entry permanent in the disk.

```
commit;
```

- To set autocommit ON.

```
set autocommit=1;
```

- Delete record

```
delete from emp
where empno=125;
```

Day 4 2-7-21

```
select sum(sal), max(sal), min(sal), avg(sal), count(*), count(comm)
from emp;
```

| sum(sal) | max(sal) | min(sal) | avg(sal) | count(*) | count(comm) |
|---|---|---|---|---|---|
| 136182 | 34567 | 800 | 7565.66 | 18 | 8 |

(Count all values) (It will ignore null value)

- Group by

- To get sum of sal from group of dept no.

```
select deptno, sum(sal) from emp
group by deptno;
order by deptno;
```

- To get min sal of every deptno.

```
select deptno, sum(sal), count(*), min(sal)
from emp
group by deptno
order by deptno;
```

- Group by job

```
select job, sum(sal), avg(sal), count(*)
from emp
group by job;
```

- Group by deptno & job and get sum of sal.

```
select deptno, job, sum(job), count(*), min(sal)
from emp
group by deptno, job;
order by deptno, job;
```

- Find sum of sal for all emp in dept no 10

```
select deptno, job, sum(sal) from emp
where deptno = 10;
```

- Find sum of sal & avg for each dept for all emp who are managers.

```
select deptno, job, sum(sal), avg(sal)
from emp
where job = 'manager'
group by dept no
order by dept no;
```

- Find min sal for each dept & arrange data based on min sal.

```
select deptno, count(*), min(sal) from emp
group by deptno
order by min(sal);
```

- Having

```
select deptno, sum(sal), sum(sal+ifnull(comm,0), min(comm)
from emp
group by deptno
having sum(sal+ifnull(comm,0)) > 2000;
```

- Display sum of sal, min, avg for each dept, if dept has more than 5 emps.

```
select deptno, sum(sal), min(sal), avg(sal), count(*)
from emp
group by deptno
having count(*) > 5
```

- Display min sal, avg of sal + 10% of sal, avg comm for all dept which has 2 managers.

```
select min(sal), sum(avg(sal)+0.1 * sal), avg(ifnull(comm,0))
from emp
group by deptno
where job='manager'
group by deptno
having count(*) >2
```

* Rules to remember for group by clause

1. if condition is based on aggregate function then put condition in having clause.
2. if condition is based on existing column or any derived column then add it in where clause.
3. in select statement you can use only columns other than aggregate function, which are used in group by clause.

- Display sum, min, max, avg of sal & count of emp if there sal is >=1250 and <=3000 and if their manager is either 7698 or 7566.

```
select sum(sal), min(sal), max(sal), avg(sal), count(*)
from emp
where sal between 1250 and 3000 and mgr in(7698,7566)
```

- Display sum, avg, min, max, count for all emp who are working under same manager.

```
select mgr, sum(sal), min(sal), max(sal), avg(sal), count(*)
from emp
group by mgr;
```

- To see different dept (To see the repeated no only once)

```
select distinct deptno from emp; (To see diff. departments)
select distinct job from job emp;
```

distinct keyword - To show unique values.

* DML (Data manipulation language)

insert, delete, update

```
insert into dept values (50, 'HR', 'chennai');

insert into dept (dname, deptno) values ('HR', 50);
```

- To insert multiple rows in one ~~column~~ statement

```
insert into dept
values (100, 'xxx', 'chennai'), (200, 'yyy', 'mumbai'),
  (300, 'zzz', 'pune');
```

- Insert all columns but not following column sequence of desc statement.

```
insert into dept (dname, deptno, dloc)
    values ('xxx', 300, 'chennai');
```

- Modify the data in existing row.

```
update dept
 set loc = 'Pune'
where deptno = 60

update dept
set loc = 'Pune', dname = 'Insurance'
where deptno = 50 and loc is null;
```

- To add sal 100 for every emp.

```
update emp
set sal = sal + 100;
```

- To delete data from existing table

```
delete from dept
where loc = 'chennai';
```

Q. Which of the following will delete all rows of table.

a) delete * from emp;
b) delete from emp;
c) delete from emp where deptno is null.
d) Both A and B

Answer :- B

rollback, commit, save point these are called as TCL (Transaction control language)

if I have dept table with 10 rows & autocommit is off

```
insert 100
insert 200
insert 300
commit                    13

rollback
select * from dept        13
```

(Because there is no entry bet$^n$ rollback & commit.

if I have student table with 13 rows in it & autocommit is off.

```
insert 100
insert 200
insert 300
commit;                   16
update 100
delete 200
update 300
select * from student;    15

100 rows

10 rows
savepoint A
7 rows
savepoint B
```

8 rows

savepoint c

s1
s2
s3 } If we made mistake in these

rollback to c (It will undo upto savepoint c)

- How to use savepoint

```
set autocommit=0;

select * from dept;
commit;
insert into dept values(301,'ccc','Pune');
insert into dept values(302,'ccc','Pune');
savepoint A;
insert into dept values(303,'cccl','Pune');
insert into dept values(304,'cccl','Pune');
savepoint B;
insert into dept values(305,'cccl','pune')
insert into dept values(306,'cccl','Pune')
rollback to B; (It will rollback upto only B)
commit;
select * from dept;
```

- To Take backup of data

d:\data\mydata.sql

mysql> mysqldump -u root -p root123 > mydata.sql

To upload data in sql

C:\mydata\data>mysql -u root -p root123 < d:\data\mydata.sql

or

mysql> source d:\data\mydata.sql

- To delete all table:
  drop table emp
  drop table dept
  drop table salgrade

(To remove the multiple entries if mistakely take sql file twice or thrice)

Day 5 3-7-21

- To find sum of sal for each department for all emp with bonus = sal * 0.15 and if I want to display all rows with bonus > 700.

```
select deptno, sum(sal) from emp
where sal * 0.15 > 700
group by deptno
order by deptno;
```

- To display min(sal+comm) for each job if count of the job > 3.

```
select job, min(sal+(comm,0)), count(*)
from emp
having count(*) > 3
order by job
```

- Display thousand seperator & $ symbol for comm, if it is null then display 0 for all emp whose name starts with A & ends with N.

```
select ename, sal, concat('$', format(ifnull(comm,0))
  from emp
  where ename like 'A%N'
```

select ename, sal, format (sal, 2, en-US)

13,234.00 (To see in US format)

select ename, sal, format (sal, 2, en-UK)
(To see in UK format)

case statement

* DDL (Data Definition language)

if comm is null then need improvement
comm < 500 ok
comm >= 500 and < 1000 good
comm >= 100 excellent

select empno, ename, sal, comm, case when comm is null then 'need improvement'
when comm < 500 then 'ok'
// when comm ~~<1000~~ between 500 and 999 "Good"
when comm < 1000 then 'good'
else 'excellent' end "Remark"
from emp;
~~where~~

- Display prodid, pname, price,
price < 50 less expensive
>= 50 < 100 moderate
>= 100 expensive
if price is null wrong price

```
select prodid, pname, price,
case when price is null or price=0 then 'wrong price'
 when price<50 then 'less expensive'
when price>=50 and price<100 then 'moderate'
else 'expensive' end "Remark"
  from product;
```

- Display pid, pname, price, cid
if cid=1 then chips
cid=2 biscuits cid=3 snacks otherwise others.

```
select pid, pname, price, case when cid=1 then 'chips'
when cid=2 then 'biscuits'
when cid=3 then 'snacks'
else 'others' end "category"
from product;
```

OR

```
select pid, pname, price, case cid when 1 then 'chips'
 when 2 then 'biscuits'
when 3 then 'snacks'
 else 'others' end 'category'
 from product;
```

- Grade level of emp according to job.

```
select empno, ename, job, case job when 'CLERK' then
 'Grade 1'
when 'SALESMAN' then 'Grade 2'
when 'MANAGER' then 'Grade 3'
when 'ANALYST' then 'Grade 4'
when 'Designer' then 'Grade 3'
 else 'Not Reg Emp' end "Grade Level"
   from emp;
```

- DDL statements

| | |
|---|---|
| create table | (new table) |
| alter table | (modify existing table) |
| drop table | (delete table) |
| truncate | (delete only data) |

All DDL statements are autocommit.

To delete entire table

```
drop table emp;
drop table dept;
```

To delete all records.

```
truncate table emp;
```

* Difference between Truncate and Delete

| Truncate | Delete |
|---|---|
| DDL | DML |
| autocommit (no rollback) | Rollback is possible till we commit |
| Cannot use where cond? | We may use where cond" to remove few rows |
| Nested query cannot be used | Nested query is possible to use |

varchar(m) is used in MySQL
varchar2(m) is used in Oracle.

* Data types in mySQL

1) Numeric
2) Date and Time
3) String type
4) enum

1) Numeric

| | |
|---|---|
| INT | |
| TINYINT | -128 to 127 |
| SMALLINT | -32768 to 32767 |
| BIGINT | |
| MEDIUMINT | |
| FLOAT(M,N) | M to display length, D for decimal places (float upto 24 decimal places) |
| DOUBLE(M,N) | (Double upto 53 decimal). |

2) Date and Time

| | | |
|---|---|---|
| DATE | YYYY-MM-DD | 1000-01-01 to 9999-12-31 |
| DATETIME | YYYY-MM-DD | HH:MM:SS |
| TIMESTAMP | | |
| TIME | HH:MM:SS | |
| YEAR(M) | store year in 2 or 4 digits. | |

3) String Type

Binary, large object

| | | | |
|---|---|---|---|
| CHAR(M) | fixed size string | citycode | 1 to 255 |
| VARCHAR(M) | variable length string | cityname | 1 to 255 |
| BLOB or TEXT | 65535 | | |

TINYBLOB or TINYTEXT 255
MEDIUMBLOB or MEDIUMTEXT 16777215
LONGBLOB or LONGTEXT

ENUM ('A', 'B', 'C')
To store specific values or to store null.

* How to create table.

Constraints
1. Table Level - primary key, unique, check, foreign key. Auto increment is technique which allows to insert unique values.
2. Field Level - not null, default

- Table level constraints can be written immediately after field or at the end in create table query.

- Field level constraints can be written immediately after the field.

```
create table myemployee (empid int primary key,
ename varchar(30) not null, price decimal (9,2)
default 100, qty int check (qty > 10),
passportnum int unique, adharnum int unique
not null);

desc myemployee;

insert into myemployee values (10, 'kishori', 1234, 34,
123454, 11111);
```

insert into myemployee values (11, 'Savita', 2345, 56, 123454, 111111);

insert into myemployee values (12, 'Vivekanand', 23452, 11, 1234521, 111131);

insert into myemployee values (13, 'chetan', 23452, 11, 12345222, 111141);

// insert into myemployee values (14, 'Neha', 23452, 11,

insert into myemployee (empid, ename, qty, adharnum) values (14, 'Neha', 11, 111143);

product (pid, pname, qty, price, cid)
category (catid, cname, desc)

create table category (catid int, cname varchar(20) not null, desc varchar (20), primary key (catid));

create table product (pid int pname varchar(20) not null, qty int check(qty>0) price decimal (11,2) check (price>0), cid int, foreign key fk_cid(cid) references catagory (catid) on delete set null on update cascade)

desc product;

insert into category values (1, 'chips', 'very crispy');

insert into category values (2, 'biscuits', 'very sweet');

insert into product values (13, 'lays', 12, 40.00, 1);

insert into product values (13, 'marie', 12, 20.00, 2);

insert into product values (14, 'glucose', 12, 20.00, 2);

insert into product values (15, 'pringles', 12, 20.00, 1);

select * from product;

update category
set catid = 100
where catid = 1;

This is on update cascade

If we update the data in the parent table i.e. catagory then it will also update the data in the child table i.e. product this is on update cascade.

select * from category;
select * from product; (cid will automatically change).

delete from category
where catid = 2;

If we delete cid = 2 from category table then in the product table the value of cid should be replaced by null, these is on delete set null.

```
insert into category values (3, 'snacks', 'very sweet');
insert into product values (16, 'nachos', 12, 20.00, 100);
insert into product values (17, 'hideandseek, 12, 20.00, 2);

select * from category;

select * from product;

delete from category
 where catid = 2;

update category
set catid = 10
where catid = 100;

select * from product;

select * from category;

  update product
   set cid = 3
  where cid is null;

  select * from product;
```

Day 6 5-7-21

* Constraints in SQL

not null - within a field null values are not allowed.

unique - no duplicate values are allowed.

check - Allows data only if satisfies given cond".

primary key - no duplicates and null keys.

default - if user enters null then given value will be assigned.

foreign key - checks the value exists in other column of same table or different table.

