If you want to store data in persistent storage, so that it will remain available till you delete it, then there 2 options

1. Store the data in file.
2. Store the data in database

Drawbacks of files

1. It is sequential access storage. Hence performing CRUD operation is tedious on files. Hence we use databases for storage of data

Types of databases

1. SQL
    a. If you need structured data
    b. It stores the data in tabular format
    c. Its secure
    d. Usually used in financial applications
    e. Examples: MYSQL,Oracle, Postgresql, SQL server
2. NOSQL
    a. It is unstructured database.
    b. Stores the data in json(javascript object notation) format
    c. But less secure than SQL
    d. Usually used in media type applications
    e. Examples – MongoDB, Cassandra, CouchbaseDB
3. GrapDB
    a. If you want to store data in graphs format(like networks)
    b. Example Neo4J
4. Memory database
    a. If you need very fast access to database, and if size of data is very small, then we may use memory database
    b. Example-MemDB,VoltDB

Where we use database

1. Single user application
2. Web Application---- this application return data in HTML format, o/p includes view
3. Web Services / Microservices----- the application which returns only data, usually in json format.
4. Mobile Applications

To create database in mysql

Create database mydb

Types of keys in table

1. Primary key
    a. Minimal set of columns which identifies the row uniquely is called as primary key
    b. It cannot contain any null value

sid	Sname	M1	M2	M3	subid

oid	itemid	name
1	1	tshirt
1	2	books
2	1	tshirt

Booking

Roomno	custno	name	bkdate	rate
1	1		1 Apr23	
1	1		10 apr	
2	1		1 Apr 23	

2. Foreign key→ referential integrity
   If for checking correctness of data in a column, we are referring the column in same table or in another table then it is called as foreign key. And the column which we refer has to be primary key of the table.
   Primary key deptid

Deptid	dname	location
12	HR	Mumbai
13	Purchase	Pune
14	sales	Pune

Primarykey –empid

Foreign key--- deptid references dept(deptid)

Foreign Key----Manager no refences emp(empid)

Empid	Ename	Sal	deptid	address	Manager no
1	xxx		12		3
2	yyy		13		
3	zzz		14		
4	Ddd				

3. Candidate key --- any minimal set of columns which identifies the row uniquely is called as candidate key
   For a table there can be more candidate keys for a table but only one primary key will be there
   (stdid, mobile, passport, adhar no)

Stud id	Sname	mobile	passportno	Adahar num	address				
1			1234567						
2									

3			2345678						

4. Super key→ any combination of columns which identifies the row uniquely is called as super key.
   (studid, stuid+sname, studid+sname+mobile
5. Unique key-→any column which has unique values, but it is not primary key.
   Unique key may contain many null values, but all not null values should be unique.

RDBMS- Relational Database management system

Data models

1. Hierarchical Model- if your data is stored in parent and child node format then it is called as hierarchical model.
2. Network model-→ if any node is connected to any other node then it is called as network model
3. Relational model--→ if the data is stored in the form of tables, then it is called as relational model

synonyms

Table-→ relation

Fields-> attributes,columns

Rows-→ record,tuple

Why to use database

1. Databases are secure.
2. Networking is possible.
3. Sharing of data is very easy.
4. It stores the data in correct form.
5. Redundancy is reduced.

Acno	Custid	Type	balance
1	100	Saving	2345
2	100	Demat	3456
3	100	Current	5555
4	200	Saving	7777

customer

Custid	Cname	Mobile
100	Kishori	22222
200	Rajan	5555

Install MySQL

https://dev.mysql.com/downloads/installer/

to start mysql

1. On windows start button type mysql > mysql command line client >enter root password
   Or
   Open terminal/cmd prompt on the machine and use following command

   mysql -u root -p

   mysql> create database DACmarch23
   -> ;
   Query OK, 1 row affected (0.04 sec)

   mysql> use DACmarch23
   Database changed
   mysql>source d:\mysql_database\demobldmysql.sql

   to drop the tables
   mysql> drop table dept;

   to find list of table names
   show tables;

   to see the columns in the tables(table structure)
   mysql>desc emp;
   mysql>desc dept;

   to see the data in the table
   mysql> select * from emp;

Types of statement	Full form	statements
DDL	Data definition language	Create table, alter table, drop, truncate
DML	Data Manipulation language	Insert, update, delete
DQL	Data query language	select
DCL	Data control language	grant , revoke
TCL	Transaction Control statement	Commit, rollback, savepoint

Arithmetic operators

+, -, *, /, %

Relational operators

>=, <=,> ,<,=, !=

Logical operators

and , or, not

*****assume that 20 records are there with 3 records have name Rajan in it and 7 records has salary >4000.

In and operator first if first condition is true then only 2nd condition gets checked

In or operator if first condition is false then only second condition gets checked

select *

from emp

where ename='Rajan' and sal>4000--------fast will check 23 conditions(20+3)

where sal>4000 and ename='Rajan'--------slower will check 27conditions(20+7)

Other operators

[not] in	It is used to check equality with multiple values with or condition	To check sal is either =2000 or 4000 Sal in (2000,4000) To check sal is not either =2000 or 4000 Sal not in (2000,4000)

[not] Between….and	To check the range of values we use between and operator. And the given values are inclusive	To check Sal>=2000 and sal <=4000 Sal between 2000 and 4000 To check Sal<2000 and sal >4000 Sal not between 2000 and 4000
[not] Like	To check the pattern then we use like operator In this operator % matches with 0 or more characters _ (underscore) → matches with 1 character	To find all names starts with A Name='A%' To find all names ends with A Name='%A' To find all names starts with A, k at 3rd position Name='A_K%' To find all names starts with A, and ends with k Name='A %K'

REGEXP---> instead of Like we can use REGEXP

^	To find the pattern at the beginning of the string
$	To find the pattern at the end of the string
+	It matches with 1 or more occurrences of preceding pattern
*	It matches with 0 or more occurrences of preceding pattern
?	It matches with 0 or 1 occurrences of preceding pattern
{m}	It matches with exactly m occurrences of preceding pattern
{m,n}	It matches with minimum m or maximum n occurrences of preceding pattern
{m,}	It matches with minimum m or maximum any occurrences of preceding pattern
[A-Za-z]	It matches with any alphabet
[0-9]	It matches with any digit
[ABC]	It means either A or B or C
.	Matches with any one character it can be alphabet or digit or special character
(abc|pqr|xyz)	Matches with either abc or pqr or xyz

^A.*N$	AN, AxN, AXXN, Asdfghjkjkljklk;kl;lN
^[AM]	Axcvdfs, Mskjdhf jkshdjfhhsd
^A..S	AweSdjkfhgjdh,AdkS
^Ax{5}N	AxxxxxN
^A.*N$\|^M\|^.*N.*R$	Ajdfhjk dfhjN, Mshdgfhgs, kjsdhfjkhsNkjefhjskhdR

Derived columns

In calculation if any column contains null value then use ifnull function to replace null value with some number.

To display empno, name, sal, commission and net salary = sal + comm

select empno,ename,sal,comm,ifnull(comm,0),sal+ifnull(comm,0)

-> from emp

Using alias name

If alias name contains space then eclosing it in double quotes is mandatory

Otherwise it is optional.

```
mysql> select empno ”Emp number” ,ename,sal,comm,ifnull(comm,0),sal+ifnull(comm,0) "Net sal"
    -> from emp;
```

Using distinct keyword

To display different values in any column (to display unique values)

```
Select distinct job
From emp;
```

Built-in functions available in mysql

The built-in functions are of 2 categories

1. Single row functions --- can be used in select clause as well as where clause
2. Aggregate functions or multirow functions

Number functions

Round(val, n)	Helps to round the value up to n decimal places	select empno,ename,sal,round(sal+sal/3,2) Netsal -> from emp;
truncate(val, n)	Helps to truncate the value up to n decimal places	select empno,ename,sal,truncate(sal+sal/3,2) Netsal -> from emp;
Ceil(val)	It gives integer o/p. always remove fraction portion and displays the next value.	select empno,ename,sal,ceil(sal+sal/3) Netsal from emp;
Floor(val)	It gives integer o/p. always remove fraction portion and displays the previous largest number.	select empno,ename,sal,floor(sal+sal/3) Netsal from emp;
Abs(val)	It display the value always in +ve	Select abs(-3)
Mod(val,n)	Displays remainder of val%n	Select mod(11,2) ans :1

Character functions

Upper()	To convert all the characters in uppercase	
Lower()	To convert all the characters in lowercase	

Substr(value,start,length)	It will display length number of characters from start position. Calculation starts from 1	
Concat(x,y,z,.....) In oracle concat function accepts only 2 parametrs Concat (Concat(x,y),z)	Concatenate all the strings	
Left(val,n)	It will retrieve n characters from left side of the string	
Right(val,n)	It will retrieve n characters from right side of the string	
Rpad(val,n,ch)	To add character ch on right side of val, so that the max length of the val=n	
lpad(val,n,ch)	To add character ch on left side of val, so that the max length of the val=n	
Instr(val,ch)	It gives the position of first occurrence of ch in the given val	
Trim(val)	To remove extra spaces from left or right side of the value then use trim function	
rtrim(val)	To remove extra spaces from right side of the value then use trim function	
Ltrim(val)	To remove extra spaces from left side of the value then use trim function	
Format(number,d)	This will display numbers with thousand separator, d number of digits after decimal point	

To generate email for all employees and email is first 3 characters of ename followed by . followed by first 3 characters of job followed by @mycompany.com

Substr(ename,1,3)

Substr(job,1,3)

Concat(Substr(ename,1,3),”.”, Substr(job,1,3),”@mycompany.com”)

```
select empno,ename,job,Concat(Substr(ename,1,3),".", Substr(job,1,3),"@mycompany.com") email
   -> from emp;
```

Date related functions

To sort the data in mysql or in oracle use order by clause

- In order by clause by default the sorting will be done in ascending order
- To sort it in descending order you have to explicitly specify desc keyword
- Order by clause is added after where clause or from clause.
- Ordering is possible on derived columns.
- In order by, if the order is ascending then
    - Null values will at the top
    - Then numeric values
    - String values

11. List the details of the employee , whose names start with 'A' and end with 'S' or whose names contains N as the second or third character, and ending with either 'N' or 'S'

Select *

From emp

Where ename like 'A%S' or ename like '_N%N' or ename like '__N%N' or ename like '_N%S' or ename like '__N%S'

Or

Select *

From emp

Where ename REGEXP '^A.*S$|^..?N.*[NS]$'

..?N ----→AN, ANN

Is null | is not null

To find all employees who earned commission

select * from emp

-> where comm is not null and comm >0;

List the empno, name, and department number of the emp who have experience of 18 or more years and sort them based on their experience.

Select empno,ename,deptno,hiredate,floor(datediff(curdate(),hiredate)/365) experience

From emp

Where floor(datediff(curdate(),hiredate)/365) >=41

Order by experience;

To find month and year in mysql/oracle

select extract(month from curdate()),extract(year from curdate());

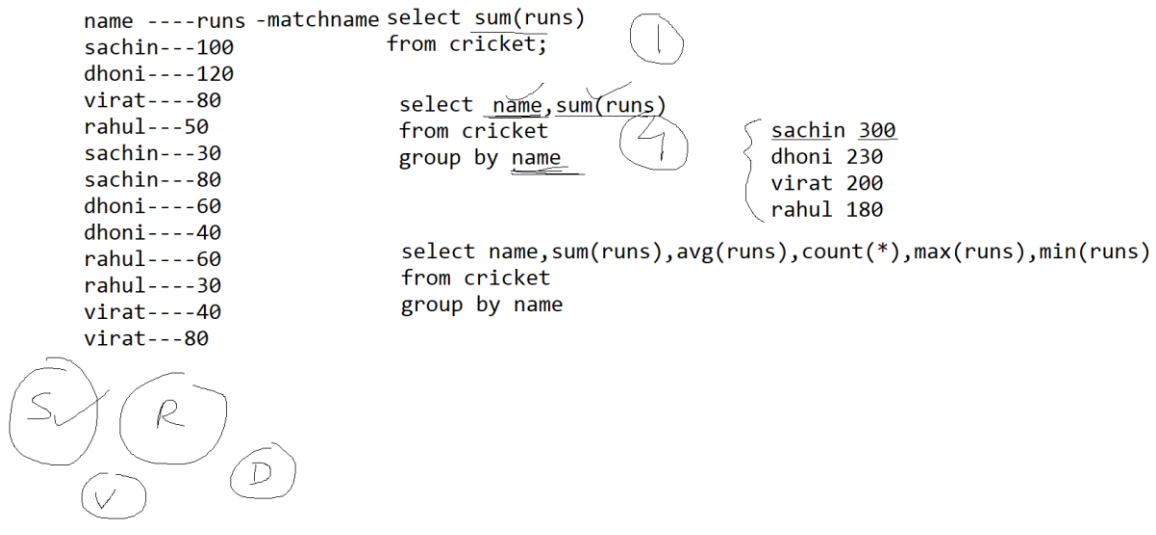
Date function

Datediff()	Find difference between 2 dates in terms of days
Date_add()	To find date after some interval
Date_sub()	To find the date before some interval
Date_format()	To display date in user understandable format
Now()	To get current date and time
Curdate()	To get only date
Day() Extract(day from curdate())	Will retrieve only date
month() Extract(month from curdate())	Will retrieve only month
year() Extract(year from curdate())	Will retrieve only year
Dayname()	Will display name of day (ex: Sunday, Monday etc)
Monthname()	Will display month name (ex: 'January, February , …etc
Week or weekofyear	To find week number
Str_to_date	Will convert user format into sql format

Aggregate function ----- avg, sum, count, min, max

We use group by clause and having clause

- In group by clause count(*) will count number of rows
  And count(column name) ex. Count(comm) will count not null values
  Null values will be ignored.