- on delete <action>
on update <action>
3 different actions

1. cascade - changes made in parent table will be reflected in child table.

2. Set null - If parent table primary key values are changed then those values in child table are set to null.

3. no action - If any value is present in child table, then that value is not allowed to be removed or change from parent table.

- Why we use constraints

1. Security
2. To avoid wrong input from user
3. Data integrity

vehicle (vid, vname, price)
customer (custno, cname, address, email)
cust-vehicle (custid, vid, purchase date, discountedprice)

One customer can have many vehicle

Vehicle

| vid | vname | price |
|---|---|---|
| 100 | Activa | 80000 |
| 200 | santro | 700000 |
| 300 | Jupiter | 83000 |

Customer

| cid | cname | Address |
|---|---|---|
| 10 | Revati | ~~Ard~~ Aundh |
| 20 | Rajan | Aundh |
| 30 | Anil | ~~Baner~~ |

Vehicle - Customer

| reg no | custid | vid | Purchase date | Discounted Price |
|---|---|---|---|---|
| MH12 1234 | 10 | 100 | 2010-02-05 | 70000 |
| MH2 1111 | 10 | 200 | 2012-04-27 | 645000 |
| MH2 1235 | 20 | 100 | 2010-02-05 | 70000 |

```
create table vehicle (
vid int primary key,
vname varchar(20)
price decimal (9,2) );
```

```
create table customer(
 cid int primary key,
 cname varchar(20) not null,
 address varchar (20) );

create table vehicle_cust(
 vid int, custid int,
 p_date date, dis_price decimal (9,2)
foreign key fk_vid (vid) references vehicle (vid)
 on delete no action on update cascade,
foreign key fk_cid(cust_id) references customer (cid)
 on delete no action on update cascade,
 primary key (vid, custid) );


insert into customer values (10, 'Revati', 'Aundh');

insert into customer values (20, 'Rajan', 'Aundh');

insert into customer values (30, 'Anil', 'Baner');

insert into vehicle values (100, 'Activa', 80000);

insert into vehicle values (200, 'Santro', 700000);

insert into vehicle values (300, 'Jupiter', 75000);

  select * from customer;

  select * from vehicle;
```

values
insert into vehicle_cust(100, 10, '2010-02-05', 75000);

insert into vehicle_cust values(200, 10, '2010-03-05', 650000)

insert into vehicle_cust values(100, 20, '2010-02-05', 75000);

insert into vehicle_cust values(300, 20, '2020-02-05', 73000);

select * from vehicle_cust;

select * from customer;

select * from vehicle;

delete from vehicle
where vid = 100;
Cannot delete or update a parent row
(foreign key references)

delete from customer where cid = 10;
Same error as above.

delete from customer
where cid = 30;

create table myemp(
empid int primary key, ename varchar(20),
mgr int, job varchar(20),
foreign key fk_mgr(mgr) references myemp(empid)

insert into myemp (empid, ename, job) values
(101, 'anil', 'CEO');

insert into myemp values ('102, 'kishori', 101, 'manager');

insert into myemp values (104, 'kishori', null, 'manager

* Create table using auto-increment.

```
create table mytable(
  id int primary key auto_increment,
  name varchar(20) not null,
  email varchar(20));
insert into mytable values(1, 'xxx', 'xxx@sdh);
insert into mytable values(10, 'xxx', 'xxx@sdh');

insert into mytable values(default, 'xxx', 'xxx@sdh');

insert into mytable values(default, 'xxx', xxx@sdh);
```

- Change the starting value for auto-increment.

```
alter table mytable
auto_increment = 1001;

insert into mytable values( default, 'zzz, 'xxx@sdh');

select * from mytable;
```

- To make changes in existing table structure.

alter table ;-
- add constraints, drop constraints
- add new column, drop column, modify column
- rename

```
select * from product;
```

- Add column at the end.

```
alter table product
add description varchar(20);

desc product;

select * from category;

insert into category values (12, 'furniture', 'ksjhxd');
```

- Add column at specific position

```
alter table category
add col1 varchar(20) after cname,
(It will get added after cname)
add col2 varchar (20) first
(It will get added at the first)

desc category;

select * from category;

alter table category
add col3 int default 100;
```

```
desc category;

select * from category;

update category
set col1 = 'xxxx';

select * from category;

alter table category
modify col1 varchar(20) not null;

alter table category                    (If table is
add col1 varchar(20), not null;          empty)
```

1. Add a column with not null constraint if table is empty.
2. But table contains data then add new column, change null values and use modify to add not null constraint.

- To delete the column.

```
alter table mytable category
drop column col2 col3;

desc category;

create table mytable1(
myid int, myname varchar(20));

desc mytable1;
```

- How to add primary key in existing table.

```
alter table mytable
add primary key (myid);

desc mytable1;
```

- How to add foreign key in existing table

```
alter table emp
add constraint fk_deptid
foreign key (deptno) references dept (deptno);
```

- How to delete foreign key in existing table

```
alter table emp
drop constraint fk_deptid;
```

- How to change name of existing table

```
alter table mytable
rename to newmytab;

desc newmytab;
```

Day 7 6-7-21

- Alter table
  1. Add column
  2. Add constraint
  3. Drop column
  4. Drop constraint
  5. Modify column data type or constraint
  6. Rename table
  7. Rename the column

cust_vehicle(cutid, vid, date of purchase, dis price)
If we have vehicle table (vid, vname, description)
- (We have to add reg. no before description, and chasis no. at the end).

1) add column

```
alter table vehicle
add reg_no int after vname,
add chasis_num
```

2) add constraint

To add table level constraint (primary key, foreign key, unique, check)

To add foreign key.

```
alter table vehicle
add constraint fk_vid foreign key (vid) references
vehicle (vid)
```

To add primary key

```
alter table customer
add constraint pk_cid primary key (cid)
```

3) drop column

~~drop~~ *

```
alter table cust_vehicle
drop column reg_no;
```

4) drop constraint

```
alter table cust_vehicle
drop constraint fk_vid;
```

5) Modify column data type or constraint

```
alter table vehicle
modify chasis_num int not null
```

To change data type.

```
alter table vehicle
modify description varchar(50)
```

6) Rename table

```
alter table vehicle
rename to myvehicle;
```

7) Rename the column

```
alter table vehicle
change column reg_no registeration varchar(20)
not null                                    (1)
rename column reg_no registeration          (2)
```

- Create table order table

order(orderid, orderdate, bill amt, ~~cust~~id cid)

customer( cid, cname, address, email)

1. Create both tables
2. add bill_amt and cid, description columns in order table
3. add primary key constraint in both table
4. foreign order
5. Change column name bill_amt to billamt
6. Change data type of billamt to decimal(9,2)
7. drop column description from order table
8. drop primary key constraint from order
9. foreign order
10. Rename order table to myorder.

1. Create both tables

```
create table customer1( cid int, cname varchar(20),
email varchar(20) );

create table order1( orderid int, order_dt date)
```

2. add bill_amt, cid and description column in order table

```
alter table order
add bill_amt int,
add cid int, add description varchar(20);
```

3. add primary key in both table

```
alter table customer1
add constraint pk_cid primary key (cid)
```

```
alter table order1
add primary key (cid)
```

4. add foreign key in order table

```
alter table order1
add constraint fk_order_cust1 foreign key(orderid)
references customer1(cid)
```

5. change column name bill_amt to billamt

```
alter table order1
rename column bill_amt to billamt;
```

6. Change data type of billamt to decimal(9,2)

```
alter table order1
modify billamt decimal(9,2);
```

7. Drop column description from order table

```
alter table order1
drop column description;
```

8. Drop primary key from order

```
alter table order1
drop primary key;
```

9. Drop foreign key from order.

```
alter table order1
drop constraint fk_order_cust1;
```

10. Rename order table to myorder

```
alter table order1
rename order1 to myorder
```

* Nested queriers or joins

- To find empno, ename, sal for all emp with sal > 2000.

```
select empno, ename, sal from emp
where sal > 2000;
```

- To find all emp with sal > Blake's sal.

```
select sal 3000 from emp ] (old method)
where ename = 'BLAKE'
```

```
select empno, ename, sal from emp
where sal > (select sal from emp
where ename = 'BLAKE');
```

- Find all emp who are working in dept acc.

```
select empno, ename, deptno from emp
where deptno = (select deptno from deptno
where dname = 'ACCOUNTING');
```

- Find all emp with sal > avg sal of deptno 10

```
select empno, ename, sal from emp
where sal > ( select avg(sal)
from emp where deptno = 10);
```

- Find all emp who are working in same dept of scott or CLARK.

```
select empno, ename, sal from emp
where deptno in (select deptno
from emp
where ename in ('SCOTT', 'CLARK'));
```

- Find all employee with sal > scott's or Clarks salary

```
select empno, ename, sal from emp
where sal > any (select sal from emp
where ename in ('scott', 'clark'));
```

- Find all emp with sal > avg sal of deptno 10

```
select empno, ename, sal from emp
where sal > ( select avg(sal)
from emp where deptno = 10);
```

- Find all emp who are working in same dept of scott or CLARK.

```
select empno, ename, sal from emp
where deptno in ( select deptno
from emp
where ename in ('SCOTT', 'CLARK'));
```

- Find all employee with sal > scott's or Clarks salary

```
select empno, ename, sal from emp
where sal > any ( select sal from emp
where ename in ('scott', 'clark'));
```

Day 8 7-7-21

- Nested Query

When you write query within query then it is nested query.
- Outer query is called as parent query and inner query is called child query.
- Output of child query is used as input to outer query.

~~in is always~~
If inner query gives multiple values then in is used to checks equals to cond$^{n}$.

- If child query is independant query then it is called as nested query and it gets executed only once.
- If child query is dependant on outer query then it is called co-related query and child query will get executed once for each row in the outer query.

- Nested query joins

- Use nested query if you need data only from one table in the output.
- Use joins if you need data from multiple table. Avoid joins as ~~well~~ much as possible because it is time consuming.
- Maximum level of nesting is 255.

- Find all emp with sal = either blake salary or smith salary.

```
select empno, ename, sal from emp
where sal in (select sal from emp
where ename in ('BLAKE', 'SMITH'));
```

- Select all emp with sal > all sal of avg sal of dept 10 and 20.

```
select empno, ename, sal from emp
where sal > all ( select avg(sal) from emp
where deptno in (10,20)
  group by deptno )
```

If nested query gives multiple values then
where sal > any (3100, 2100) any sal's.
where sal > all (3100, 2100) all sal.

- select all emp with sal > ~~min~~ avg sal of deptno 10 and < max sal of deptno 20.

select empno, ename, sal from emp
where sal between (select avg(sal) from emp
where deptno=10) and (select max(sal) from emp
where deptno=20)

* Nested queries in DML

-Update martin's sal with blake's sal.

update emp ~~empno, ename,~~
set sal = (select sal from emp) s
where ename = 'BLAKE')
where ename = 'martin'

- Change sal of all emp who working in ~~dept~~ at any dept to sal of miller.

update emp
set sal = (select sal from (select sal
from emp
ename = 'miller') s)
where deptno = (select ~~sal~~ deptno from ~~emp~~ (select deptno
~~where~~ from emp
where ename = 'ALLEN') d

- Delete all emp who are working in blake's dept and sal > 3000

```
delete from emp
where deptno = (select deptno from
(select deptno from emp
where ename = 'BLAKE') d) and sal > 2000;
```

- Delete all emp who are working in Allen's dept and sal > avg (sal of dept 10).

```
delete from emp
where deptno = (select deptno from (
select deptno from emp
where ename = 'ALLEN') s) and
sal > (select avg(sal) from (select sal from emp
where deptno = 10) a );
```

* Co-related Queries
- Find all emp with sal > min (sal) of its own deptno.

```
select empno, ename, sal
from emp e
where sal > (select min(sal) from emp d
where d.deptno = e.deptno)
order by deptno;
```

- Find all emp with sal > avg sal of all emp who work under his manager.

```
select empno, ename, sal
from emp e
where sal > (select avg(sal) from emp a
where a.mgr = e.mgr);
```

* Exists, not exists.

exists :-
If returns true if child query returns one or more rows otherwise returns false.

not exists:-
If returns true if child query do not return any rows otherwise returns false.

- Display all dept in which no emp are there.

```
select deptno, dname
from dept d
where not exists (select * from emp e
where e.deptno = d.deptno);
```

- Display dept in which emp are there

```
select deptno, dname
from dept d
where exist ( select * from emp e
where e.deptno = d.deptno);
```

- To see which are not present products

```
select cid, cname
from category c
where not exists (select *
 from product p
where p.cid= c.cid);
```

- Display all emp who are not working as managers.

```
select *
from emp e
where not exists (select * from emp e1
where e.empno = e1.mgr);
```

- faculty (fid, fname, skills)

| | | |
|---|---|---|
| 10 | Rohit | Java |
| 20 | Sarika | java |
| 30 | Narendra | DBT |
| 40 | Tejshree | securit |

- room (rid, rname, rloc)

| | | |
|---|---|---|
| 100 | lotus | 1st floor |
| 200 | jasmin | 1st floor |
| 300 | mogra | 2nd floor |

course (cid, cname, rid, fid)

| | | | |
|---|---|---|---|
| 1 | eDAC | 100 | 10 |
| 2 | eDBDA | 200 | 30 |
| 3 | eDITIS | | |
| 4 | eDAI | | |

- To Find all vacant room.

```
select *
from room r
where not exists ( select * from course c
where c.rid = r.rid);
```

- To find all facultyies who are not assigned to any course & skill is python.

```
select *
from faculty f
where not exists (select * from course c
where c.fid= f.fid) and skill= 'python' ;
```

- To find depts in which no empts are there and dept loc is pune.

```
select deptno,dname
from dept d
where not exists( select * from emp e
where e.deptno = d.deptno)
and loc = 'PUNE';
```

Day 9 8-7-21

* Nested Queries
* Co-related quaries and nested quaries

| Nested Query | Co-related query |
| --- | --- |
| 1) Is not dependant on parent query. | 1) Dependant on parent query. |
| 2) The child query get executed only once. | 2) child query gets executed once per each row in parent query. |
| 3) Exist & not exists cannot be used. | 3) We use exists & not exists operator to check whether child query returns row or not. |

- Maximum level can be 255.

* Joins

When to use joins

1. To display data from multiple table

Types of joins

1. Cross join
2. Inner join
   a. equi join
   b. non equi join
   c. self join
3. Outer join
   a. left outer join
   b. right outer join
   c. full outer join .. this is not directly supported in mysql.

In from if we add two table then it is Joins.

- To display emp details & dept in which employee works.

```
select empno, ename, sal, emp.
e.deptno, d.deptno, dname
  from emp e, dept d
  where e.deptno = d.deptno;
```

* Cross Join (No condition is given)
- If emp table contains 17 rows if dept table contains 4 rows then cross join contain 68 rows.

* Inner Join

If we add condition to get required data then it is Inner join.

```
select *
from product p, category c
where p.cid = c.cid;

select empno, ename, sal, d.deptno, dname
from emp e inner join dept d on
  e.deptno = d.deptno
```

( We have give condition is 'on' if we we inner join.)
( Join cond$^n$ is written in 'on' cond$^n$ & filter cond$^n$ is written in 'where').

- In joins if you are joining n tables then (n-1) conditions will be there.

-Display all product pid, pname, qty, price, cname, description for all product with price >50.

```
select pid, pname, qty, price, cname, cdesc
from product p, category c
where p.cid = c.cid and price >50;
```
(1)

```
select pid, pname, qty, price, cname, cdesc
from product p inner join category c on
p.cid = c.cid
where price >50;
```
(2)

vehicle (vid, vname, chassisnumber)
customer( cid, cname, address)
cust_vehicle ( custid, vehid, dop, price)

(1)
```
select vid, vname, cid, cname
from customer c, vehicle v, cust_vehicle cv
where c.cid = cv.custid and v.vid = cv.vehid;
```

(2)
```
select vid, vname, cid, cname
from customer c inner join cust_vehicle
on c.cid = cv.custid inner join vehicle v
on v.vid = cv.vehid;
```

Example of non-equi join

-Display empno, ename, sal, grade of all emp.

```
select empno, ename, sal, grade
from emp e inner join salgrade s on
e.sal between s.losal and s.hisal;
```

-Display empno, ename, dname, salgrade

(1)
```
select empno, ename, sal, grade, dname
from emp e, salgrade s, dept d
where e.deptno = d.deptno and sal between
losal and hisal;
```

(2)
```
select empno, ename, salgrade, dname
from emp e inner join dept d on
e.deptno = d.deptno inner join salgrade s
on sal between losal and hisal.
```

-To combine table with itself is called self join.

(1)
```
select e.empno, e.name, m.empno mgrno,
m.ename mnane
from emp e, emp m
where e.mgr = m.empno;
```

(If we combine table with that table itself it is self join).

(2)
```
select e.empno, e.name, m.empno mgrno,
m.ename mname
from emp e inner join emp m
on e.mgr = m.empno
```

* Outer join ( To get non matching rows )

```
select * from emp;
select * from dept;

// select empno,ename,e.deptno,dname
// from emp e, dept d.
// where e.deptno = d.deptno;

insert into emp (empno, ename, sal, job) values
(1111, 'Ashwini', 2345, 'ANALYST');
```

- To bring matching as well as non matching rows in the o/p then use outer join.

```
select empno, ename, e.deptno, d.deptno, dname
from emp e right join dept d
on e.deptno = d.deptno;

select empno, ename, e.deptno, d.deptno, dname
from dept d left join emp e
on e.deptno = d.deptno
```

- Find dept which do not have emp

```
select empno, ename, e.deptno, d.deptno, dname
from dept d left join emp e
on e.deptno = d.deptno
where ename is null;
```

Full join :-

In MySQL we have to get union of left join and right join.

-To display all matching rows as well as non matching row from

```
select empno,ename,e.deptno,d.deptno,dname
from emp e right join dept d
 on  e.deptno = d.deptno
union
select empno,ename, e.deptno, d.deptno, dname
from dept d left join dept d
 on  e.deptno= d.deptno
```

```
insert into product (pid, pname, qty, price) values
(145, 'bread', 30, 40.00);
```

```
select * from product;
```

```
 select *
from product p left join category c on
p.cid= c.cid    union
select *
from product p  right join category c on
 p.cid = c.cid
```

faculty (fid, fname, skills)
room ( rid, rname, loc)
course ( cid, cname, rid, fid)

- To find course and faculty assigned to course

```
select cid, cname, fid, fname
from course c inner join faculty f
on c.fid = f.fid;
```

- To find all rooms assigned to course

```
select cname, rname
from course c inner join room r
on c.rid = r.rid;
```

- To find all courses, room assigned to course and faculty assigned to course.

```
select cname, rname, fname
from course c inner join room r
on c.rid = r.rid inner join faculty f
on c.fid = f.fid;
```

- Display all rooms assigned to course as well as rooms not assigned to course

```
select cid, cname, rid, rname
from course c right join room r on c.rid=r
```

-Display all rooms assigned to course as well as not assigned to course and course which do not have any room.

```
select rid, rname, cid, cname
from course c left join room r on c.rid=r.rid
union
select rid, rname, cid, cname
from course c right join room r on c.rid=r.rid;
```

Day 10 9-7-21

- List all faculties who are not allocated on any course and rooms which are not allocated to any course

```
select fid, fname, cname, rname
from faculty f left join course c
on f.fid = c.fid left join room r
on c.rid = r.rid union
select f.fid, fname, cname, rname
from faculty f left join course c on
f.fid = c.fid right join room r on
    c.rid = r.rid
```

-To insert records from existing table

If dept table is there and it contains 3 rows if you want to store the data into another.

create table mydept (
id int,
name varchar(20) )

- Create table using existing table & also populate data

create table mydept as
select * from dept
where deptno > 20

- To get only table structure without data

create table mydept as
select deptno, dname from dept
where 1=2 ;

- To insert data using nested query

insert into mydept (deptno, dname)
select deptno, dname
from dept;

- If you want to create temporary table

create temporary table mytemp
(id int,
name varchar(20) );

* Storage Engines

InnoDB
MyISAM
Archive
Federated

(By default InnoDB will get selected)

InnoDB :- It is most widely used if you want to perform secure transaction,
- it support ACID property
- supports row level locking.

MyISAM :- This is origin engine.
- It is fast storage engine, usually used in data warehousing or if you need table level locking

```
create table mytab (
int id, name varchar(20)
) ENGINE='MyISAM'
```

- To change engine

```
alter table mytab ENGINE='InnoDB'
```

- To know which engine using.

```
select Engine from information_schema.tables
where table_name='dept';
```

* Indexes in mySQL

Indexes are created automatically for primary key and unique.

## Create index

```
create index my_sal_idx
on emp(sal);
```

| empno | ename | sal | deptno | desg |
|---|---|---|---|---|
| 12 | Raja | 34000 | 20 | Clerk |
| 14 | Rashmi | 45000 | 30 | Salesman |
| 15 | Anil | 150000 | 10 | CEO |
| 16 | Swapnil | 25000 | 20 | Programmer |

```
select * from emp where
sal between 20000and 30000.
```

my_sal_idx

| sal | position |
|---|---|
| 25000 | 4,5 |
| 34000 | 1 |
| 45000 | 2 |
| 150000 | 3 |

## Drawback :-

1. DML operations will become slow.
2. memory requirements will grow.

## Advantages / Why to use index

1. To run queries faster which uses where clause or order by clause.

2. Optimize the query execution for group by clause.
3. Finding min() and max() faster.

- To create composite index.

```
create index my_sal_idx
on emp(sal desc, job)
```
(desc = descending)

- Create index using alter table

```
alter table emp
add index(sal)
```

* Types of indexes

1. Unique index :- This does not allow duplicate values in column.

```
create unique index passport_idx
on emp (passport)
```

2. Primary key :- To create this index automatically add primary key constraint on table.

3. Regular or Normal Index :-

```
create index sal_idx
on emp (sal,job)
```

4. Full text :-

fulltext
create index sal_idx
on emp (job, sal)

There indexes helps to search certain words, in large text.
- These indexes are uses in e-commerce site, search engines
- Fulltext indexes are supported by InnoDB, MyISAM can be created only on column of type char, varchar, text.

this is test. Test of database using engine InnoDB.