1. To find sum of sal, sum of netsal, count all emp, count number who earned comm, for each department

```
select deptno,sum(sal),sum(sal+ifnull(comm,0)),count(*),count(comm)
   -> from emp
   -> group by deptno;
```

2. To find min, max sal in our company

```
Select min(sal) minsal ,max(sal) maxsal
From emp;
```

3. Count number of employees, min sal,max sal, sum of sal, for every job

```
Select job,sum(sal),min(sal),max(sal),count(*)
From emp
Group by job;
```

4. To find sum, max min of sal department wise and job wise

```
select deptno,job,sum(sal),count(*),min(sal),max(sal)
   -> from emp
   -> group by deptno,job
```

5. To find sum, max min of sal department wise and job wise, only if the sum(sal)>2000

```
select deptno,job,sum(sal),count(*),min(sal),max(sal)
from emp
group by deptno,job
Having sum(sal)>2000;.
```

6. To find sum of sal of all employees departmentwise for all clerks

```
Select deptno,sum(sal)
From emp
Where job='CLERK'
Group by deptno
```

7. To find sum of sal of all employees departmentwise for all analyst only if the department has 2 or more analyst

```
Select deptno,sum(sal)
From emp
Where job='ANALYST'
Group by deptno
Having count(*)>=2;
```

1. To find sum of sal of all employees departmentwise for all CLERKs only if the department has 2 or more CLERK

Select deptno,sum(sal)

From emp

Where job='ANALYST'

Group by deptno

Having count(*)>=2;

Select deptno,job,count(*)

From emp

Where sal>2000

Group by deptno;

a. Deptwise, jobwise count
b. Only deptwise count
c. Only jobwise count
d. Error

Select count(*)

From emp

Where sal>2000

Group by deptno;

a. Deptwise, jobwise count
b. Only deptwise count
c. Only jobwise count
d. Error

What is the sequence in which statements will get executed

```
Select count(*)   1
From emp          2
Where sal>2000    3
Group by deptno   4
having count(*)>2   5
order by count(*)     6
```

a. 1, 2,3, 4,5,6

b. 2,3,4,5,6,1
c. 3,1,2,4,5,6
d. None of the above

Which of the following statement(s) are true

a. Derived column can be used in order by clause
b. Derived column can not be used in order by clause
c. Derived column alias name can be used in order by clause
d. Both A and C
e. None of the above

Which of the following statement(s) are false

a. Conditions with aggregate functions can be used with having clause
b. Conditions with aggregate functions can be used with where clause
c. Conditions with column names in tables can be used in where clause
d. None of the above

DML(insert, update , delete)

1. To add record in a table

Insert into emp values(101,'Ashutosh','CLERK',7902,'2000-10-11',3456,345,10)

2. To add a record in the table with few values, then add list of columns after table name

Insert into emp(ename,empno,sal) values('Tanaya',103,5555);

3. Insert many records in the table

Insert into emp(empno,ename,sal,comm,job)

-> Values (12,'Rajani',3456,456,'Analyst'),

-> (13,'Meenal',4444,567,'Manager'),

-> (14,'Monica',3333,333,'Astmgr');

To delete the record

1. To delete all the rows

Delete from emp; --→ it is available in mysql and oracle

Delete emp-→ available in oracle

2. To delete all rows who are working as CLERK
   Delete from emp where job='CLERK';

Drop table	delete
Drop will delete data and table both	Delete will delete only data from table

To update data from all the rows
To update salary of all employees by 10%,comm=2% of sal
Update emp
Set sal=sal*1.1,comm=0.02*sal,deptid=20

To update salary by 2000 of SMITH
Update emp
Set sal=sal+2000
Where ename='SMITH'

Nesting of query

1. To find all employees who are working in dept 20
   Select * from emp
   Where deptno=20

2. To find all employees who are working in smith's dept
   Select deptno
   From emp
   Where ename='SMITH'

   Select * from emp
   Where deptno=20

   Select * from emp
   Where deptno=(Select deptno
   From emp
   Where ename='SMITH'
)
3. To find all employees whose sal > jones sal
   Select * from emp
   Whare sal>(select sal from emp where ename='JONES')

4. To find all employees who are working in either smith's dept or JONES dept

   Select * from emp

   Where deptno in (select deptno

   From emp

   Where ename in ('SMITH','JONES'))

5. To find all employees with sal > either jones salary or Miller's salary
   Select * from emp
   Whare sal > any (select sal from emp where ename in ('JONES','Miller))

6. Find all employees with sal > average salary of dept 10
   Select *
   From emp
   Where sal>(
   Select avg(sal)
   From emp
   Where deptno=10)

7. Find all employees with sal > avg(sal) of dept 10 ,
   and ename starts with either “K” or “A”

Select *

From emp

Where sal>(Select avg(sal) From emp Where deptno=10) and ename like ‘J%’

Or
Select *
-> From emp
-> Where ename REGEXP '^[KA]' and sal >(select avg(sal) from emp where deptno=10)

8. To find all employee with sal > smith’s sal and sal< jones salary
   Select sal from emp where ename=’SMITH’ 800
   Select sal from emp where ename=’JONES’ 2500

   Select *
   From emp
   Where sal between (Select sal from emp where ename=’SMITH’) and (Select sal from emp where ename=’JONES’
)

Corelated query

- If the nested query is dependent on parent query for data, then it is called as co-related query
- In co-related query inner query gets executed n times, if in parent table n rows are there.
- The inner query will get executed once for each row in the parent table
- Use nested query when you want to show data in the o/p only from one table.

In co-related query we use exists and not exists operator

Exists will return true if rows are found and false if rows are not found.

Not exists will return true if rows are not found and will return false if rows are found.

pid	pname	qty	price	cid
2	Nachos123	302	150.00	1
3	Pringles123	29	150.00	1
4	Marie gold	20	50.00	2
5	nice1234	35	45.00	2
6	good day123	45	60.00	2
20	Hide and seek	45	40.00	2

cid	cname	cdesc
1	chips	very crispy
2	biscuits	sweet and tasty
3	chocolate	yummmmmy
4	cold drinks	thanda thanda cool cool

1. Find first topmost 2 product based on qty
   Select *
   From product
   Order by qty desc
   Limit 2
2. Find minimum price for each category.
   Select cid, min(price)
   From product
   Group by cid

3. Find all products of either chips or cold drink category.
   Select * from product
   Where cid in (select cid from category where cname in('chips','cold drinks'))
4. Find all products with price >25 and <100
   Select * from product where price between 26 and 99
5. Find all products for category 1
   Select * from product where cid=1;

   DDL statement -----

   create table to create a table
   write a query to create room table to store rid, rname and loc
   create table room(
   rid int primary key,
   rname varchar(30),

```
  loc varchar(30)
)
```

write a query to create faculty table to store fid, fname and skills

```
create table faculty(
   fid int primary key,
   fname varchar(20),
  skill varchar(30)
)
```

Write a query to create table product to store pid, pname, qty, price, cid

```
Create table product(
Pid int primary key,
Pname varchar(20),
Qty int,
Price float(7,2)
cid int);
```

Write a query to create table category to store cid, cname, cdesc

```
Create table category(
   cid in primary key,
cname varchar(20),
cdesc text
)
```

Create table

1. In create table, we can assign 2 types of constraints
    a. Field level constraints → these constraints can be written immediately after field definition.
        i. Not null
        ii. Default
        iii. Unique
        iv. Auto_increament
    b. Table level constraints -→ these constraints can be written immediately after field definition or after last field definition.
        i. Foreign key
        ii. Primary key
        iii. Check constraint.

Not null	It does not allow null values in the column
default	It specifies the value to be added in the field, if user has given null value
unique	It does not allow duplicate values in the column
Auto_increament	It generates the value automatically
Primary key	• It does not allow null values in the column and It does not allow duplicate values in the column, • In the table there will be only one primary key • If primary contains one column, then it is called as simple primary key. • If primary key contains many columns, then it is called as composite key. • Composite key has to be defined after last field definition
Check constraint	It checks some condition on the value entered by user, if condition satisfies then only the value will be stored in the column
Foreign key	It references primary key column of same or other table, if value exists in that column, then only it allows to store the value in the current column Primary key can also be a foreign key One table can have more than one foreign key

1. To create a table product, store following fields in the table

pid	int	Primary key	Create table product(pid int primary key, pname varchar(20) not null unique, qty int check(qty >0) default 10, price double(9,2) check(price>0), catid int, foreign key fk_cid(catid) references category(cid)
pname	Varchar(20)	Not null , unique	
qty	int	Check (qty > 0) default-10	
price	Double(9,2)	Check(price >0)	

catid	int	Foreign key reference category(cid)	on delete set null on update cascade);

2. To create a table Category, store following fields in the table

cid	int	Primary key	Category table : Create table category(cid int primary key , cname varchar(20) not null, cdesc varchar(20));
cname	Varchar(20)	Not null	
Cdesc	Varchar(20)		

3. Create a table studene_marks

sid	int	create table stud_marks(sid int, course varchar(20), marks int, primary key(sid,course), foreign key fk_sid(sid) references student(sid), foreign key fk_course(course) references coursedata(cname))
course	Varchar(20)	
marks	int	
Primary key	Sid+course	

Sid course marks
1 java 98
1 c++ 96
1 database 100
2 java 95

1. On delete cascade / set null
   on update cascade

2. Auto_increament-→
   - in one table there can be only one auto_increment column
   - in the auto increament column values can be added explicitly
   - the start value is by default 1, but it can be changed by using alter table

ALTER TABLE student AUTO_INCREMENT=1001

Create table student(
Sid int primary key auto_increment,
Sname varchar(20),

Address varchar(20));

Create table coursedata(
cname varchar(20) primary key,
duration int)

- to insert data in auto increment columns
  insert into student(sname,address) values('Rajan','Baner')
  insert into student values(default,'Revati','Baner')
  insert into student values(20,'Revati','Baner')

ALTER TABLE student AUTO_INCREMENT=1001

Alter table statement

Add new column	ALTER TABLE table_name ADD new_column_name column_definition [FIRST | AFTER column_name], ADD new_column_name column_definition [FIRST | AFTER column_name], ...
Delete a column	ALTER TABLE table_name Drop column column_name
Modify the column type	ALTER TABLE table_name MODIFY column_name column_definition [FIRST | AFTER column_name];
Change column name	ALTER TABLE table_name

	change column old-column_name new-col-name data type
Add new constraint	ALTER TABLE table_name Add new-constraint
Drop a constraint	ALTER TABLE table_name drop constraint
Rename the table	ALTER TABLE table_name Rename to new-table-name

1. To add roomid column in coursedata table

ALTER TABLE table_name

Add roomid int after cname;

2. Add faculty id column in coursedata
   Alter table coursedata
   Add facid int ,
   Add cdesc varchar(20);
3. To drop the column
   Alter table coursedata
   Drop column cdesc
4. To modify data type of facid from int to varchar(20)
   Alter table coursedata

   Modify facid varchar(20)

5. Change column name
   Alter table coursedata
   Change column facid fid int;

   Create table mytable(
   id int primary key,
   name varchar(20) unique,

price int)

6. To drop primary key constraint
   Alter table mytable
   drop primary key

7. To drop foreign key constraint
   ALTER TABLE `table_name`
   DROP FOREIGN KEY `id_name_fk`;
8. Add new primary key constraint
   Alter table mytable
   Add primary key(id)
9. Add new foreign key constraint
   Alter table mytable
   Add constraint f12 foreign key(cid) references category(cid)
10.

create table mytable1(
-> id int primary key,
-> name varchar(20),
-> cid int,
-> constraint f11 foreign key(cid) references category(cid),
Constraint un unique(name));

To drop the constraint
Alter table mytable1
Drop foreign key f11

11. To change table name mytable1 to mytable_dac
    alter table mytable1
    rename to mytable_dac

------to find constraint name

1. To see create table query to find constraint names
   Show create table <tablename>
2. To see find constraint names
   Select table_name,constraint_type,constraint_name
   From information_schema.table_constraints
   Where table_name=<name>

1. To find all employees who are not manager of any other employee.
   Select *
   From emp e
   Where not exists(select * from emp m where e.empno=m.mgr)

Joins in the table

If you want to retrieve the information from more than one table, then use joins

There are 3 type joins

1. Cross join –combining data from multiple tables
2. Inner join (natural join)--- combining data from more than one table with join condition is called as inner join
   a. Equi join -----if join condition is based on = sign then it is called as equi join
   b. Non equi join
   c. Self join
3. Outer join
   a. Left outer join
   b. Right outer join
   c. Full outer join

Create table mytable3

(id int primary key

Name varchar(20));

To add not null constraint

- Adding not null constraint will be allowed if the table is empty, otherwise, initially add the column, then update values and then modify column to add not null constraint.

Create table mytable3

Add mobile int not null

To add unique constraint

Create table mytable3

Add column address(20)

If the table constains data then to add unique constraints, existing values should satisfy the unique constraint

Alter table mytable3

Add constraint un unique(address)

Types of joins

1. Cross join
2. Inner join
    a. Equijoin
        i. If join condition is based on = sign then it is called as equijoin
    b. Non equi join
        i. If the join condition is based on not equal to operator
    c. Self join
3. Outer join--- if we want to display matching as well as non matching rows then use outer join
    a. Left outer --→ to get matching and nonmatching rows from left side table, then use left outer join
    b. Right outer-→ to get matching and nonmatching rows from right side table, then use right outer join

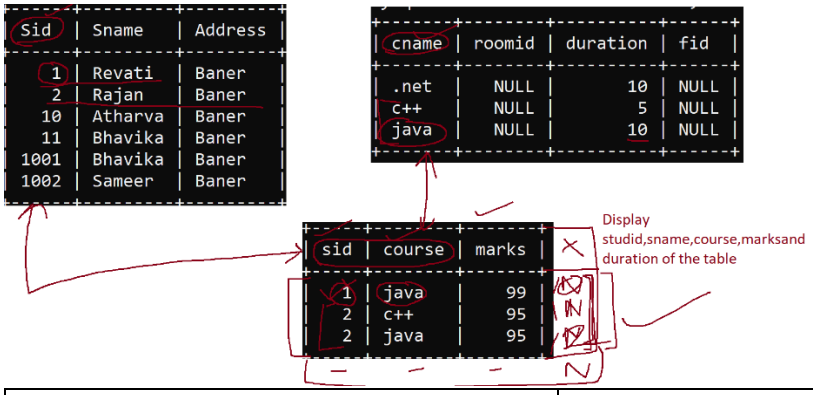
    c. Full outer-→ to get matching and nonmatching rows from both side table, then use full outer join

Emp(empid,ename,sal,job,deptno)

Dept(deptno,dname,loc)

1. To find sid,name,marks and duration of all the courses
   select *

from student s, stud_marks sm,coursedata c
where s.sid=sm.sid and sm.course=c.cname;



2. To find the marks and duration of all courses for which Rajan appeared.
   select *
   -> from student s, stud_marks sm,coursedata c
   -> where s.sid=sm.sid and sm.course=c.cname and sname='Rajan';
   3. To display empno,name,sal,deptno,grade and dname for all employees
   Select empno,ename,sal,e. deptno,grade,dname
   From emp e,dept d, salgrade s
   Where e.deptno=d.deptno and e.sal between s.losal and s.hisal

3. To display empno,name,sal,deptno,grade and dname for all employees who are working in either sales department or purchase dept

Select empno,ename,sal,e. deptno,grade,dname

From emp e,dept d, salgrade s

Where e.deptno=d.deptno and e.sal between s.losal and s.hisal and dname in ('sales','purchase')

4. To display empno,name,sal,deptno,grade and dname for all employees who are working in either sales department or Accounting dept and sal >2000

Select empno,ename,sal,e. deptno,grade,dname

From emp e,dept d, salgrade s

Where e.deptno=d.deptno and e.sal between s.losal and s.hisal and dname in ('sales','accounting') and e.sal>2000

5. To find empno,ename,sal,mgrno mnager name,manager sal

Select e.empno,e.ename,e.sal,e.mgr "managerno",m.ename "manager name",m.sal "manager sal" From emp e, emp m Where e.mgr=m.empno	Select e.empno,e.ename,e.sal,e.mgr "managerno",m.ename "manager name",m.sal "manager sal" From emp e inner join emp m on e.mgr=m.empno

6. Display empno,ename,sal,dname for all employees

Select empno,ename,sal,dname From emp e,dept d Where e.deptno=d.deptno	Select empno,ename,sal,dname From emp e inner join dept d on e.deptno=d.deptno

1. Display empno,ename,sal,dname for all employees , and also display departments in which no employees are there
   Select empno,ename,sal,dname
   From emp e right join dept d on e.deptno=d.deptno;

2. Display empno,ename,sal,dname for all employees , and also display employees for whom no department is assigned.
   Select empno,ename,sal,dname
   From emp e left join dept d on e.deptno=d.deptno;

3. Display empno,ename,sal,dname for all employees , and also display employees for whom no department is assigned. And also display dept in which no employees are there
   Rule for union --→ to use union both queries should have same number of columns
   Corresponding columns data type should match

Select empno,ename,sal,dname

From emp e left join dept d on e.deptno=d.deptno

union

Select empno,ename,sal,dname

From emp e right join dept d on e.deptno=d.deptno;

To take full join of 3 tables in mysql

1. In mysql full join operator does not exists
2. So to take full join of 2 tables
   a. Take left join of 2 tables
   b. Take right join of 2 table
   c. Do union of 2 queries
3. To take full outer join of 3 tables
   a. Take left join of all 3 tables ------step a
   b. Take right join of 2 table add filter condition
      where <some column from leftside table> is null
   c. Add dummy columns to match number of columns of the query executed in step -a
   d. And take union of these 2 queries

Faculty table

Fid	Fname	skills
1	X	Java
2	y	C++
3	Z	.net

Room table

Roomid	Rname	location
10	Mogra	First floor
20	Lotus	First floor
30	Jasmin	Second floor

Course

Courseid	Cname	Std_count	rid	fid
1	DAC	240	10	1
2	DBDA	60		
3	DTISS	50	20	

o/p of query

From faculty f left join course c on c.fid=f.fid left join room r on r.roomid=c.rid

Courseid	Cname	Std_count	rid	fid	fid	fname	skill	Rid	Rname	location
1	DAC	240	10	1	1	X	Java	10	Mogra	First floor
					2	y	C++			
					3	Z	.net			

o/p

Courseid	Cname	Std_count	rid	fid	**null**	null	null	roomid	rname	location
								20	Lotus	First floor
								30	Jasmin	Second floor

From faculty f left join course c on c.fid=f.fid left join room r on r.roomid=c.rid

From course c right join room r
Where c.cid is null

1. Display cname,fname,rname which are assigned to courses also display faculties who are not assigned to courses also display rooms which are not assigned to the courses
   Select cname,fname,rname
   From faculty f left join course c on c.fid=f.fid left join room r on c.rid=r.roomid
   union
   Select cname,null,rname

From course c right join room r on c.rid=r.roomid
Where c.cname is null;

2. Display cname,fname which are assigned to courses also display faculties who are not assigned to courses also display courses for which no faculties are assigned

   Select c.cname,f.fname

   From course c left join faculty f on c.fid=f.fid

   Union

   Select c.cname,f.fname

   From course c right join faculty f on c.fid=f.fid

3. Display all courses and rooms which are assigned to courses and also display rooms which are available.
   Select c.cname,r.rname,r.roomid
   From course c right join room r on c.rid=r.roomid

4. Display all cname and faculties allocated tot the course, also display faculties who are not assigned to any course

Select cname,f.fid,f.fname From course c right join faculty f on c.fid=f.fid

5. Display course name and faculty name allocated to the course

Select c.cname,f.fname From course c inner join faculty f on c.fid=f.fid	Select c.cname,f.fname From course c , faculty f where c.fid=f.fid

6. Display all coursename and rname allocated to course

Select c.cname,r.rname From course c inner join room r on c.rid=r.roomid	Select c.cname,r.rname From course c , room r Where c.rid=r.roomid

7. Display courseid, course name, rname,faculty name

Select c.courseid,c.cname,r.rname,f.fname From course c inner join room r on c.rid=r.roomid inner join faculty f on c.fid=f.fid	Select c.courseid,c.cname,r.rname,f.fname From course c , room r, faculty f c.fid=f.fid Wheren c.rid=r.roomid and c.fid=f.fid

Create table

account(acid int,custid int, type varchar(20), balance double(9,2))

primary key ----acid

foreign key ---custid references customer(custid)

customer(custid int, cname varchar(20),address varchar(20),mgrid int)

primary key----custid

foreign key----mgrid references manager(mgrid)

Managers(mgrid int,mname varchar(20),mobile char(15))

primary key----mgrid

create table manager(

mgrid int primary key,

mname varchar(20) not null,

mobile char(15)

)

Create table customer(

Custid int primary key,

Cname varchar(20) not null,

Address varchar(20),

Mgrid int,

Constraint fk_mgr foreign key (mgrid) references manager(mgrid)

On delete set null

On update cascade

)

Create table account(

Acid int primary key,

Custid int,

Type varchar(20),

Balance double(9,2),

Constraint fk_custid foreign key (custid) references customer (custid)

On delete set null

On update cascade

);

----------insert record

insert into manager values(100,'tanaya',0100020);

insert into manager values(101,'ram',0102020);

insert into manager values(103,'raj',0103030);

insert into customer values(200,'raju','pune',100);

insert into customer values(202,'rama','mumbai',101);

insert into customer values(203,'karan','nashik',101);

insert into customer values(204,'kirti','nashik',100);

insert into account values(1000,200,'saving',2000);

insert into account values(1001,200,'dmat',2050);

insert into account values(1002,203,'dmat',3023);

insert into account values(1002,202,'dmat',3000);

insert into account values(1003,202,'current',4000);

1. Display cusid, name, acid, balance, manager id for all customers
2. Display cusid, name, acid, balance, manager id for all customers who stays in nashik
3. Display cusid, name, acid, balance, manager id for all customers whose relation manager is tanaya
4. Display cusid, name, acid, balance for all customers whose relation manager name starts with r.
5. Display all customer details and account details who has demat account
6. Display all manager details and customer details of all customers who has saving account
7. Display all account details and customer details whose balance > 3000 and customer stays in pune
8. Display all customer details , manager details also display all managers who is not relation manager of any customer
9. Display all customer details , account details also display all customers who has not opened any account
10. Display all customer details , account details and manager details, also display all customers who has not opened any account, and also display all managers who is not relation manager for any customer
11. To find all managers who is not relation manager for any customer.

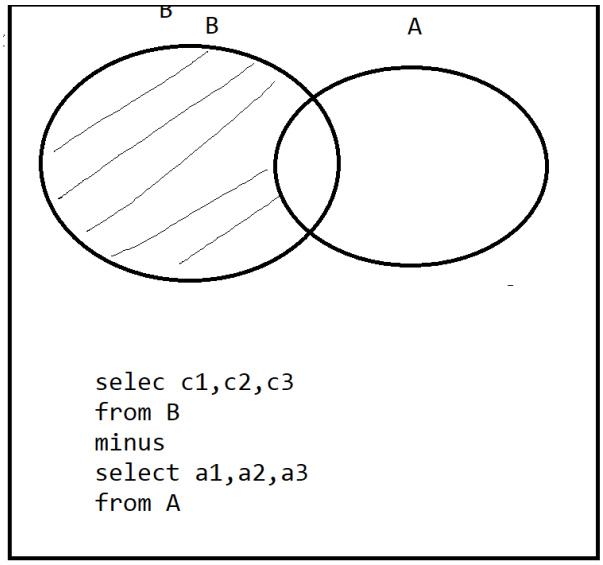
12. Display all manager details whose customer stays in nashik

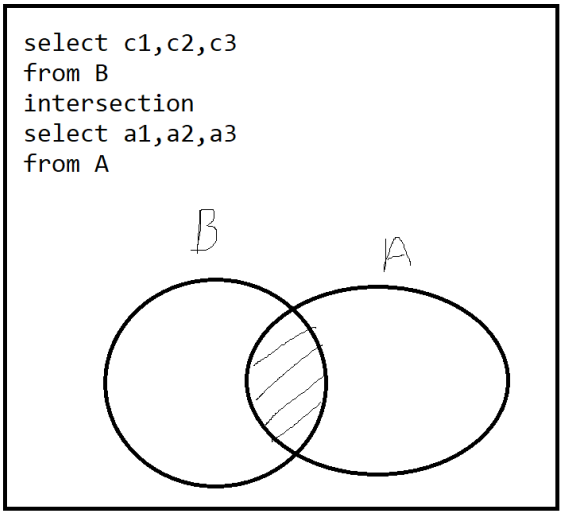
Set operators--- in mysql only union operator works, but in oracle all 3 operators work