5. Spital index :- Not widely used
- These are created on column which may contain most of values null and we want to add only not null values in the column.

create spital index sal_idx
on emp (sal, job)

6. Descending index

create index sal_idx
on emp ~~(sal, job)~~ (sal desc, job);

- To see all indexes.

show indexes from emp;

- To drop index

drop index indexname on tablename

- To check which index is used in table for query.

(1) explain select * from emp where ename = 'BLAKE'
(2) select * from emp
use index (ename_idx, sal_idx)
where ename = 'BLAKE'

* Views

Why to create view

1. To give only restricted information
2. To hide complexity of query
3. To increase security - by hiding tablename.

create view mgr10
as select empno, ename, job from emp
where deptno = 10;

select * from mgr10;

Here mgr10 is a virtual table.

insert into emp (empno, ename, job, hiredate, deptno) values (111, 'Devendra', 'Analyst', '2000-04-25', 10);

select * from emp

select * from mgr10;

If we add the emp in the emp table i.e in the main table it will be automatically added in the virtual table.

- To Drop the view

drop view mgr10;

It will delete the mgr10 i.e: virtual table

create view mgr10
as
select empno, ename, job, sal, deptno from emp where deptno = 10;

insert into mgr10 values (222, 'Swapnali', 'analyst', 3456, 10);

select * from mgr10;

select * from emp;

(Entry of new emp i.e 'Swapnali' is added in the main table as well

- We can add entry through virtual table as well

We can insert the data in view if it based on single table.

- To allow to add only rows with deptno=10

```
create view mgr10
as
select empno,ename,job from emp
where deptno=10
with check option;
```

```
insert into mgr10 values(223,'sonali','analyst',
3457,10);
```

- Above entry will be added in table.

```
insert into mgr10 values(224,'swapnil','analyst',
3457,20);
```

- Above entry will not be added because in check condition deptno is not matching).

```
create view myempdept
as
select empno,ename,sal,dname
from emp e inner join dept d
on e.deptno = d.deptno;
```

```
insert into myempdept values(123,'xxx',3456,'HR');
```

- Above entry will not be added because it is based on two ~~condition~~ tables).

emp_india(empno,ename,sal,job,location)
emp-US(empno, ename, sal, job, location)
emp-japan(empno, ename, sal, job, location)

```
create view allemp
as
select * from emp-india
union
select * from emp-us
union
select * from emp-japan

select * from allemp;
```

```
create view allemp
as
select * from emp_india
union
select * from emp_us
union
select * from emp_japan

select * from allemp;
```

Day 11 10-7-21

- To list all view in mySQL

```
Select table_name from information_schema.tables
where table_type like 'view' and table_schem
= 'iacsdedacmay21';

create view myview
as
 select deptno, max(sal), min(sal), avg(sal),
count(*)
from emp
where job = 'analyst'
group by deptno
 having count(*) >=2;

select * from myview;
```

```
create view managerview
as
select * from emp
where job= 'manager'
with check option;
```

- Materialized view (It is not working mysql working in oracle)
for budget purpose

```
create materialized view myview as
select * from product
where type = 'consumable'
```

- materialized view is virtual table, in which the nested query is stored in the database and we can get that stored data (not updated data).

- How to find nth record in mysql

Uses limit clause

```
select * from emp
limit 12;
```

- How to find 12th record in mysql

```
select * from emp
limit 11, 1;
```

- To display highest paid emp.

```
select * from emp
order by sal desc
limit 1;
```

- To display 3rd hightest paid emp

```
select * from emp
order by sal desc
limit 2,1;
```

- To find nth highest in oracle.

```
select sal from emp e1
where n-1= (select count (distinct (sal)) .
            from emp e2
            where e1.sal > e2.sal)
```

* DCL (Data Control language)

grant <previleges> on <tablename> to <username> @ <server ip>

all - all previleges
public(*) - all employee

= To assign permission to all emp.

```
grant all on noticetab to '*'@'localhost'
```

- To revoke previleges.

revoke <previleges> on <tablename> from

'user' @ 'localhost'.

| | |
|---|---|
| Select | Delete |
| Insert | Update |
| Index | Create |
| Alter | Drop |
| All | Grant option |

grant select, insert on category to 'ul' @ 'localhost' with grant option (with grant option ul can also give the same permission to other user).

- To revoke the permission.

revoke select, insert on category from 'ul'@'..'

* PL-SQL (Procedural language)

if statement, loops, variable declaration, procedure, functions, triggers, exception handling

* Normalization :-

1NF, 2NF, 3NF, BCNF (Boyce code NF)

To divide the data into multiple table to reduce reduancy is called normalization.

| Cust id | Cname | address | A/c no | balance | Type | relmgr |
|---|---|---|---|---|---|---|
| 1 | Kishori | Aundh | 1 | 1234 | Saving | AA |
| 1 | Kishori | Baner | 2 | 51 | current | AA |
| 1 | Kishori | Baner | 3 | 45 | Demat | AA |
| 2 | Rajan | Aundh | 4 | 1243 | Saving | BB |
| 3 | Revati | Aundh | 11 | 5421 | curent | BB |
| 4 | Sachin | Deccan | | | | |

Insertion anamoly
Updation anamoly
Deletion anamoly

- Instead of keeping everything in single table we can create or make them in small tables then insertion, updation deletion anamoly these drawback will be removed and reduancy will be removed.

Normalization ---: Data modeling, E-R diagram

1) 1NF :- If every row and column in the table contains atomic values (single value)

2) To check table is in 2NF

- The table should be in 1NF
- The table should not have any partial dependancy.

Partial dependancy → if any non prime attribute (the attribute which is not part of candidate key) is not dependent of candidate key.

Candidate key :-

Non prime attributes → subject name, marks, student name

Prime attributes → student id, subject id

Day 12 12-7-21

many doctors in the hospital

| patientid | pname | date | time | drid | dname | speciality |
|---|---|---|---|---|---|---|
| 1 | Rishabh | 12-7-21 | 9:00 am | 100 | Sanjay | Orthopedic |
| 1 | Rishabh | 12-7 | 2:00pm | 200 | Archana | Neuro |
| 1 | Rishabh | 13-7 | 2:00pm | 200 | Archana | Neuro |
| 1 | Rishabh | 13-7 | 4:00pm | 200 | Archana | neuro |

Patient id + drid + date + time = Candidate key

Functional Dependiences

Table is in 1NF — yes
We want to find is it in 2NF.

~~patient id + drid + date + time~~
prime attribute → patient id + drid + date + time
~~patient id →~~
Non prime attributes → pname, dname, address, speciality, receptionist

patient id → pname, address

drid → dname, ~~speciality~~ speciality

patient id + drid + date + time → receptionist

| patient id | pname | address |
|---|---|---|
| 1 | Rishabh | Aundh |

| drid | dname | speciality |
|---|---|---|
| 100 | Sanjay | Ortho |
| 200 | Archana | neuro |

| patient id | date | time | drid | receptionist |
|---|---|---|---|---|
| 1 | 12 July | 9 am | 100 | Ashu |
| 1 | 12-7 | 2pm | 200 | Ashu |
| 1 | 13-7 | 2pm | 200 | Deepa |
| 1 | 13-7 | 4pm | 200 | Deepa |

* To check whether it is in 3NF.

1. The table should be in 2NF.
2. There should not be any transitive deperdancy.

$x \rightarrow y \rightarrow z$

studid sname address courseid cname cdesc

studid → courseid → cname

studid → course id → cdesc

(course id, cname, desc)

* Check the table in 4NF (BCNF)

1. Table should be in 3NF.
2. for dependancy $x \rightarrow y$ then x should be super key
x cannot be non prime attribute if y is prime attribute

one student can take many courses.
one faculty can teach only one course

| studid | subject | faculty |
|---|---|---|
| 1 | Java | Rahmi |
| 1 | C++ | Deepa |
| 2 | Java | Rajan |
| 2 | DBMS | Tejas |
| 3 | Java | Rahmi |

fname → subject

| fid | subject | faculty |
|---|---|---|
| 1 | Java | Rashmi |
| 2 | C++ | Deepa |
| 3 | Java | Rajan |
| 4 | DBMS | Tejas |

| studid | fid |
|---|---|
| 1 | 1 |
| 1 | 2 |
| 2 | 3 |
| 2 | 4 |
| 3 | 1 |

- One employee works on many projects

| Proj code | Proj Type | Proj Desc | emp no | ename | Grade | Sal scale | Proj DOJoin | Alloc Time |
|---|---|---|---|---|---|---|---|---|
| 001 | AAP | LNG | 46 | Jones | A1 | 5 | 12/1/98 | 24 |
| 01 | AAP | LNG | 92 | Smith | A2 | 4 | 2/1/99 | 24 |
| 01 | AAP | LNG | 96 | Black | B1 | 9 | 2/1/99 | 18 |
| 04 | MAI | SHO | 72 | Jack | A2 | 4 | 2/4/99 | 8 |
| 04 | ~~AAP~~ 002 | SHO | 92 | Smith | A2 | 4 | 5/5/99 | 6 |
| 002 | ~~AAP~~ XXXX | LNG | 72 | Jack | A2 | 4 | 12/1/98 | 12 |

Is table is in 1NF --- yes

To check table is in 2NF

prime attributes → proj code + empno → proj joining date, alloc time

proj code → proj type, proj desc.

empno → ename, grade, sal scale

| Proj code | Proj Type | Proj Desc |
|---|---|---|
| 001 | AAP | LNG |
| 001 | AAP | LNG |
| 001 | AAP | LNG |
| 004 | MAI | SHO |
| 004 | SHOMAI | SHO |
| 002 | APP | LNG |

we can remove duplica to reduce reduancy

| emp no | ename | Grade | Sal scale |
|---|---|---|---|
| 46 | Jones | A1 | 5 |
| 92 | Smith | A1 | 4 |
| 96 | Black | B1 | 9 |
| 72 | Jack | A2 | 4 |
| 92 | Smith | A2 | 4 -x |
| 72 | Jack | A2 | 4 -x |

| Proj code | emp no | date of joining | Alloc Time |
|---|---|---|---|
| 001 | 46 | 12/1/98 | 24 |
| 001 | 92 | 12/1/98 | 24 |
| 001 | 96 | 2/1/99 | 18 |
| 004 | 72 | 2/4/99 | 6 |
| 004 | 92 | 5/5/99 | 6 |
| 002 | 72 | 12/1/98 | 12 |

Check for 3NF

empno → grade → sal scale

| Grade | sal scale |
|---|---|
| A1 | 5 |
| A2 | 4 |
| B1 | 9 |
| A2 | 4 → |

| emp no | ename | Grade |
|---|---|---|
| 46 | Jones | A1 |
| 92 | Smith | A2 |
| 96 | Black | B1 |
| 72 | Jack | A2 |

| Proj code | empno | Proj date of joining | Alloc Time |
|---|---|---|---|
| 001 | 46 | 12/1/98 | 24 |
| 001 | 92 | 2/1/99 | 24 |
| 001 | 96 | 2/1/99 | 18 |
| 004 | 72 | 2/4/99 | 6 |
| 004 | 92 | 5/5/99 | 6 |
| 002 | 72 | 12/1/98 | 12 |