Union

Intersection

Minus





Union of 2 queries

```
select empno,ename
    -> from emp
    -> where sal>2000
    -> union
    -> select empno, ename
    -> from emp
    -> where deptno=10;
```

Combining o/p of many tables

```
select *
    -> from emp_us
    -> union
    -> select *
    -> from emp_india
    -> union
```

-> select *

-> from emp_japan;

DDL- Data definition language—these statements are autocommit

Create table	To create new table
Alter table	To modify table structure
Drop table	To delete table and data both
Truncate table	To delete only data from table , it will keep empty table

Both the queries will delete all rows from emp table

Delete from emp;

Truncate table emp;

Difference between truncate and delete

delete	truncate
In delete statement we may use where clause	Where clause cannot be used in truncate
It is DML statement	It is DDL statement
Rollback is possible	Since all DDL statements are autocommit, rollback is not possible

TCL --- transaction control language

Can be used only for DML operation changes

commit	It makes the changes in the table permanent
Rollback Or rollback to <savepoint name>	It undo the changes in the table , if it is not committed, or upto some savepoint
Savepoint A	It add marks in between the statement

To set autocommit off

Set autocommit=0

To set autocommit on

Set autocommit=1

DCL--- data control language

grant	It assigns the permission for user to the table

revoke	It removes permissions for user from the table

- CREATE- allows them to create new tables or databases
- DROP- allows them to them to delete tables or databases
- DELETE- allows them to delete rows from tables
- INSERT- allows them to insert rows into tables
- SELECT- allows them to use the `SELECT` command to read through databases
- UPDATE- allow them to update table rows
- GRANT OPTION- allows them to grant or remove other users' privileges

To assign all permissions to user1 on table emp

```
GRANT ALL PRIVILEGES ON emp TO 'user1'@'localhost';
```

```
GRANT ALL PRIVILEGES ON emp TO 'user1'@'localhost' with grant option
```

To grant all persissions to all databases all tables to newuser

```
GRANT ALL PRIVILEGES ON * . * TO 'newuser'@'localhost';
```

To assign only select, insert permissions to user1 on table emp

```
GRANT select,insert ON test.emp TO 'user1'@'localhost';
```

To make these permissions permanent

```
FLUSH PRIVILEGES;
```

To remove the permission

```
REVOKE type_of_permission ON database_name.table_name FROM
'username'@'localhost';
```

To remove select and create permissions for user 1 on emp

Revoke select,create on test.emp from 'user1'@'localhost'

Temporary table

Temporary table will remain available only till current session is active, it will be deleted once you logout.

Create temporary table mytab(

Id int,

Name varchar(20))engine=MYISAM

ACID properties in database

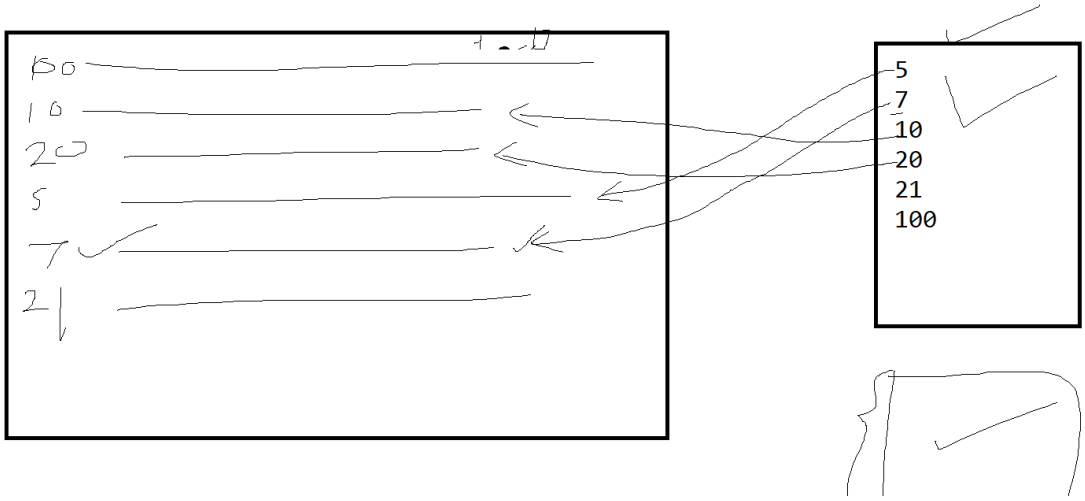
Atomicity	Every transition will be executed as single unit Begin transaction Check a/c Withdraw amount from source a/c Write updated balance in source a/c Deposit amount in destination a/c Write updated balance in destination a/c End transaction
Consistency	If the correctness of data is retained for every transaction
Isolation	If o/p of every transaction is visible to other users only after commit
durability	It supports all these properties for longer period of time

Indexes and views

Indexes are used for faster searching.

Indexes are special files in which it stores keys and its position.

The data in index file is always in the sorted order of keys.



Select *

From emp

Where sal>2000;

Indexes

- Indexes are used to search data faster
- Indexes gets automatically created for primary key and unique key constraints
- Unique indexes do not allow to add duplicate values
- We will not be able to see the index files
- Which index to use for searching will be decided by search engines in mysql(INNODB,MYISAM)
- Indexes are always in sorted order of keys, can be arranged in ascending or descending order

2 types of indexes

1. Clustered index
2. Non clustered index

Clustered vs non clustered

Clustered	Non clustered
There is only one clustered index	There will be many non-clustered index
Clustered indexes are stored with data	These indexes are stored externally
These indexes require no extra space	These indexes require extra space

Indexes always stores key and the position of the key in the table

Types of indexes

primary	Does not allow duplicate keys and null values
unique	Does not allow duplicate keys
fulltext	Searches word, usually used in search engines, works only on text,char and varchar type data
regular	Indexes can be stored in ascending or descending order, can be based on one or more keys

Syntax

Create index idx_dname_loc

On dept(dname)

To create unique index

Create unique index idx_deptno

On dept(dname)

To delete index idx_deptno

Drop index idx_deptno

To see the list of all indexes

show indexes from <table name>;

show indexes from dept;

To find which index is used by your query, then use Explain

explain select *

-> from dept

-> where deptno between 10 and 40;

Views

- Views are virtual table based on base query
- If you fire a query on views then actually base query will get executed

Why use views

1. Give limited access to few columns or rows of existing table.
2. To hide table names, to increase security
3. To hide complexity of the query

Create view:
Create view mgr10
As
Select *
From emp
Where deptno=10;

To create a view so that DMl operations will work only for deptno =10
Create view mgr10
-> As
-> Select *
-> From emp
-> Where deptno=10
-> with check option;

Views can be readonly, but readonly views are not supported by mysql, but works in oracle.

To delete views
Drop view mgr10;
Views which contains all not null columns, and it is based on single table, and it does not contain aggregate functions, or does not contains group by statement, or does not contain union

create view all_emp
-> as
-> select * from emp_us

```
   -> union
   -> select * from emp_india
   -> union
   -> select * from emp_japan;
```

To create materialized view

```
create materialized view all_emp
   -> as
   -> select * from emp_us
   -> union
   -> select * from emp_india
   -> union
   -> select * from emp_japan;
```

PL-SQL(procedural language structured query language)
Procedures
Functions
Triggres
Exception
Cursors

Statements
If statement

Loops
While
For
Repeate until
Loop..endloop

Procedures	These are blocks of code which gets executed on the database server
Functions	These are blocks of code which gets executed on the database server and it returns one value Functions can be called in select statement and where clause
Triggres	These are block of code which gets executed automatically, when some DML operation gets executed

Why are using PL SQL

1. We can hide table names from developers.
2. Complexity of queries can be hidden.
3. We may add multiple queries in one procedure and give the o/p to java/python program, so interactions between middleware programs and database server can be reduced.
4. Which also reduces network traffic
5. It increases security, by hiding table names from developers

```
Delimiter //
```

```
Create procedure <name of procedure>(parameters)
Begin
   Select * from emp;
End//
```

```
delimiter //
Create procedure myproc()
Begin
   Select * from emp;
End//
delimiter ;
Call myproc();
```

```
Delimiter //
Create procedure insrec(pid int,pname varchar(20),qty int,price double(9,2))
Begin
    Insert into product values(pid,pname,qty,price);
End//
Dlimiter ;
Call insrec(10,'50-50',34,56)
```

```
-----to find number of employees in dept 10
Select count(*) from emp
Where deptno=10
```

In a procedure we can pass 3 types of parameters

in	By default the parameters are of type in, these are read only parameters and used only for reading data
out	these are write only parameters and used only for getting data as o/p
inout	These read and write parameters, can be used for sending data i/p and getting modified information as o/p

```
Delimiter //
Create procedure getcnt_by_dept(pdno int,out cnt int)
Begin
Select count(*) into cnt from emp where deptno= pdno;
```

```
End//
Delimiter ;
Call getcnt_by_dept(10,@c)


Select @c;


Delimiter //
Create procedure getcnt_by_dept(pdno int,out cnt int,out minsal double(9,2))
Begin
Select count(*),min(sal) into cnt,minsal from emp where deptno= pdno;
End//
Delimiter ;
Call getcnt_by_dept(10,@c,@m);
Select @c,@m;
```

```
delimiter //
create procedure get_cnt_min(in dno int,out cnt int,out minsal double(9,2))
   begin
select count(*) ,min(sal) into cnt,minsal
from emp
where deptno=dno;
end//
delimiter ;
 ----write procedure to increase count by 10
delimiter //
create procedure inccnt(inout cnt int)
begin
set cnt =cnt+10;
end//
delimiter ;
set c=5



call inccnt(@c)
```

```
select @c;
```

write a procedure to display all employees in given dept and sal > given sal

```
delimiter //
create procedure disp_data(dno int,s double(9,2))
begin
   select *
  from emp
 where deptno=dno and sal>s;
end//
delimiter ;
call disp_data(10,2000);
```

-------- write a procedure to assign remark based on comm

If comm is null or =0 then ‘need improvement’

Else if comm<300 then ‘ok’

Else if comm>=300 and comm <500 then ‘good’

Otherwise ‘excellent’

```
Delimiter //
Create procedure get_remark(peno int, out remark varchar(20))
Begin
Declare vcomm int default 0;
Select comm into vcomm
From emp
Where empno=peno;
If vcomm is null or vcomm=0 then
     Set remark=’need improvement’;
Elseif vcomm<300 then
    Set remark=’ok’;
Elseif vcomm>=300 and vcomm<500 then
    Set remark=’good’;
Else
    Set remark=’excellent’;
```

```
End if;
Select remark
End//
Delimiter ;
Call get_remark(7902,@remark );
Select @remark;
```

------write a procedure to find net Sal of employee by using formula sal+comm for given empno

If netsal <1000 then “less”

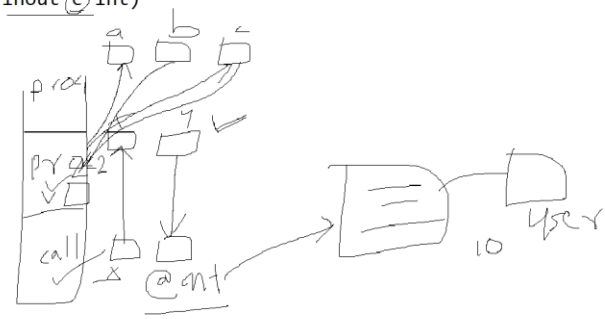
Else netsal>=1000 and <2000 ‘ok’

Else netsal >=2000 and <3000 ‘good’

Else “better”

```
Delimiter //
Create procedure get_sal(pempno int,out psc double(9,2),out remark varchar(20))
Begin
Select sal+ifnull(comm,0) into psc
From emp where empno=pempno;
If psc <1000 then
  Set remark='less';
Elseif psc>=1000 and psc<2000 then
 Set remark='ok';
Elseif psc>=2000 and psc<3000 then
Set remark='good';
Else
   Set remark='better';
End if;
Select psc,remark;

End//
```

```
delimiter //
create procedure proc1(in a int,out b varchar(10),inout c int)
begin
declare var1 int
end//
```