* E-R Diagram :-

proj (pid, proj type, proj desc)

employee (empno, ename, grade)

grade (grade, sal scale

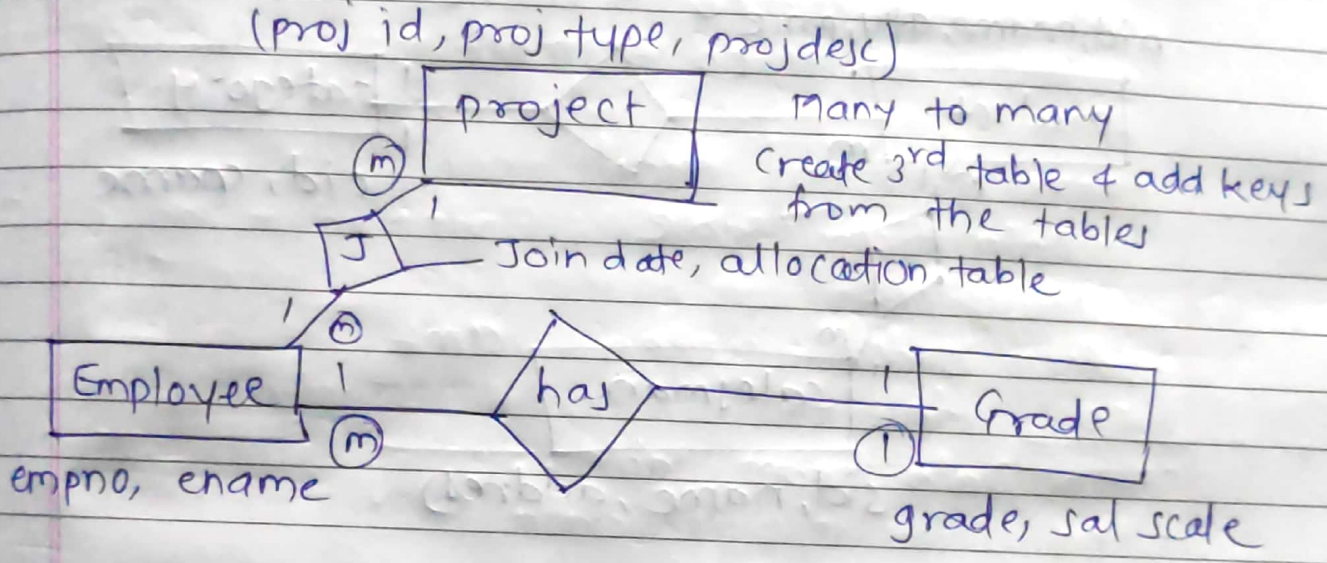
proj-emp (proj id, empno, joining date, alloc time)

one to one
one to many
many to one ] - same
many to many

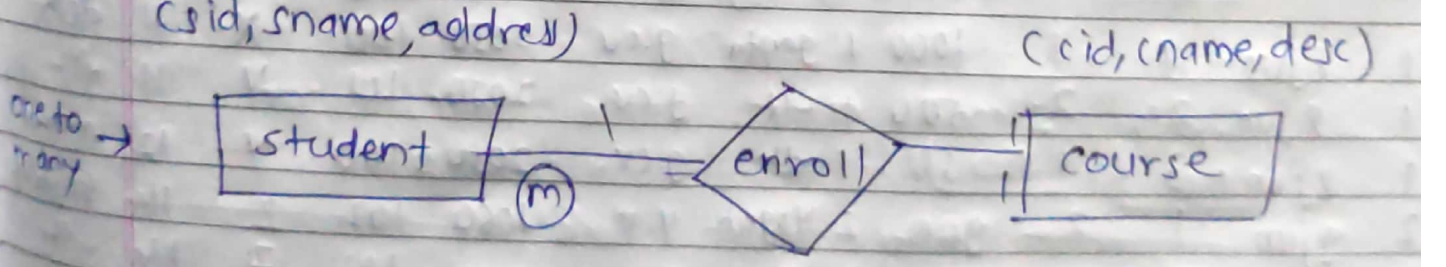
Data modelling --- structured data

ER diagram

proj, emp, grade




One to many - The key of one side should be added in many side table

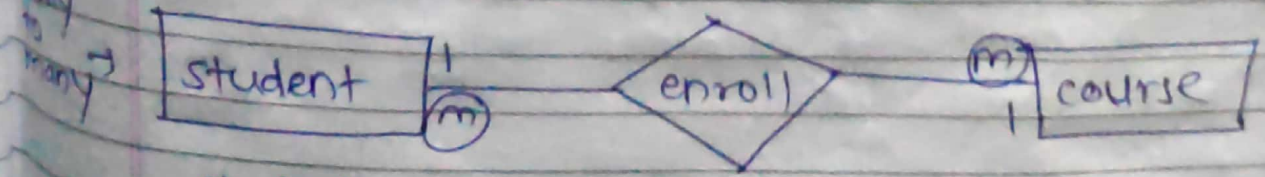



student (sid, sname, address, cid)

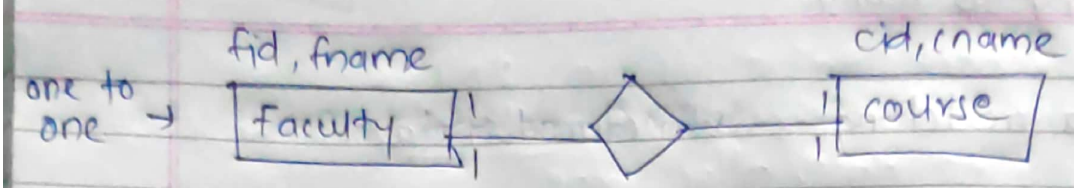
course (cid, cname, desc)

sid, sname, address

cid, cname, desc

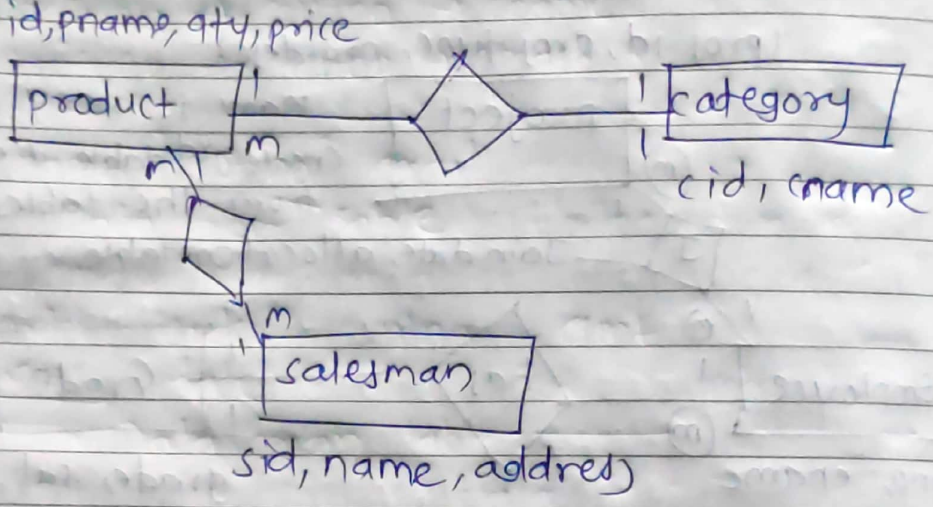



Student (sid, sname, address)
course (cid, cname, desc)
Stud_course (sid, courseid)




one to one →

key of any one of the sides will go to other




| Order no | Order date | Item no | qty | price | cut no → cname | email | cust | Sales person no | sname | locid | lname |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 8 jun | 1 | 1 | 1000 | 1 gayti | g.cv | 2050 | 100 | X | 11 | delhi |
| 1 | 8 jun | 2 | 2 | 500 | 1 -11- | g.cv | 2050 | 101 | Y | 11 | -11- |
| 1 | 8 jun | 3 | 1 | 50 | 1 -11- | g.cv | 2050 | 100 | Y | 12 | mumbai |
| 2 | 9 jun | 1 | 2 | 900 | 2 yogesh | y.y | 1060 | 102 | Z | 14 | Gujarat |
| 2 | 9 jun | 5 | 1 | 60 | 2 -11- | y.y | 1060 | 102 | Z | 14 | -11- |
| 3 | 9 jun | 5 | 1 | 60 | 3 sagar | s.y | 4200 | 100 | X | 100 | X |
| 3 | 9 jun | 10 | 2 | 70 | 3 -11- | s.y | 4200 | 102 | Z | 14 | Gujarat |
| 3 | 9 jun | 2 | 4 | 1000 | 3 -11- | s.y | 4200 | 100 | X | 100 | X |

The table is in 1NF

Check → is it in 2NF

candidate key (order no + item no)

prime attribute - order no, item no

Non prime - Remaining

orderno + item no - qty, price, salesman no, sname, lname, loc id.

item no -

order no - order date, cust name, cust no, email, amt

order

Order no | order date | cname | email | amt

—— Data from table ——

Order no | Item no | qty | price | salesperson no | sname | locid | lname

——— Data from table ———

Are they in 3NF

check for transitive dependience

Order table is not in 3NF.
orderno → custno → cname
orderno → custno → email

| cname | custno | email |
|---|---|---|

Data from Table

| Order no | order date | custno | amt |
|---|---|---|---|

Data from table

In order item table

order id + item id → salesmanid ↛ sname
→ locid → lname

| Order no | Item no | qty | price | salesperson no | locid |
|---|---|---|---|---|---|

Data from Table

| Salesperson no | sname |
|---|---|

Data from Table

| locid | lname |
|---|---|

Data from Table

* PL-SQL

Procedural language :-

if, loops, ~~executions~~ exceptions, cursors, procedures, functions, triggers)

Advantages of PL-SQL

- To increase security.
- To hide the complexity of the query.
- To reduce network traffic
- To increase the speed.

Day 13 13-7-21

Code blocks in mySQL

3 Types of code block

1. Procedure :- -This is block of code which does not return any value. -no return statement
- Procedures can not be used in select statement

2. Function :-

- We can function in select statement
- return single value

3. Trigger :-

- The block which gets called implicity automatically after or before some DML operation happens on the table.

* How to write Procedure

1. Write a procedure to insert data into table.

```
insert into dept values (100, 'HR', 'mumbai');
delimiter //
create procedure insertDeptRec(pdno int,
pdnm varchar(20), pdloc varchar(20))
begin
insert into dept values(pdno, pdnm, pdloc);
end; //
delimiter;
```

This is in java

```
call insertDeptRec(17, 'HR', 'Mumbai');
```

- This is for inserting data

- To see procedures in database.

```
show procedure status where db = 'iacsdedacmay21';
```

- Write a procedure to find no of emp in dept 10.

```
select count(*) from emp
where deptno = 10;
```

Within a procedure you can pass parameter of 3 type

1. in - when you want to pass data as i/p to proced
2. out - This are readonly parameter
3. inout - By defaults the parameters of in type.

in = i/p
out = o/p
in out = i/p & o/p

2. out :-
- When you want to get o/p from procedure then you can use out type parameter
- These are write only parameters.

3. inout :-
- These parameters can be used to pass data as i/p and also get the o/p.
- These are read write parameters.
- We can modify values of parameter and we can also pass the i/p to procedure.

```
delimiter //
create procedure getempcnt(in pdno int,
out pcnt int)
begin
select count(*) into pcnt from emp
  where deptno = pdno;
end //
delimiter ;
call getempcnt(10,@cnt);
select @cnt;
```

- To pass count to procedure and increase count by 10

```
delimiter //
create procedure increasecnt(inout pcnt int)
begin
  set pcnt = pcnt+10;
end //

set @cnt = 12
call increasecnt(@cnt)
```

- To find ename, job, deptno, sal of a emp whose id is given.

```
delimiter //
create procedure getempdata (in empeid,
out pename varchar(20), out psal decimal(9,2),
out pdno int, out pjob varchar(20))
begin
select ename, job, sal, deptno into
pename, pjob, psal, pdno from emp
where empno = peid;
end //

delimiter ;

call getempdata (7902, @enm, @job, @s, @dno);

select @enm, @job, @s, @dno;
```

* Rules for select .... into statement

1. This can be used inside procedure or function.
2. select .... into this query should return only one row.
3. Number of columns in select statement and number of variable after into should be same.

- To write a procedure to calculate bonus.

sal < 2000 bonus = sal * 10% + comm ~~* 10%~~
otherwise bonus = sal * 15% + comm ~~* 15%~~

syntax of if

If

```
if condition then
   statements
else
   statements
end if
```

else-if

```
if condition then
   statements
elseif condition then
   statements
else
   statements
end if
```

```
delimiter //
create procedure calcBonus(pnm varchar(20))
begin declare vsal, vcomm, vbonus int
select sal into vsal, vcomm from emp
where ename = pnm
if ~~vsals~~ vsal < 2000 then
   set vbonus = vsal * 0.10 + ifnull(vcomm, 0) * ;
else
set vbonus = vsal * 0.15 + ifnull(vcomm, 0);
end if;
select pnm, vsal, vcomm, vbonus;
end //
delimiter ;
call calcBonus('SMITH');
```

* Scope of variable

1. Local variable :-

The variables declared inside procedure are called as local variables and can be used only inside procedure.

2. Session variables :-

- These are variables whose name starts with @ and are called as session variables.
- These variables are accessible till logout.

* Loops in PLSQL

1. while --- top tested loop

```
while condition do
    statements
  end while;
```

2. Repeat loop is called as bottom tested.

```
Repeat
 statements
until condition
 end repeat;
```

3. loop ... end loop

```
loop
statement
```

```
if condition then
    leave label
end if
end loop
```

leave → break statement in java
iterate → continue

- To test while loop
1, 2, 3, 4, 5.

```
delimiter //
create procedure test_while()
begin declare i int default 0;
declare data varchar(20);
set data = '';
while i<=5 do
    set data= concat(data, i, ',');
    set i=i+1;
end while;
select data;
end //

delimiter ;

call test_while();
```

- To declare variable

```
declare x int default 0;
declare y int default 20;
declare y int;
```

```
declare x,y int;
declare vd date;
declare vnm varchar(2);
```

* Repeat until

1. Bottom tested loop
2. It gets executed minimum once.
3. It gets executed until given condition is false, as soon as condition become true it terminate loop.

- To print 1,2,3,4,5.

```
delimiter //
create procedure test_repeat()
begin
declare i int default 1;
        data varchar(20);
set data= ' ';
Repeat
  set data= concat(data,i,',');
  set i=i+1;
until i<=8 i>5
 end repeat;
select data;
 end //
   delimiter ;
call test_repeat();
```

iterate - It is same as continue.

```
label 1 : loop
    statements
if condition then
   iterate label 1;
else
   leave label 1;
end if
end loop
```

To print 1,2,3,4,5.

```
delimiter //
create procedure test_loop()
 begin
declare i int default 1;
declare data varchar(20) default '';
 label 1:
    xyz: loop
 if i >5 then
    leave xyz;
 end if
set data=concat(data, i, ',');
set  i=i+1;
 end loop;
 end //

 delimiter ;
 call test_loop();
```

- Write a procedure to display all numbers between 1 to 20 which are divisible by 5.
5, 10, 15, 20

```
loop
if x mod 5 = 0 then
```

```
delimiter //
create procedure divi_test()
begin
dechre i int default 1;
declare data varchar(20) default '';
label 1: loop
  if i>20 then
    leave label 1;
  end if;
  if i mod 5 = 0 then
  set data=concat(data, i, ',');
  end if;
  set i=i+1;
end loop;
select data;

delimiter ;
call divi_test();
```

- Write a procedure to calculate experience of emp whose name is given.

```
delimiter //
create procedure calexp(pname varchar(20),
out pexp int, out pdate date)
```

```
begin
declare i int default 1
select hiredate into pdate from emp
where ename = pname;
 set pexp = floor (datediff(hiredate, curdate())
/365 );
 select pexp;
 end //

 delimiter ;
call calexp('SMITH', @e, @dt);
select @e, @dt;
```

- To drop procedure

```
drop procedure calexp;
```

```
begin
declare i int default 1
select hiredate into pdate from emp
where ename = pname;
set pexp = floor (datediff(hiredate, curdate())
/365 );
select pexp;
end //

delimiter ;
call calexp('SMITH', @e, @dt);
select @e, @dt;
```

- To drop procedure

```
drop procedure calexp;
```

Day 14 14-7-21

- We have a table mytab (color 1, color 2, pname)

| pname | color 1 | color 2 |
|---|---|---|
| table | brown | walnut |
| chair | white | brown |

- Write a procedure to find color available for given product.

i/p name display color in which product is available and the color that you are getting

Find other product with same color as color2
option

```
delimiter //
create procedure findcolor ( pname varchar(20),
out pcolor varchar(20) )
begin declare vc1, vc2, vname2 varchar(20);
select pname, vc1, vc2 into pname, vc1, vc2
from mytab
where col1 name = pname;
select name into vname2
from mytab
where col1 = vc1;
select vname2, vc1
select pname, vc1, vc2
end //
```

- Find discounted price for the product based on price.

price = price - discount
if price < 100 discount 15%.
>= 100 bt < 150 10%.
>= 150 8%.

* Cursors in plsql

- Cursors are similar to arrays.
- And is used when select statement within procedure uses multiple rows then we use cursor.
- If we want to read rows line by

Cursors are similar to arrays.

line from a table and process them

* Steps to use cursor

1. declare cursor

This is datatype for cursor

declare <cursor_name> cursor from select statement (this returns multiple rows)

Exception handler →

declare continue handler for not found
set cset = 1.

2. Open cursor --- When you open the cursor then the query will get executed and the data will be populated within cursor.

open <cursor_name>

3. Read next row from the cursor.

fetch <cursor_name> into <list of variable>

4. Check whether we have reached to last row

if cset = 1 then
leave label 1
end if

5. Process the data

6. Repeat step 3 to 5 till you reach end of cursor.

7. close cursor

close <cursor_name>

- Write a procedure to display all employee names and hiredate for all emp working in deptno = 10.

```
create procedure display_empdata() begin
declare cset int default 0;
declare vname varchar(20),
declare vdt date;
declare empcur cursor for select ename,
hiredate from emp where deptno=10;
declare continue handler for NOT FOUND
set cset =1;
open empcur; label1 : loop
fetch empcur into vname, vdt ;
 if cset =1 then
      leave lable 1;
 end if;
select vname,vdt
end loop
close empcur;
end //

call display_empdata();
```

- Discounted price on product.

```
create procedure calculatediscount()
begin
declare cset int default 0;
declare vnm varchar(20);
declare vprice, vdisp decimal(9,2);
```

```
declare prdcur cursor for select pname, price
from product;
declare continue handler for NOT FOUND
set cset = 1;
open prdcur;        label 1: loop
fetch prdcur into vnm, vprice;
if cset = 1 then
        leave label 1;
end if;
  if vprice < 100 then
set vdispr = vprice - vprice * 0.15;
 elseif vprice >= 100 and vprice < 150 then
set vdispr = vprice - vprice * 0.10;
 else
set vdispr = vprice - vprice * 0.08;
end if;
select vnm, vprice, vdispr;
 end loop;
close prdcur;
 end //
```

- To write a procedure to display emp name and job

```
create procedure dis-empdata()
begin
declare cset int default 0;
declare vnm, vjob varchar(20);
declare data varchar(200) default
declare empcur cursor for select ename,
job from emp;
declare continue handler for NOT FOUND
    set cset = 1;
```

```
open empcur;    label l : loop
fetch empcur into vnm,vjob;
 if cset = 1 then
        leave label l;
 end if;
set data= concat(data,vnm,'---',vjob,',');
  end loop;
  select data;
  close empcur;
  end //

call display_empdata;
```

Output of this in SQL is shown in table and in the java it will be shown ~~as~~ as string.

-Update sal of emp

job manager increase sal by 10%.
job analyst increase sal by 15%.
job clerk increase sal by 30%.
otherwise increase sal by 35%.

```
create procedure update_sal()  begin
declare cset int default 0;
declare vempno, vsal int;
declare vjob varchar(20);
declare empcur cursor for select empno, job,
sal from emp;
 declare continue handler for NOT FOUND
  set cset =1;
```

```
open empcur;
label1 : loop
fetch empcur into vempno, vjob, vsal;
if cset = 1 then
    leave label1;
end if;
if vjob = 'MANAGER' then
    update emp
    set sal = vsal * 1.10
    where empno = vempno;
elseif vjob = 'ANALYST' then
    update emp
    set sal = vsal * 1.15
    where empno = vempno;
else if vjob = 'CLERK' then
    update emp
    set sal = vsal * 1.30;
    where empno = vempno;
else
    update emp
    set sal = vsal * 1.35
    where empno = vempno;
end if;
end loop;
close empcur;
end //

call update_sal();
```

## 2. Functions

- If you want to return single value then you write function.

concat( ), lpad, rpad, substr, datediff

where we can use this functions

select, where inside some procedure.

* How to create user defined function.

```
SET GLOBAL log_bin_trust_function_creators=1;
-- this allows to write function in mysql.
create function calculateexp (dt1 date, dt2 date)
  returns int
  begin
  declare vexp int;
  set vexp = floor(datediff(dt1,dt2)/365)
     return vexp;
    end //

select empno, ename, hiredate, calculateexp(
curdate(), hiredate) experience
 from emp
where calculateexp(curdate(), hiredate) >35;
```

- To see the code for function or procedure.

```
show create function calculateexp;
```

- Write a function to calculate discount.

price <100 discount 10%.
>=100 and <150 discount 15%.
otherwise discount 20%.

```
create function calculatediscount (ppr decimal(9,2))
 returns int
begin declare vdis int default 0;
 if ppr <100 then
    set vdis = 10;
 elseif ppr <150 then
   set vdis = 15;
 else
    set vdis=20;
 endif;
 return vdis;
 end //
```

- To drop function

```
drop function calculatediscount;

select * from product;

select pid, pname, price, calculatediscount(price)
 from product;
```

- We can use user defined functions inside the procedure as well.