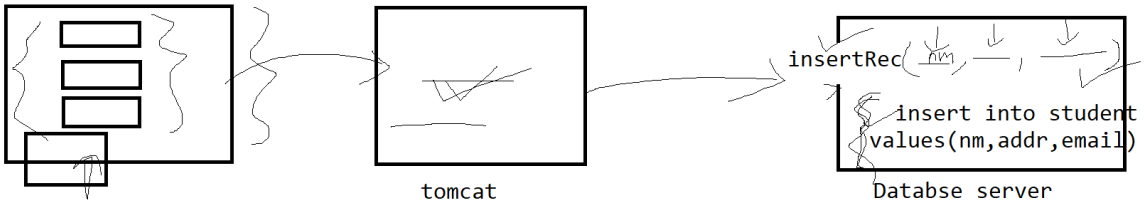
```
create procedure proc2(in x int, out y int)
begin
declare v int
 select cnt(*) into f
 call proc1 (x,@d,@f)
end//

delimiter ;

set x=12
 call proc2 ( x ,@cnt)
```

session variable

```
set x=12
 call proc2 ( x ,@cnt)
```

session variable

Execution flow in 3 tier application



HTML
CSS
JAVASCRIPT
REACT

tomcat
spring
servlet-jsp
.net-c#
python
PHP
PERL

Databse server
Backend
ORacle
mysql
postgeSQL

MongoDB

parameters

In—used to take input

Out – used to send output

Inout ---used to get i/p and send o/p

Write a procedure getdiscount to find discount% and discounted amount from product table for a particular product

If price <20 3%

Else >=20 <40 7%

Else discount 10%

```
Delimiter //
Create procedure getdiscount(in ppid int,out discount float(5,2),out dis_amt double(9,2))
Begin
Declare vprice double(9,2);
select price into vprice
from product where pid=ppid;
if vprice <20 then
set discount = 0.03;
elseif vprice <40 then
set discount=0.07;
else
set discount=0.1;
end if;
set dis_amt= vprice - vprice*discount;

select discount,dis_amt;
end //
delimiter ;
call getdiscount(10,@discount,@amount)
```

In PLSQL there are 3 loops

While expression do
Statements
End while;
	This is top tested loop, will repeat statements till the condition is true

REPEAT statements; UNTIL expression END REPEAT	This is bottom tested loop, will repeat statements until the given condition is false

Label1:Loop
If condition then
Leave Label1
End if
endloop
	This is infinite loop , will continue execution till leave statement gets executed, leave statement is same as break statement, it forcefully stops the loop.
	
	In this loop you may use iterate statement, it is similar to continue statement in java,
	It will transfer the control to the beginning of the loop.

1. Write a procedure which accepts start and stop values and display all numbers between start and stop

2. Example displaydata(10,20) o/p 10,11,12,13,14,15……20

```
Delimiter //

Delimiter //

Create procedure displaydata(in start int,stop int)

Begin

Declare cnt int;

Declare str varchar(100) default '';

Set cnt=start;

While cnt<=stop do

   set str=concat(str,cnt,',');

   Set cnt=cnt+1;

End while;

set str=substr(str,1,length(str)-1);

Select str;

End//

Delimiter ;Delimiter ;
```

3. Write a procedure to accept a number from user and display its factorial

```
Delimiter //
Create procedure displayfactorial(in num int,out fact int)
Begin
  Declare start int default 1;
 Set fact=1;
  While start<=num do
      Set fact=fact*start;
      Set start=start+1;
  End while;
End//
```

Using repeat until loop

1. Write a procedure which accepts start and stop values and display all numbers between start and stop(use repeat …until loop)

```
Delimiter //

Create procedure  displaydatarepeat(in start int, in stop int)

Begin

   Declare cnt int default start;

   Declare str varchar(100) default '';

    Repeat

     Set str=concat(str,cnt,',');
```

```
    Set cnt=cnt+1;

    Until  cnt > stop

  End repeat;

Set str=substr(str,1,length(str)-1);

Select str;

End//

Delimiter ;
```

2. Write a procedure to find factorial of a number(repeat until)

```
Delimiter //
Create procedure displayfactorialrepeat(in num int, out fact int)
Begin
Declare start int default 1;
Set fact=1;
Repeat
   Set fact=fact*start;
   Set start=start+1;
   Until start>num
End repeat;
Select fact;
End//

Delimiter ;
```

Loop …endloop

3. Write a procedure which accepts start and stop values and display all numbers between start and stop(use loop …end loop)

```
Delimiter //
Create procedure displaydataloop(in start int,in stop int)
Begin
Declare str varchar(100) default ‘’;
Declare cnt int default start;

L1:Loop

   Set str=concat (str,cnt,’,’);

   Set cnt=cnt+1;

   If cnt>stop then

      Leave l1;

   End if;

End loop;

 Set str=substr(str,1,length(str)-1)

 Select str;
```

```
End//
Delimiter ;
```

4. Write a procedure to find factorial of a number using loop …end loop;

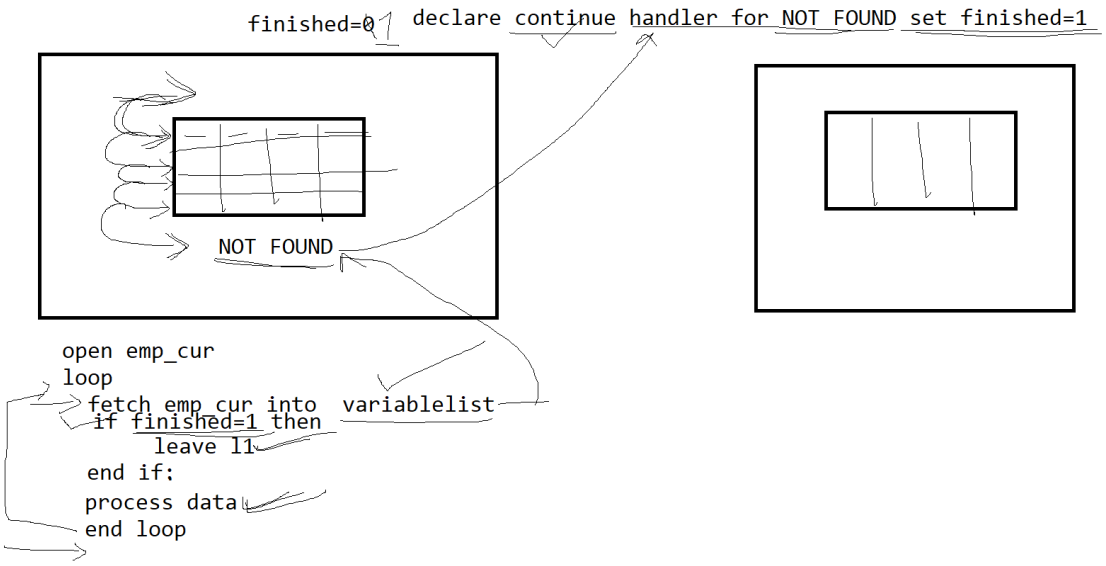
```
Delimiter //
Create procedure displayfactorialloop(in num int,out fact int)
Begin
   Declare start int default 1;
   Set fact=1;
   L1:loop
      Set fact=fact*start;
      Set start=start+1;
      If start >num then
          Leave l1;
       End if;
   End loop
   Select fact;
End//
```

Cursors

Cursors are used to read the data from the table row by row, and process it

Step by step procedure to use cursor

1. Declare cursor.
2. declare continue handler to stop the loop
3. open the cursor.
4. fetch the row from the cursor.
5. check whether reached to last row leave the loop
6. process the row.
7. goto step 4
8. once come out of the loop then close the cursor.



Step by step

1. Set the variable finished to 0;
   Declare finished int default 0;
   Declare variables for cursor
2. Declare cursor

   Declare emp_cur CURSOR for select * from emp;

3. Declare continue handler for NOT FOUND
   Declare continue handler for NOT FOUND set finished=1;
4. Open cursor
   Open emp_cur;
5. Fetch the cursor one row at a time
   Fetch emp_cur into vempvo,venm,vjob,vhiredate,vmgr,vsal,vcomm,vdeptno
6. Check value of finished

   If finished=1 then

   Leave l1;

   End if;

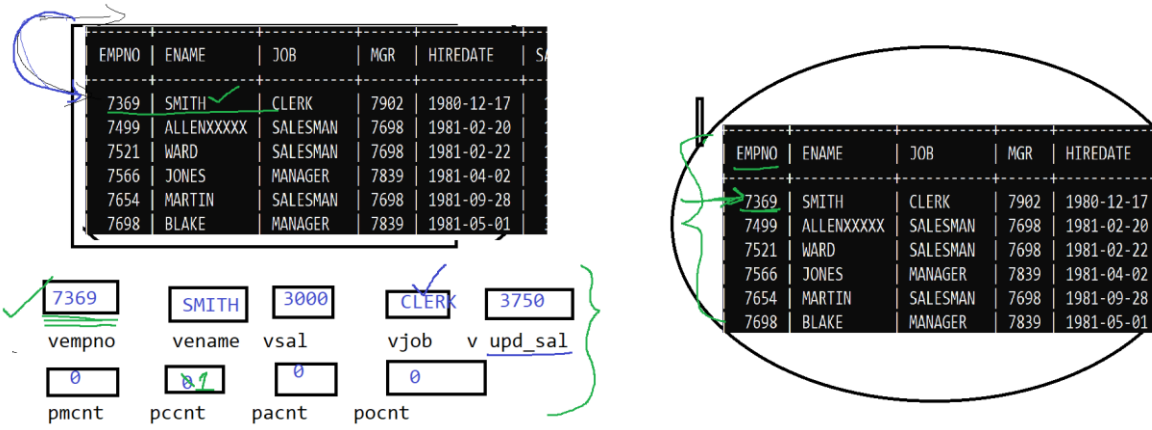
7. Process data
8. Repeat steps 5 to 7 till we do not leave the loop
9. Once come out of the loop then close the cursor
   Close emp_cur

   Example

   1. Write a procedure to display employee details row by row using cursor
      delimiter //
      create procedure display_emp_cur()
      begin
       declare finished int default 0;
       declare vempno,vmgr,vdeptno int;

```
 declare vename,vjob varchar(20);
 declare vhiredate date;
 declare vsal,vcomm double(9,2);
 declare emp_cur cursor for select * from emp;
 declare continue handler for NOT FOUND set finished=1;

 open emp_cur;
 l1:loop
   fetch emp_cur into
vempno,vename,vjob,vmgr,vhiredate,vsal,vcomm,vdeptno;
   if finished=1 then
     leave l1;
     end if;
   select vempno,vename,vjob,vsal,vcomm;
  end loop;
  close emp_cur;
end//
delimiter ;
```

2. Update sal of employee, also give cnt of each type as output.
   If manager, then increase it by 10%
   If analyst, then increase by 20%
   If CLERK, the increase by 25%
   Otherwise increase by 8%




```
delimiter //
create procedure update_emp_sal(out pmcnt  int,out pacnt  int,out pccnt  int,out pocnt  int)
begin
 declare finished int default 0;
 declare vempno,vmgr,vdeptno int;
 declare vename,vjob varchar(20);
 declare vhiredate date;
 declare vsal,vcomm,vupd_sal double(9,2);
 declare emp_cur cursor for select * from emp;
 declare continue handler for NOT FOUND set finished=1;
 set pmcnt=0;
```

```
 set pacnt=0;
 set pccnt=0;
 set pocnt=0;
 open emp_cur;
 l1:loop
    fetch emp_cur into vempno,vename,vjob,vmgr,vhiredate,vsal,vcomm,vdeptno;
    if finished=1 then
       leave l1;
 end if;
 if vjob='manager' then
    set vupd_sal=1.1*vsal;
    update emp
    set sal=1.1*sal
   where empno=vempno;
   set pmcnt=pmcnt+1;
 elseif vjob='analyst' then
    set vupd_sal=1.2*vsal;
    update emp
    set sal=1.2*sal
   where empno=vempno;
   set pacnt=pacnt+1;
 elseif vjob='clerk' then
    set vupd_sal=1.25*vsal;
    update emp
    set sal=1.25*sal
   where empno=vempno;
   set pccnt=pccnt+1;
 else
    set vupd_sal=1.08*vsal;
    update emp
    set sal=1.08*sal
   where empno=vempno;
   set pocnt=pocnt+1;
 end if;
 select vempno,vename,vjob,vsal,vcomm,vupd_sal;
   end loop;
   select pmcnt,pacnt,pccnt,pocnt;
   close emp_cur;
 end//
 delimiter ;
```

3. Write a procedure to update price of product using cursor.
   If the category is chips then increase the price by 10%
   If the category is cold drink then increase the price by 20%
   Else increase by 8 %

```
delimiter //
create procedure changeprice()
begin
  declare finished int default 0;
```

```
declare vpid,vcid,vchipscatid,vdrinkcatid,vqty int;
declare vprice double(9,2);
declare vname varchar(20);
declare prod_cur cursor for select * from product;
declare continue handler for NOT FOUND set finished=1;

open prod_cur;
select cid into vchipscatid
from category
where cname='chips';

select cid into vdrinkcatid
from category
where cname='cold drink';
l1:loop
  fetch prod_cur into vpid,vname,vqty,vprice,vcid;
    if finished=1 then
      leave l1;
    end if;
    if vcid = vchipscatid then
       update product
              set price=1.1*ifnull(price,1)
              where pid=vpid;
    elseif vcid = vdrinkcatid then
       update product
              set price=1.2*ifnull(price,1)
              where pid=vpid;
    else
       update product
              set price=1.08*ifnull(price,1)
              where pid=vpid;
    end if;

  end loop;
end//
```

4. Write a procedure to find comma separated list of emails.

```
delimiter //
create procedure generate_email()
begin
  declare str varchar(1000) default '';
  declare vemail,vename,vjob varchar(50);
  declare finished int default 0;
  declare emp_cur cursor for select ename,job from emp;
  declare continue handler for NOT found set finished=1;

  open emp_cur;
  l1:loop
    fetch emp_cur into vename,vjob;
      if finished=1 then
```

```
          leave l1;
        end if;
        if vjob is not null then
                set
vemail=concat(substr(vename,1,3),'.',substr(vjob,1,3),'@mycompany.com');
                set str=concat(str,vemail,',');
        end if;

  end loop;
  close emp_cur;
  select str;

end//
delimiter ;
```

To see the list of all procedures and functions

```
SELECT ROUTINE_DEFINITION
FROM information_schema.ROUTINES WHERE
SPECIFIC_NAME='procedurename'

SHOW FUNCTION STATUS WHERE Db = 'db_name';
SHOW procedure STATUS WHERE Db = 'db_name';
```

difference between functions and procedure

procedure	function
1.it doesnot return any value	it returns a single value
2. use call statement to call a procedure, you cannot use it in select statement	we can call functions in select statement

to allow create functions

```
SET GLOBAL log_bin_trust_function_creators = 1;
```

Function

When you want to return one value as output then write functions

1. Write a function to generate email

```
Delimiter //
Create function get_email(name varchar(20),jb varchar(20)) returns varchar(50)
Begin
   Declare email varchar(50)
   Set email=concat((substr(name,1,3),'.',substr(jb,1,3),'@mycompany.com');
   Return email;
End//
Delimiter ;
```

2. Write a function to find exp

```
delimiter //
create function get_exp(hdate date) returns int
begin
  declare exp int;
  set exp=floor(datediff(curdate(),hdate)/365);
  return exp;
end//
```

3. Write a function which accepts price and qty as i/p and returns discounted price.

   If qty < 20 then apply 10% discount on price
   Else if qty >= 20 and <=30 discount 20%
   Otherwise 30% discount

```
Delimiter //
Create function get_discount(pr double(9,2), qty int) returns double(9,2)
Begin
   Declare dis_price double(9,2) default 0;
If qty!=0 then
    If qty<20 then
        Set dis_price=0.9*pr;
    Elseif qty<30 then
       Set dis_price=0.8*pr;
    Else
       Set dis_price=0.7*pr;
   End if;

End if;

   Return dis_price;

End//
```

Tiggers

Tiggers are procedures which are automatically called
Used for monitoring DML activities on tables by all users.

Tiggers can be executed either before or after the dml statement,
There are 2 types of triggers

1. Row level trigger
2. Statement level trigger –this trigger does not work in mysql

In mysql we can use row level trigger on all DML operation

Timings

1. Before---- before trigger gets executed before the actual statement
2. After----- after trigger gets executed after execution of the actual statement

3. Insteadof ----these triggers are used only on views, but mysql doe not support it

Before creation of trigger we need to create a table to store the required information needed for monitoring the table

```
Create table empsecurity(
Empno int,
Ename varchar(20),
Action varchar(20),
Oldsal double(9,2),
Newsal double(9,2),
Uname varchar(20),
Act_date date);
```

```
Create trigger emp_update before update on emp
For each row
Begin
   Insert into empsecurity values(OLD.empno,OLD.ename,'update',OLD.sal,NEW.sal,user(),curdate())
End//
```

Tiggers

Triggers are block of code which gets executed automatically.

Trigger can be written on all DML operation in mysql

In mysql only row level triggers are allowed. Statement level triggers are not allowed in mysql.

If we want to write the trigger, then the
first step is to decide which monitoring table you want to create, and what fields will be stored in the table.

Step2 – what will be the trigger timing.

Tigger timing can either before or after

The timing can be insteadof --→ these triggers are not supported by mysql. But are supported by oracle, these triggers are only used with views.

Step3 ---- on which action the trigger can be executed.

Action can be ----→insert, update or delete

Step 4------ decide which table the trigger should monitor

Step 5-----decide trigger level

The trigger level can be row level or statement level

Statement level----→ for each DML statement only one entry will be there

Example: if a delete statement, deletes 10 rows , then we need only one entry in the table, then write statement level trigger

These triggers are not supported by mysql , but are supported by oracle.

Row level trigger--→ for each DML statement one entry will be there, for each row getting affected by the dml statement.

Example: if a delete statement, deletes 10 rows , then we need 10 entries in the table, then write row level trigger

Syntax :

Create trigger <name> {before|after} {action} on <table name>

For each row

Begin

    statements

End//

Where to use triggers:

1. For monitoring changes happening in the table by all users
2. To manage data in complex view
3. To maintain integrity of denormalized data

In trigger there are 2 special variables NEW and OLD

	OLD	NEW
insert	null	Will have row that will be created after insertion is done
delete	Will have row that exists in the table	Null
Update	Will have row that exists in the table	Will have row that will be created after changes are applied

To access data from OLD and NEW variables
Example OLD.empno, OLD.ename
NEW.sal

In mysql to get the name of current user the function is user()

```
Create table empsecurity(
Empno int,
Ename varchar(20),
Action varchar(20),
Oldsal double(9,2),
Newsal double(9,2),
Uname varchar(20),
Act_date date);
```

1. If any user insert data in emp table then insert a record in empsecurity table.

```
Delimiter //
Create trigger insertemp after insert on emp
For each row
Begin
  Insert into empsecurity values(NEW.empno,NEW.ename,'insert',null,NEW.sal,user(),curdate());
End//
Delimiter ;
```

1. If any user delete data in emp table then insert a record in empsecurity table.

```
Delimiter //
Create trigger deleterec after delete on emp
For each row
Begin
  Insert into empsecurity values(OLD.empno,OLD.ename,'delete',OLD.sal,null,user(),curdate());
End//
Delimiter ;
```

```
Create table discounts(
Pid int,
 Disc_percent int)
```

```
 Create table product_dis(
Pid int,
Pname varchar(20),
Price double(9,2),
Discounted_amt double(9,2))
```

discounts

Pid	Disc_percent
1	3
2	20

product_dis

pid	pname	price	Discounted_amt
1	chair	2000	1940

Write a trigger to update disounted _amt in product_dis table , as soon as we change Disc_percent in discounts table

```
Update discounts
Set  Disc_percent =7
Where pid=1;
```

	pid	Disc_percent
old	1	3
new	1	7

```
Create trigger update_discount after update on discounts

For each row

Begin

Update product_dis

Set Discounted_amt=price-price*(NEW. Disc_percent  /100) ,pid=NEW.pid

Where pid=OLD.pid;

 End //
```

Exception handling

Any error that occurs at run time, because user has entered wrong data, and these errors can be handled programmatically, then it is called as exceptions, otherwise it is called as errors.

1. Types of exception in mysql
   SQLEXCETION
   SQLSTATE 23000
   Error code
   NOT FOUND

2. Handlers
   Continue--- continue handler will handle the error and resume the execution of the procedure
   Exit--- exit handler will handle the error and stop the execution of the procedure

   To declare handler
   Declare (exit|continue) handler for SQLEXCEPTION select 'error occurred';
   Declare (exit|continue) handler for SQLEXCEPTION begin
   select 'error occurred';
   set finished=1;
   rollback;
   end;

   1. Write a procedure to insert a record in dept, if department number is duplicate then show message error occurred.

```
Delimiter //

Create procedure insertdept(in pdno int,in pdnm varchar(20),in ploc varchar(20))

Begin

   Declare exit handler for SQLEXCEPTION select "error occurred";

   Insert into dept values(pdno,pdnm,ploc);

  Select * from dept;
```

End//

Delimiter ;

2. Create procedure to insert record in product table, if any error occurred because of duplicate pid show error message duplicate key, if error occurred because of -ve qty or -ve price then show error message values cannot be -ve, otherwise show error message error occurred.

```
Delimiter //
Create procedure insertproduct(in ppid int, pnm varchar(20), pqty int,pprice double(9,2),pcid int)
Begin
   Declare continue handler for 1062 select 'duplicate key' msg;
     Declare exit handler for 3819 select 'value should be > 0' msg;
   Declare continue handler for SQLEXCEPTION select 'error occurred' msg;
   Insert into product values(ppid,pnm,pqty,pprice,pcid);
    Select * from product;
End//
```

Normalization

For proper data modelling we use rules of normalization and ER diagram

Acid	cid	Cname	address	balance	type	Relmgrid	relmgrname
1	100	Kishori	Aundh	4567	saving	120	ANIL
2	100	Kishori	Baner	5555	curret	120	ANIL
3	100	Kishori	Baner	6666	demat	120	ANIL
5	101	Rajan	Baner	7777	saving	121	Bhavika
null	102	Atharva	Aundh			122	Revati

Insertion anamoly—

In the above table primary key is acid.

Hence if any new relation manager joins the bank, then unless I assign any account to the manager, we will not be able to add the record

If any customer comes for enquiry, and does not open account still I will not be able to add customer details in the table

This problem is called as insertion anamoly

Updation anamoly

In above table if Kishori submits the request for change in the address with a/c no 1, then the change may happen only in one account,and may keep old address for account 2 and 3, which creates a problem.

This is called as updation anamoly.

Deletion anamoly:

In the above table if Rajan closes the a/c, so we will delete the record from the table,

Along with that record we will loose the customer information, and also looe information of relation manager Bhavika.

This is called as deletion anamoly.

To remove these problems we need to divide the table into multiple tables

Acid	cid	balance	type
1	100	4567	saving
2	100	5555	curret
3	100	6666	demat
5	101	7777	saving

cid	Cname	address	Relmgrid
100	Kishori	baner	120
101	Rajan	Baner	121
102	Atharva	Aundh	122

Relmgrid	relmgrname
120	ANIL
121	Bhavika
122	Revati

Rules of normalization

1NF

According to the E.F. Codd, a relation will be in 1NF, if each cell of a relation contains only an atomic value.

studid	name	marks	course
1	Revati	99,89,96	Java,.net,c++
2	Ashu	88,89,95	Java,.net,c++

Since marks column contains multiple values so the given table is not in 1NF

studid	name	course	marks
1	Revati	java	99
1	Revati	.net	89

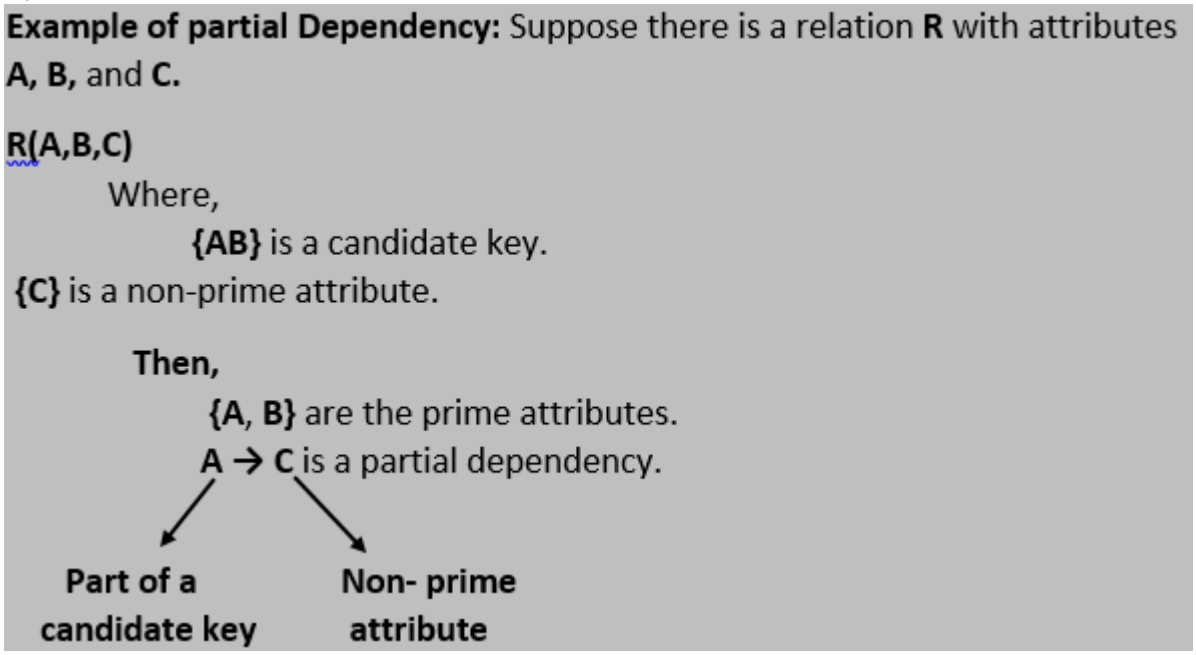
1	Revati	C++	96
2	Ashu	java	88
2	Ashu	.net	89
2	Ashu	C++	95