- Generate e-mail
concat email with 1st 3 char of job followed by @mycompany.com

```
create function generateemail (pnm varchar(20),
pjob varchar(20)) returns varchar(30)
begin
declare vemail varchar(30);
set vemail=concat (pnm,'.',substr (pjob,1,3),
'@mycompany.com');
return vemail;
end //
```

- Generate e-mail
concat email with 1st 3 char of job
followed by @mycompany.com

```
create function generateemail (pnm varchar(20),
pjob varchar(20)) returns varchar(30)
begin
declare vemail varchar(30);
set vemail=concat (pnm,'.',substr (pjob,1,3),
'@mycompany.com');
return vemail;
end //
```

Day 15 15-7-21

- Calculate net sal for one emp.
DA - 10%
HRA - 15%
pf - 0.08%
Net sal = psal + da + hra - pf

```
create function netsal (psal decimal (9,2)) returns decimal
begin
declare vnetsal, vda, vhra, vpf decimal (9,2);
set vnetsal = psal * 0.1 + psal * 0.08 + psal * 0.15
+psal;
return vnetsal;
end //
```

Call the function

```
select empno, ename, sal, netsal (sal)
  from emp;
```

- Based on comm we need to display performance of use.

comm null or 0 then "need improvement"
comm <300 then ok
comm <500 then good
else excellent

```
create function getperformance(pcomm decimal(9,2))
returns varchar (20)
begin
declare vperform varchar(20) default '';
if pcomm is null or pcomm=0 then
set vperform = 'need improvement';
 else if pcomm<300 then
 set vperform = 'ok'
 elseif pcomm <500 then
 set vperform = 'good'
 else
 set vperform = 'excellent'
end if;
return vperform;
end //
```

- Write a procedure to display empno, ename, sal, comm, performance of all emp of particular dept.

```
create procedure display_empdata (pdeptno int)
begin
select empno, ename, sal, comm, getperformance
(sal) from emp
where deptno = pdeptno;
end //
```

* Triggers:- (It will get called automatically)

Timing

1. before
2. after
3. instead of --- views -- doesnot work in mysql

2 types of trigger

1. statement level:- This is not supported in mysql.
2. row level:-

* Create Trigger

step 1:-
create necessary audit table
step 2:-
create trigger

in audit table if you want to store
deptno, dname, username, date, action

step 1:-
```
create table emp_audit( deptno int,
dname varchar(20), username varchar(20),
dt_action datetime, action varchar(20));
```

step2 :- Write a trigger to audit insert, delete, update information on dept.

```
create trigger insert_dept before insert on dept
for each row
insert into dept_audit values(NEW.deptno,
New.dname, current_user(), now(), 'insert')

create trigger delete_dept before delete on dept
for each row
Insert into dept_audit values(OLD.deptno,
Old.dname, current_user(), now(), 'delete')

select * from dept;

insert into dept values(50, 'xxx', 'mumbai');

select * from dept;

select * from dept_audit;
```

| Deptno | Dname | location |
|---|---|---|
| 10 | HR | Mumbai |
| 20 | Accounts | Chennai |
| 30 | Purchase | Pune |
| 40 | Sales | Pune |

```
delete from dept
where deptno > 20.
```

OLD

| | | |
|---|---|---|
| 30 | Purchase | Pune |
| 40 | Sales | Pune |

insert into dept values(50, 'xxx', 'Pune');

New

| | | |
|---|---|---|
| 50 | xxx | Pune |

Old

Nothing in Old.

update dept
set dloc = 'chennai', dname = 'newsales'
where deptno = 40;

Old

| | | |
|---|---|---|
| 40 | Sales | Pune |

New

| | | |
|---|---|---|
| 40 | newsales | Chennai |

insert into dept values(60, 'yyy', 'mumbai');

select * from dept_audit;

select * from dept;

delete from dept
where deptno > 40;

```
select * from dept;

select * from dept_audit;

create trigger update_dept before update on dept
for each row
insert into dept_audit values( OLD.deptno,
old.dname, current_user(), now(), 'update');

update dept
set dname = 'newsales'
where deptno = 30;

select * from dept;      (new values)

select * from dept_audit; (old values)
```

- To write triggers on product table.

```
create table product_analysis (pid int,
pname varchar(20), oldrate ~~int~~ decimal(9,2),
newrate decimal(9,2), uname varchar(20),
dt_time datetime, action varchar(20));

create trigger update_pr after update on product
for each row
insert into product_analysis values( OLD.pid,
OLD.pname, OLD.price, New.price, current_user(),
now(), 'update');
```

```
select * from product;

update product
set price=100
where pid>100;

select * from product_analysis;
```

= To delete the trigger

```
drop trigger <trigger_name>
```

- To display all triggers.

```
show trigger
```

* Exception handling

handler
1. exit ---- (stop)
2. continue --- (Resume)

Condition
mysql-error_code
SQLEXCEPTION
NOT FOUND

```
declare continue handler for SQLEXCEPTION
select 'Error occured'

declare exit handler for SQLEXCEPTION
set cset=1
```

```
create procedure insertproduct (ppid int,
ppname varchar(20), paty, pprice decimal(9,2),
pcid int)
 begin declare exit handler for SQLEXCEPTION select 'error occurd
insert into product values( ppid, ppname, paty,
pprice, pcid)
end// select 'done';
     end //


create procedure insert_userentity111(puid int,
puname varchar(20), paddrid int)
 begin
declare exit handler for 1062 select 'duplicate userid';
declare exit handler for 1366 select "data type mismatch";
declare exit handler for SQLEXCEPTION select 'error occurd';
 insert into userentity values (puid, puname, paddrid);
 select 'done';
 end//
```

* NoSQL - MongoDB

CRUD --- create, read, update, delete

-Use installation step by step to install mongodb
setup path to
c:\Program Files\MongoDB\Server\4.2\bin

step1 :- start server
open cmd
c:\system32>mongod -dbpath c:\data\db

27017 ← By default server

step 2 :- start client
open cmd
c:\system32> mongo
> show dbs (To list all available databases)

step 3 : import data c:\mydata - store json file
.json
restaurant.json
movie.json — in class
] - in ~~class~~ assignment

open new cmd
c:\system32> mongoimport --db test --collection movie --file c:\mydata\movie.json

c:\system32> mongoimport --db test --collection restaurant --file c:\mydata\restaurant.json

[use test --- create and switch database

show collections -- to list all collections ]
- in mongo client window
>db.movie.find().pretty() --- to see data from movie

> db.restaurant.find().pretty() ——— restaurant

Mongodb - stores data internally in binary-JSON (BSON)

JSON (Javascript object notation)

{
empid: 123,
ename: ~~'Prakash'~~ 'kishori',
skills: ['java', 'Python', 'mongodb', 'spring boot', 'hibernate'],

Table 1

JSON stores in key, value pair.

```
joining-dt : ISODate('2000-04-27'),
 dept : { deptno : 11, dname : 'hr', dloc : 'Pune'},
experience : [{name : 'hsbc', years : 3},
              {name : 'igate', years : 4},
              {name : 'capgemini', years : 5}],
maritalStatus : null }
```

Terminologies

| RDBMS | NOSQL |
|---|---|
| Database | Database |
| Table | collection |
| Record | Document |
| Index | Index |

| RDBMS | NoSQL |
|---|---|
| 1) Structured | 1) Unstructured |
| 2) Vertical scaling | 2) Horizontal scaling |
| 3) Less available compared to NOSQL | 3) Highly available --- replica is available |
| 4) Compare to NoSQL this is slower | 4) Faster - sharding |
| 5) Transaction control (ACID) | 5) No transaction (CAP) |

- Create a database ^iacsdedac collection
  add 5 emp data.

- Create a friend collection
  add 5 documents
  name, bdate, hobbies, address, mobileno

~~mongo~~

Day 16 16-7-21

CAP -- Consistency, Availability, partition tolerance

CA - EricBrewer who say that at a time NoSQL does not support CA.

* NoSQL Databases

Cassandra, CouchbaseDB, MongoDB, NoSQL

- To get the data from JSON file.
  [From table 1]
  "dept.dname"
  "experience.1.name" (int)(1 = array index)
  "skills.3" (spring boot)

In mongoDB '_id' is always primary key.

CRUD - Create, read, update, delete

- To create collection

(1) db.createCollection("mycollection")

(2) db. mycollec. insert ({ name : 'xxx', desig : 'yyy' })

- Capped collection

- Restrict number of document
- You cannot delete data from capped collection.

blog - 5 years ago
comments --- lots comments
latest 100 comments

- To create capped collection

db. createCollection ("mycoll", { capped : true, max : 2 , size : 40000 })

db. mycoll. insert ({ name : "Rajan", comment : "good blog" })

db. mycoll. insert ({ name : "Revati", comment : "excellent" })

db. mycoll. insert ({ name : "Raj", comment : "Good" })

db. mycoll. find()

- To see indexes in collection

db. emp . getIndexes()

- Read data

find() --- function to find all record

findOne() --- To find one record (1st inserted)

To see in sorting

db.emp.find().sort({empno:1}).pretty();

1 for ascending

-1 for descending

- If we don't want to see id. then -id=0.

db.movie.find({}, {name:1, rating:1, price:1, _id:0}).sort({rating:1}).pretty();

- If we want to see all except ticket number.

db.movie.find({}, {ticket_name:0, _id:0}).sort({rating:-1}).pretty();

- To see top 5 record

db.movie.find({}, {name:1, _id:0}).sort({rating:-1}).limit(5).pretty()

- To see 4th highest rating record.

db.movie.find({}, {name:1, _id:0}).sort({rating:-1}).limit(1).skip(3).pretty();

Query → operators

$in, $nin, $eq, $ne, $gt, $lt,

- To list all movie with name is 'Padmavat'

db.movie.find({name:'Padmavat'}, {name:1, rating:1, price:1, _id:0}).pretty();

- To find all movie with rating 3.

db.movie.find({rating:3}, {name:1, rating:1, price:1, _id:0}).sort({price:1}).pretty();

- To find all movie with rating=3 and price=260

(1) db.movie.find({rating:3, price:260}, {name:1, rating:1, ticket_no:1, _id:0}).pretty();

(2) db.movie.find({$and:[{rating:3}, {price:260}]}, {name:1, rating:1, ticket_no:1, _id:0}).pretty();

- To find all movie with price=200

db.movie.find({price:200}, {name:1, rating:1, price:1}).pretty();

$gt = greater than  $ne = not equal to
$gte = greater than equal to
$any = not equal to

- To find all movies with price >200.

db.movie.find({price:{$gt:200}}, {name:1, rating:1, price:1}, pretty()

- To find movie with rating <5

db.movie.find({rating:{$lt:5}}, {name:1, rating:1, price:1}, pretty()

- To find all movies with rating <5 & price >300

db.movie.find({rating:{$lt:5}}, price:{$gt:300}}, {name:1, rating:1, price:1}, pretty()

- To find all movies with rating <5 or price >300

db.movie.find({$or:[{$lt:5}, price:{$gt:300}]}).pretty()

- List all movie with price = 260 or 300 or 450

db.movie.find({price:{$in:[260, 300, 450]}}).pretty();

- List all movies with price not equal to 260, 300 or 450.

db.movie.find({price:{$nin:[260, 300, 450]}}).pretty().

- To list all movies withe price = 260 or 300 or 450 and rating > 3.

```
db.movie.find({$and:[{price:{$in[260,300,450]}},
rating:{$gt:3}]})
```

- To find all movies which do not have rating key.

```
db.movie.find({rating:{$exists:false}}).pretty()
```

false :- not having
true :- having

-Movie having rating and rating is null.

```
db.movie.find({rating:{$in:[null],$exists:true}})
```

- To find all movies with even ratings.

```
db.movie.find({rating:{$mod:[2,0]}})
```

- To find all movies with odd ratings.

```
db.movie.find({rating:{$mod:[2,1]}})
```

-Find all movies in which Amitabh has acted.

```
db.movie.find({actors:'Amithabh'})
```

If we want to see at particular index i.e. ('actor.1') then it has to be in quotes.

- Find all movies in which Amitabh has acted and it is at 1st index position.

db.movie.find({"actor.1": 'Amitabh'})

- To find using regular expression.

db.movie.find({~~name~~ actor: /^[Aa]mitabh/})

The value in the REGEXP has to come in / /.

- To find all movies with name ends with t.

db.movie.find({~~actor~~ name: /t$/}).

- To find all movies whose name do not start with digits.

db.movie.find({name:/^[^0-9]/})

-To find all movies which has a in it.

db.movie.find({name:/[Aa]/})

- To find all which starts with p ends with t and a somewhere in between.

db.movie.find({name:/^[Pp].*a.*[Tt]$/})

- To find all movies with size of actor array=3 (movies with 3 actors)

db.movie.find({actor:{$size:3}})

- movie name ~~with~~ having string type

db.movie.find({name:{$type="string"}})

- Movie name having number type

db.movie.find({name:{$type="number"}})

db.student.find({'grade.std':10})

- To find all students with grade=82 & std=12 in same obj.

db.student.find({'grade':{$elemMatch:{grade:82, std:12}}}).pretty()

- To insert multiple documents.

db.emp.insertMany([{},{},{}])

- To rename collection

db.emp.renameCollection("employee")

- To drop collection

db.employee.drop()

- To delete the records.

remove
deleteMany
deleteOne

(1) db.movie.remove({}) -- (It will remove all docs).

(2) db.movie.deleteMany({})

db.movie.remove({name:{$type:'number'}})
-To delete emps with name revati.
db.emp.remove({name:'Revati'})

- To delete the records.

remove
deleteMany
deleteOne

(1) db.movie.remove({}) -- (It will remove all docs)

(2) db.movie.deleteMany({})

db.movie.remove({name:{$type:'number'}})
-To delete emps with name revati.
db.emp.remove({name:'Revati'})

Day 17 17-7-21

* Update and Index

update - update one or many documents
updateOne - update only first matching document
updateMany - updat all matching element.

- Update function

1. if you want to add new key value pair.
2. delete existing key value pair
3. rename key name existing
4. Overwrite value of a key
5. increment or decrement value key
6. multiplication value of existing key
7. min and max replace value

8. replace entire document
9. adding or deleting element from array.

```
db.collection.update(
    <query>,
    <update>
    {
      update: <boolean>
      multi: <boolean>
    }                          */
```

db.collection.update(<query>, <update>, <options>)

db.collection.updateOne(<query>, <update>)

db.collection.updateMany(<query>, <update>)

$set, $inc, $mul, $min, $max, $currentDate, $unset

- Increase the price by 100 for all movies with name starts with kahani.

```
db.movie.update({name:/^kahani/},
  {$inc:{price:100}}, {multi:true})
```

multi:true — update all
(* If we dont write multi condition then it will update only 1 i.e. first kahani values).

• pretty doesnot comes with update.

- Increase ticketnumber by 20 for all movies with rating >4.

db.movie.update({rating:{$gt:4}},
{$inc:{ticket_no:20}}, {multi:true}).

- Decrease ticketnumber by 20 for all movies with rating >4.

db.movie.update({rating:{$gt:4}},
{$inc:{ticket_no:-20}}, {multi:true})

- Overwrite the rating by 4 & increase price by 30 for all movies with price >300.

db.movie.update({price:{$gt:300}},
{$set:{rating:4}, $inc:{price:30}},
{multi:true})

- Increase the price by 20% for all movies.

db.movie.update({}, {$mul:{price:1.20}},
{multi:true})

- To set the rating to 5 if current rating is >5.

db.movie.update({}, {$min:{rating:5},
{multi:true})

- delete a key value pair --- $unset

- To remove key ticket_no for all doc.

```
db.movie.update({},{$unset:{ticket_no:''}},
{multi:true})
```

- To increase the price of padmavat movie by 20 and also want to store data on which modification has happened.

(1)
```
db.movie.update({name:'Padmavat'},
{$inc:{price:20},$currentDate:{'modified':true}},
{multi:true})
```

(2)
```
db.movie.update({name:'Padmavat'},
{$inc:{price:20},$currentDate:{'lastchange.
modified':true},$set:{'lastchanged.reason':
'public demand'}},{multi:true})
```

- To decrease ticket_no by 10 and increase price by 10% for all movies with rating>4.

```
db.movie.update({rating:{$gt:4}},
{$mul:{price:1.10},$inc:{ticket_no:-10},
{multi:true})
```

- To overwrite rating by 4 for all movies with price is either 350 or 260 or 500 or name has n anywhere.

db.movie.update({$or:[{price:{$in[350, 260, 500]}}, {name:/n/}]},{$set:{rating:4}, $currentDate:{lastmodified:true}, {multi:true}}

* Arrays

$, $[], $[item], $push, $pull, $pop, $each, $position

- To add only one actor.

db.movie.update({name:'uri'}, {$push: {actor:'mohit raina'}}, {multi:true})

- To add 'kirti kilhari', 'ranjit kapoor', 'akash deep' in movie uri

db.movie.update({name:'uri'}, {$push:{actor:{$each:['kirti kilhari', 'ranjit kapoor', 'akash deep']}}}, {multi:true})

- To add R madhavan and sharman joshi in 3 idiots and particular position.

db.movie.update{name:'3 idiots'}, {$push:{actor:{$each['R madhvan', 'sharman joshi'], $position:2}}}, {multi:true}

-1 deletes from beginning
1 deletes from end (default)

- delete data from array

$pop - deletes the values from the end of the array.

-To delete last actor from movie.

db.movie.update({name: '3 idiots'},
{$pop: {actor: 1}}, {multi: true})

- To delete first actor in movie.

db.movie.update({name: '3 idiots'},
{$pop: {actor: -1}}, {multi: true})

-To delete known value.

To delete particular element

db.movie.update({name: 'chichore'},
{$pull: {actor: 'zzz'}}, {multi: true})

-To delete multiple actors.

db.movie.update({name: 'chichore'},
{$pull: {actor: {$in: ['aaa', 'bbb']}}}, {multi: true})

- To update docs which are matched. (increment mean by 3 where grade is gt 88.)

db.student.update({_id:6}, {$inc:{'grade.$[elem].mean':3}}, {multi:true, arrayFilters:[{'elem.grade':{$gt:88}}]})

- To update all prices of ~~eatable~~ product with name eatable by 10.

db.product.update({name:'eatable'}, {$inc:{'items.$[].price':10}}, {multi:true})

- To update all price of all items with category chocolate

db.product.update({'item.category':'chocolate'}, {$inc:{'items.$.price':10}}, {multi:true})

- $ replace - to rename the key

db.movie.update({}, {$rename:{oldkey:newkey}}, {multi:true})

db.movie.update({name:'kahani2'}, {$rename:{'releasedate':release_dt}})

- Overwrite entire doc of movie Padmavat by new docs.

db.movie.replaceOne({name:'Padmavat'}, {name:'xxx', price:260, ticket_no:456})

# Web Programming Technologies

# WPT

Day 1 19-7-21

* WPT:-

Web Programming Technologies
Traditional approach for learning WPT:
1- HTML Programming
2- CSS
3- Javascript Programming
4- jQuery
5- Bootstrap
6- NodeJs
7- ReactJs

- To develope and build online web applications.

Fullstack Application Development:

JS Full stack:- language: Javascript
- MERN Fullstack (Mongo express react node)
React: MVC framework
- MEAN Full stack (Mongo express angular node)
- MEVN Full stack (Mongo express vue.js node)

Java Fullstack: language: Java dB connectivity: database: Oracle, mysql, sql server (JDBC, Hibernate)
Web: JSP servlet, Spring MVC, Spring Boot(UI)
Runtime: Java Runtime (JVM)
[HTML, CSS, jQuery, bootstrap
React or Angular or Vue
Tomcat ]

[In case of JS Fullstack:
webservers are developed wing nodeJs
HTTPwebserver: using Nodejs, Programming using JS]

## .NET Fullstack:

Language : C#
Database : Oracle, sql server, mysql
Runtime : CLR (.net runtime) ... Dotnetcore/ Dotnet framework
Web : asp.net MVC + IIS
HTML, CSS, jquery, bootstrap ... core classical web technologies
React or angular or vue
DB connectivity: ado.net, entity framework (ORM)
MVC : Model View Controller Design Pattern

## LAMP Fullstack:

(Linux Apache MySQL PHP)

Operating system: Linux
Web server : Apache
Database : MySQL
Web server : PHP Web Pages (Hypertext Preprocessor)
DB connectivity : using PHP fn

Web server :-
A dedicated program running continuously on machine and capable of hosting more than on web applications.

web servers :-
Microsoft : IIS server
TOMCAT
nginx
webLogic
Glassfish, etc.

* WebProgramming:

Website, Web page, Web Application;-
// Web server, Web service (REST API),
// Multi Web

~~Website~~,
WebPage:
Static Webpage / Dynamic web Page
Web Application (Website)
Static web Application / Dynamic web applic.
Multi web Page (MPA)
Single Page Application (SPA)
Web server
Wer services (REST API)

* WPT:- (MERN
FullStack Developer:
Language : Javascript (syntax)
Runtime : Node Js (Appl$^n$ execution using engine)
Server Programming : Express (web Framework)
Dynamic web UI : React, jQuery (Dynamic web Pages dom)
Responsive UI : Bootstrap Framework (responsive UI)
Classical web technologies: HTML, CSS (web contents UI ele)
Database Connectivity : DB connectors (ODBC connectors)
mysql connectors
mongoose connectors

* Fullstack Developer:

Front end developement (UI)
Server side programming (middleware development)
(core server side)
Backend Development (Backend, Database dev)

MERN
MEAN
MEVN
Java
.net
LAMP
Python
Go

* MERN Fullstack;-

Starting point for web Programming learning

Hello World: Program

1: NodeJS (core execution Platform)
2: Javascript Programming (Programming concepts)
3: ~~HTML~~ Database connectivity (DB connectivity)
4: HTML (Presentation)
5: jQuery (Dynamic UI presentation)
6: React JS (JS library for dynamic web app)
7: CSS (visual apperance)
8: Bootstrap (Responsive UI framework)

Getting Started:

1. Write a simple HelloWorld Application
   Tool: Visual code ..... Text editor
   Lang: Javascript
Runtime: NodeJS

a. Create a folder HelloWorldApp
b. Add new hello.js file
c. Write simple source code to display HelloWorld message
d. Run hello.js file using runtime engine NodeJS

HelloWorld

```
console.log("HelloWorld");
```

To open cmd to run code:- CTL shift +c

cmd:- node hello.js
o/p : "HelloWorld"

var is datatype used for all datatype in JS i.e. int, float, boolean, etc.

e.g.
```
var name = "Pratik";
var count = 45;
var status = false;
console.log(name);
console.log(count);
console.log(status)
```

In cmd =
node hello.js
o/p
"Pratik"
45
false

Any file which get executated
- using node is server side.
- HTML are client side.

* Why are we learning Node JS, javascript as part of web programming.

Web Programming :

Writing server side code
browerless code

- The code which will wait for client request
- The code which will perform core task at server
- The code which will perform some background work
- The code which will execute some alagrithm against data collection
- The code which will generate response after processing incoming request.
- The code which will send response to calling client Application

Web Technology :-

| Client side Technologies | Server side Technologies |
| --- | --- |
| - Front end logic UI presentation | - Business logic |
| - HTML, CSS, Bootstrap | - Node JS, Express JS |
| - Javascript : Validation, external data access | - Javascript : Data Access logic, request processing logic, response generation logic, session state mgmt logic, authentication, authorization logic server listening logic, etc |

- Server side code is dependant on event driven mechanism.

```
flowers = [
{ id=23, title = "Gerbera", description= "Wedding flower"},
{id=24, title = "Rose", description= "Valentine flower"},
{id=25, title = 'Jasmine", description= "Smelling flower"},
{id=26, title= "Lotus", description = "Worship flower"}.
];

console.log(3)
o/p=  id=26
      title = Lotus
      description = "Worship flower".
```

* Functions

```
function show() {
    var count = 67;
    count ++;
    console.log(count);
};
function display(company) {
    console.log ("company name = " + company);
};

var provider= "Transflower Learning Pvt. Ltd."
display(provider);
show();
```

[o/p]= company name = "Transflower Learning Pvt. Ltd.
68.

```
function processJob1() {
 console.log("Processing by 1"); }

function processJob2() {
 console.log("Processing by 2"); }

function processJob3() {
 console.log("Processing by 3"); }

setInterval(processJob1, 5000);
```

- In this processJob1 is get automatically called by setInterval function.
- SetInterval function is used to repeatedly calling the function itself with some (given) time interval.
- processJob1 is function is called automatically with 5000 millisecond time.

```
setInterval(processJob2, 1000);
setInterval(processJob3, 10000);
```

o/p :-

~~Processing by 2~~ Processing by 2