2NF

According to the E.F. Codd, a relation is in **2NF,** if it satisfies the following conditions:

- A relation must be in 1NF.
- And the candidate key in a relation should determine all non-prime attributes or no partial dependency should exist in the relation.
  Prime attribute: the fields which are part of candidate key(Primary key) are called as prime attribute
  Non prime attributes: the fields which are not part of candidate key(Primary key) are called as non prime attribute

Partial Dependency: If a non-prime attribute can be determined by the part of the candidate key in a relation, it is known as a partial dependency. Or we can say that, if L.H.S is the proper subset of a candidate key and R.H.S is the non-prime attribute, then it shows a partial dependency.
Example of partial Dependency: Suppose there is a relation **R** with attributes **A, B,** and **C.**



studid	name	course	marks
1	Revati	java	99
1	Revati	.net	89
1	Revati	C++	96
2	Ashu	java	88

2	Ashu	.net	89
2	Ashu	C++	95

Primary key -→{studid,course}

Prime attributes: studid,course

Non prime attributes: name, marks

Studid-->name

Course-->

Studid+course→marks

studid	course	marks
1	java	99
1	.net	89
1	C++	96
2	java	88
2	.net	89
2	C++	95

Student

studid	name
1	Revati
2	Ashu

3NF

According to the E.F. Codd, a relation is in **third normal form (3NF)** if it satisfies the following conditions:

- A relation must be in second normal form (2NF).
- And there should be no transitive functional dependency exists for non-prime attributes in a relation.
- Third Normal Form is used to achieve data integrity and reduce the duplication of data.

A relation is in 3NF if and only if any one of the following conditions will satisfy for each non-trivial functional dependency **X→ Y:**

1. X is a super key or candidate key
2. And, Y is a prime attribute, i.e., Y is a part of candidate key.

Transitive Dependency: If **X → Y and Y→ Z** are two functional dependencies, **X → Z** is called as a transitive functional dependency.

Acid	cid	Cname	address	balance	type	Relmgrid	relmgrname
1	100	Kishori	Baner	4567	saving	120	ANIL
2	100	Kishori	Baner	5555	curret	120	ANIL
3	100	Kishori	Baner	6666	demat	120	ANIL
5	101	Rajan	Baner	7777	saving	121	Bhavika

a/c id---→ cid--→cname, address,relationmgrid,relname

the table is not in 3NF

cid	Cname	address	Rel id
100	Kishori	Baner	120
101	Rajan	Baner	121

Relmgrid	relmgrname
120	ANIL
121	Bhavika

Acid	cid	balance	type
1	100	4567	saving
2	100	5555	curret
3	100	6666	demat
5	101	7777	saving

BCNF(3.5 NF)

Boyce-Codd Normal Form (BCNF) is the advance version of the third normal form (3NF) that's why it is also known as a **3.5NF**
According to the E.F. Codd, a relation is in **Boyce-Codd normal form (3NF)** if it satisfies the following conditions:

- A relation is in 3NF.
- And, for every functional dependency, **X** → **Y,** L.H.S of the functional dependency (X) be the super key of the table.

Normalize the following tables upto 3 NF form

Proj Code	Proj Type	Proj Desc	Empno	Ename	Grade	Sal scale	Proj Join Date	Alloc Time
001	APP	LNG	46	JONES	A1	5	12/1/1998	24
001	APP	LNG	92	SMITH	A2	4	2/1/1999	24
001	APP	LNG	96	BLACK	B1	9	2/1/1999	18
004	MAI	SHO	72	JACK	A2	4	2/4/1999	6
004	MAI	SHO	92	SMITH	A2	4	5/5/1999	6

- Orderno
- Orderdate
- Itemno
- Qty
- Price
- Cname
- Custno
- Email
- Orderamt
- Salespersonno
- Salespersonname
- Locationid ----------location from where item dispatched
- Location name

One customer can place many order
One order contains many items
One order will be managed by one salesperson
One order belong to one customer
One order can be dispatched from different location

Tiggers

Triggers are block of code which gets executed automatically.

Trigger can be written on all DML operation in mysql

In mysql only row level triggers are allowed. Statement level triggers are not allowed in mysql.

If we want to write the trigger, then the
first step is to decide which monitoring table you want to create, and what fields will be stored in the table.

Step2 – what will be the trigger timing.

Tigger timing can either before or after

The timing can be insteadof --→ these triggers are not supported by mysql. But are supported by oracle, these triggers are only used with views.

Step3 ---- on which action the trigger can be executed.

Action can be ----→insert, update or delete

Step 4------ decide which table the trigger should monitor

Step 5-----decide trigger level

The trigger level can be row level or statement level

Statement level----→ for each DML statement only one entry will be there

Example: if a delete statement, deletes 10 rows , then we need only one entry in the table, then write statement level trigger

These triggers are not supported by mysql , but are supported by oracle.

Row level trigger--→ for each DML statement one entry will be there, for each row getting affected by the dml statement.

Example: if a delete statement, deletes 10 rows , then we need 10 entries in the table, then write row level trigger

Syntax :

Create trigger <name> {before|after} {action} on <table name>

For each row

Begin

    statements

End//

Where to use triggers:

1. For monitoring changes happening in the table by all users

2. To manage data in complex view
3. To maintain integrity of denormalized data

In trigger there are 2 special variables NEW and OLD

	OLD	NEW
insert	null	Will have row that will be created after insertion is done
delete	Will have row that exists in the table	Null
Update	Will have row that exists in the table	Will have row that will be created after changes are applied

To access data from OLD and NEW variables
Example OLD.empno, OLD.ename
NEW.sal

In mysql to get the name of current user the function is user()

```
Create table empsecurity(
Empno int,
Ename varchar(20),
Action varchar(20),
Oldsal double(9,2),
Newsal double(9,2),
Uname varchar(20),
Act_date date);
```

1. If any user insert data in emp table then insert a record in empsecurity table.

```
Delimiter //
Create trigger insertemp after insert on emp
For each row
Begin
  Insert into empsecurity values(NEW.empno,NEW.ename,'insert',null,NEW.sal,user(),curdate());
End//
Delimiter ;
```

1. If any user delete data in emp table then insert a record in empsecurity table.

```
Delimiter //
Create trigger deleterec after delete on emp
```

```
For each row
Begin
  Insert into empsecurity values(OLD.empno,OLD.ename,'delete',OLD.sal,null,user(),curdate());
End//
Delimiter ;
```

```
Create table discounts(
Pid int,
Disc_percent int)
```

```
Create table product_dis(
Pid int,
Pname varchar(20),
Price double(9,2),
Discounted_amt double(9,2))
```

discounts

Pid	Disc_percent
1	3
2	20

product_dis

pid	pname	price	Discounted_amt
1	chair	2000	1940

Write a trigger to update disounted _amt in product_dis table , as soon as we change Disc_percent in discounts table

```
Update discounts
Set  Disc_percent =7
Where pid=1;
```

	pid	Disc_percent
old	1	3
new	1	7

```
Create trigger update_discount after update on discounts
For each row
Begin
```

Update product_dis

Set Discounted_amt=price-price*(NEW. Disc_percent /100) ,pid=NEW.pid

Where pid=OLD.pid;

End //

Exception handling

Any error that occurs at run time, because user has entered wrong data, and these errors can be handled programmatically, then it is called as exceptions, otherwise it is called as errors.

1. Types of exception in mysql
   SQLEXCETION
   SQLSTATE 23000
   Error code
   NOT FOUND

2. Handlers
   Continue--- continue handler will handle the error and resume the execution of the procedure
   Exit--- exit handler will handle the error and stop the execution of the procedure

   To declare handler
   Declare (exit|continue) handler for SQLEXCEPTION select 'error occurred';
   Declare (exit|continue) handler for SQLEXCEPTION begin
   select 'error occurred';
   set finished=1;
   rollback;
   end;

   1. Write a procedure to insert a record in dept, if department number is duplicate then show message error occurred.

```
Delimiter //

Create procedure insertdept(in pdno int,in pdnm varchar(20),in ploc varchar(20))

Begin

  Declare exit handler for SQLEXCEPTION select "error occurred";

  Insert into dept values(pdno,pdnm,ploc);

 Select * from dept;

End//

Delimiter ;
```

   2. Create procedure to insert record in product table, if any error occurred because of duplicate pid show error message duplicate key, if error occurred because of -ve qty or -

ve price then show error message values cannot be -ve, otherwise show error message error occurred.

```
Delimiter //

Create procedure insertproduct(in ppid int, pnm varchar(20), pqty int,pprice double(9,2),pcid int)

Begin

    Declare continue handler for 1062 select 'duplicate key' msg;

      Declare exit handler for 3819 select 'value should be > 0' msg;

    Declare continue handler for SQLEXCEPTION select 'error occurred' msg;

    Insert into product values(ppid,pnm,pqty,pprice,pcid);

     Select * from product;

End//
```

Normalization

For proper data modelling we use rules of normalization and ER diagram

Acid	cid	Cname	address	balance	type	Relmgrid	relmgrname
1	100	Kishori	Aundh	4567	saving	120	ANIL
2	100	Kishori	Baner	5555	curret	120	ANIL
3	100	Kishori	Baner	6666	demat	120	ANIL
5	101	Rajan	Baner	7777	saving	121	Bhavika
null	102	Atharva	Aundh			122	Revati

Insertion anamoly—

In the above table primary key is acid.

Hence if any new relation manager joins the bank, then unless I assign any account to the manager, we will not be able to add the record

If any customer comes for enquiry, and does not open account still I will not be able to add customer details in the table

This problem is called as insertion anamoly

Updation anamoly

In above table if Kishori submits the request for change in the address with a/c no 1, then the change may happen only in one account,and may keep old address for account 2 and 3, which creates a problem.

This is called as updation anamoly.

Deletion anamoly:

In the above table if Rajan closes the a/c, so we will delete the record from the table,

Along with that record we will loose the customer information, and also looe information of relation manager Bhavika.

This is called as deletion anamoly.

To remove these problems we need to divide the table into multiple tables

Acid	cid	balance	type
1	100	4567	saving
2	100	5555	curret
3	100	6666	demat
5	101	7777	saving

cid	Cname	address	Relmgrid
100	Kishori	baner	120
101	Rajan	Baner	121
102	Atharva	Aundh	122

Relmgrid	relmgrname
120	ANIL
121	Bhavika
122	Revati

Rules of normalization

1NF

According to the E.F. Codd, a relation will be in 1NF, if each cell of a relation contains only an atomic value.

studid	name	marks	course
1	Revati	99,89,96	Java,.net,c++
2	Ashu	88,89,95	Java,.net,c++

Since marks column contains multiple values so the given table is not in 1NF

studid	name	course	marks
1	Revati	java	99
1	Revati	.net	89
1	Revati	C++	96
2	Ashu	java	88
2	Ashu	.net	89
2	Ashu	C++	95

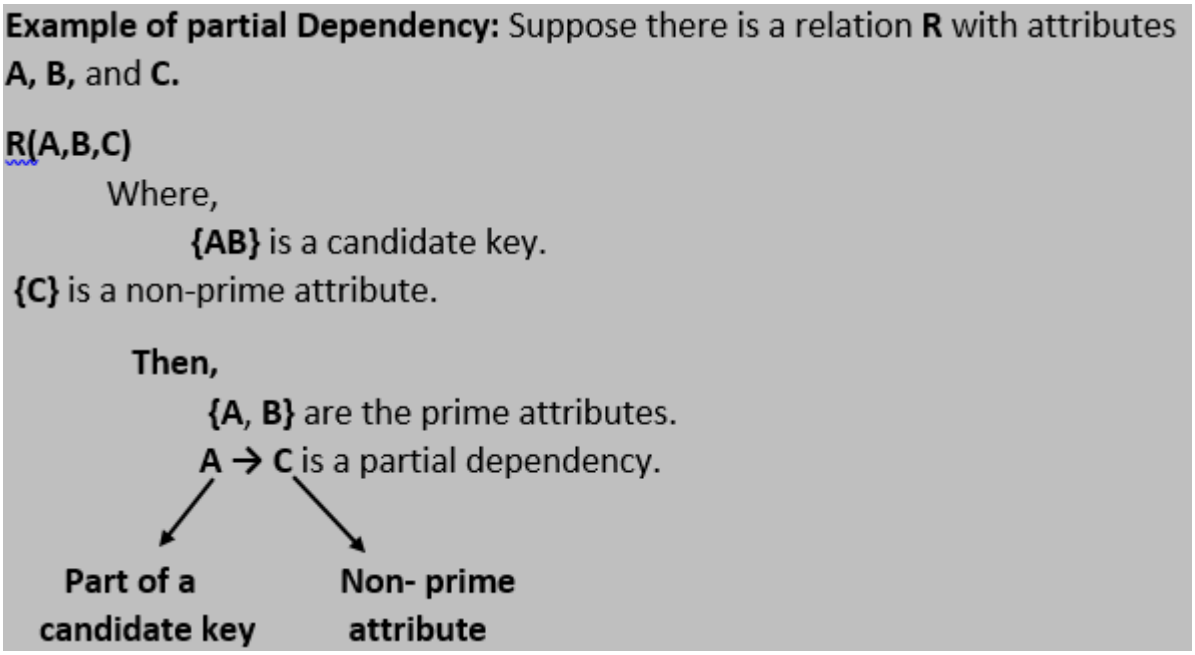
2NF

According to the E.F. Codd, a relation is in **2NF,** if it satisfies the following conditions:

- A relation must be in 1NF.

- And the candidate key in a relation should determine all non-prime attributes or no partial dependency should exist in the relation.
  Prime attribute: the fields which are part of candidate key(Primary key) are called as prime attribute
  Non prime attributes: the fields which are not part of candidate key(Primary key) are called as non prime attribute

Partial Dependency: If a non-prime attribute can be determined by the part of the candidate key in a relation, it is known as a partial dependency. Or we can say that, if L.H.S is the proper subset of a candidate key and R.H.S is the non-prime attribute, then it shows a partial dependency.
Example of partial Dependency: Suppose there is a relation **R** with attributes **A, B,** and **C.**



studid	name	course	marks
1	Revati	java	99
1	Revati	.net	89
1	Revati	C++	96
2	Ashu	java	88
2	Ashu	.net	89
2	Ashu	C++	95

Primary key -→{studid,course}
Prime attributes: studid,course
Non prime attributes: name, marks
Studid-->name
Course-->
Studid+course→marks

studid	course	marks

1	java	99
1	.net	89
1	C++	96
2	java	88
2	.net	89
2	C++	95

Student

studid	name
1	Revati
2	Ashu

3NF

According to the E.F. Codd, a relation is in **third normal form (3NF)** if it satisfies the following conditions:

- A relation must be in second normal form (2NF).
- And there should be no transitive functional dependency exists for non-prime attributes in a relation.
- Third Normal Form is used to achieve data integrity and reduce the duplication of data.

A relation is in 3NF if and only if any one of the following conditions will satisfy for each non-trivial functional dependency **X→ Y:**

1. X is a super key or candidate key
2. And, Y is a prime attribute, i.e., Y is a part of candidate key.

Transitive Dependency: If **X → Y and Y→ Z** are two functional dependencies, **X → Z** is called as a transitive functional dependency.

Acid	cid	Cname	address	balance	type	Relmgrid	relmgrname
1	100	Kishori	Baner	4567	saving	120	ANIL
2	100	Kishori	Baner	5555	curret	120	ANIL
3	100	Kishori	Baner	6666	demat	120	ANIL
5	101	Rajan	Baner	7777	saving	121	Bhavika

a/c id---→ cid--→cname, address,relationmgrid,relname

the table is not in 3NF

cid	Cname	address	Rel id
100	Kishori	Baner	120
101	Rajan	Baner	121

Relmgrid	relmgrname
120	ANIL
121	Bhavika

Acid	cid	balance	type

1	100	4567	saving
2	100	5555	curret
3	100	6666	demat
5	101	7777	saving

BCNF(3.5 NF)

Boyce-Codd Normal Form (BCNF) is the advance version of the third normal form (3NF) that's why it is also known as a **3.5NF**
According to the E.F. Codd, a relation is in **Boyce-Codd normal form (3NF)** if it satisfies the following conditions:

- A relation is in 3NF.
- And, for every functional dependency, **X → Y,** L.H.S of the functional dependency (X) be the super key of the table.

we have a relation R with three columns: Id, Subject, and Professor. We have to find the highest normalization form, and also, if it is not in BCNF, we have to decompose it to satisfy the conditions of BCNF.

Id	Subject	Professor
101	Java	Mayank
101	C++	Kartik
102	Java	Sarthak
103	C#	Lakshay
104	Java	Mayank

Interpreting the table:

- One student can enroll in more than one subject.
- Example: student with **Id 101** has enrolled in **Java and C++**.
- Professor is assigned to the student for a specified subject, and there is always a possibility that there can be multiple professors teaching a particular subject.
- Every professor is teaching only on course

Check if any nonprime attribute gives you any prime attribute, then the table is not in BCNF

Professor	Subject
Mayank	Java
Kartik	C++
Sarthak	Java
Lakshay	C#

Id	Professor
101	Mayank
101	Kartik
102	Sarthak
103	Lakshay
104	Mayank

Normalize the following tables upto 3 NF form

Proj Code	Proj Type	Proj Desc	Empno	Ename	Grade	Sal scale	Proj Join Date	Alloc Time
001	APP	LNG	46	JONES	A1	5	12/1/1998	24
001	APP	LNG	92	SMITH	A2	4	2/1/1999	24
001	APP	LNG	96	BLACK	B1	9	2/1/1999	18
004	MAI	SHO	72	JACK	A2	4	2/4/1999	6
004	MAI	SHO	92	SMITH	A2	4	5/5/1999	6

1NF --- yes

2NF

Primary key

Projcode+empno

Prime attribute--→ proj code, empno

Non prime --→ proj type,proj desc, ename, grade,sal,proj join date,alloc time

Proj code-→ proj type, proj desc

Empno-→ename,grade, sal scale

Projcode+empno---→ project join date, alloc time

Project

Proj Code	Proj Type	Proj Desc
001	APP	LNG
001	APP	LNG
001	APP	LNG
004	MAI	SHO

004 MAI SHO

Employee

Empno	Ename	Grade	Sal scale
46	JONES	A1	5
92	SMITH	A2	4
96	BLACK	B1	9
72	JACK	A2	4
92	SMITH	A2	4

Proj_emp

Proj Code	Empno	Proj Join Date	Alloc Time
001	46	12/1/1998	24
001	92	2/1/1999	24
001	96	2/1/1999	18
004	72	2/4/1999	6
004	92	5/5/1999	6

In employee table empno→grade->salscale so transitive dependency is there hence

It is not in 3NF

employee

Empno	Ename	Grade
46	JONES	A1
92	SMITH	A2
96	BLACK	B1
72	JACK	A2
92	SMITH	A2

grade

Grade	Sal scale

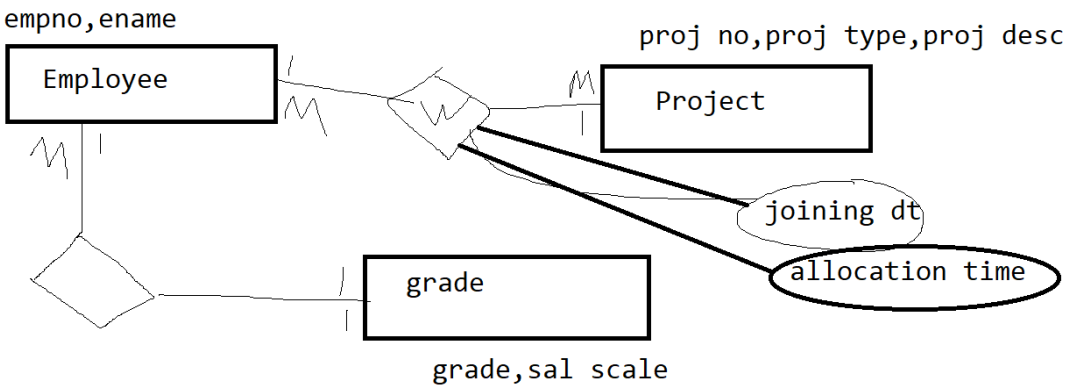
A1 5

A2 4

B1 9

A2 4

A2 4



- **Orderno**
- **Orderdate**
- **Itemno**
- **Qty**
- **Price**
- **Cname**
- **Custno**
- **Email**
- **Orderamt**
- **Salespersonno**
- **Salespersonname**
- **Locationid** ----------**location from where item dispatched**
- **Location name**

One customer can place many order
One order contains many items
One order will be managed by one salesperson
One order belong to one customer
One order can be dispatched from different location

Orderid	Order date	Custno	Cname	Itemno	Item name	stockqty	qty	rate	Buyingprice	
1	10 Apr	100	Xxxx	1000	Tshirt	100	2	3456	2000	

1	10Apr	100	xxxx	2000	jeans				5000	
2	10 Apr	102	yyyyy	1000	Tshirt					
3	12Apr	100	Xxxx	1000	Tshirt					

Orderid+itemno

1. Table is in 1 NF
2. Is it in 2 NF
   Primary key ----→ orderid+itemid
   Prime attribute-→orderid, itemid

   Non prime--→ orderdt, qty, buyingprice,custname, custno, email,orderamt,cname,salespersonid, salesperson name, locid, lname,stockqty,itemrate

Orderid--→orderdate,custno, custname,email, orderamt, salespersonid,sname,

Orderid,itemno--→qty, buying price, , locid, lname

Itemno--→stockqty,item rate

Orderid	Order date	Custno	Cname	email	orderamt	sid	sname
1	10 Apr	100	Xxxx				
1	10Apr	100	xxxx				
2	10 Apr	102	yyyyy				
3	12Apr	100	Xxxx				

Itemno	Item name	stockqty	rate
1000	Tshirt	100	3456
2000	jeans		
1000	Tshirt		
1000	Tshirt		

Orderid	Itemno	Ordered qty	Buyingprice	Locid	loname
1	1000	2	2000		
1	2000		5000		
2	1000				
3	1000				

To check it in 3NF

Primary key ----→ orderid+itemid
Prime attribute-→orderid, itemid

Non prime--→ orderdt , qty, buyingprice, custname, custno, email, orderamt,cname,salespersonid, salesperson name, locid, lname,stockqty,itemrate,stockqty,rate

Locid-→lname

Salespersonid→sname

Custno--→name,email

Order_cust

Orderid	Order date	Custno	orderamt	sid
1	10 Apr	100		
1	10Apr	100		
2	10 Apr	102		
3	12Apr	100		

customer

Custno	Cname	email
100	Xxxx	
102	yyyyy	

saleman

sid	sname

order

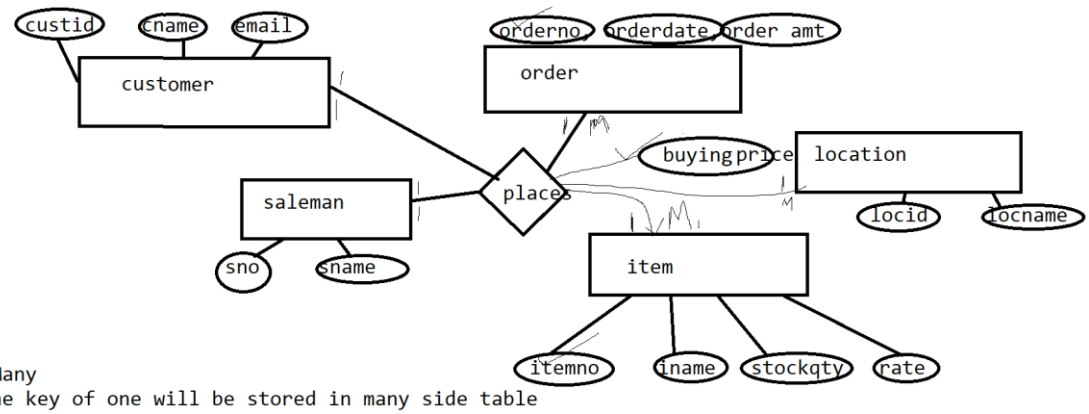
Orderid	Itemno	Ordered qty	Buyingprice	Locid
1	1000	2	2000	
1	2000		5000	
2	1000			
3	1000			

location

Locid	loname

item

Itemno	Item name	stockqty	rate
1000	Tshirt	100	3456
2000	jeans		
1000	Tshirt		
1000	Tshirt		



one-Many
the key of one will be stored in many side table

Many-to Many
the key of both side will be stored in 3rd table

one to one
key of any one side will go to other side

Drid	Dname	Patientid	Pname	Appointmentdate	time	
100		1		10 apr	9.00am	

Movieid	Mname	bkdt	Showid	Showtime	Screenid	Seatno	Customerno	Cname	rate
123	x		M	8 am	1	1	123	fghfg	

bkdt+showid+screenid+seatno

3. Is the table in 1NF→ since every row and every column contains single value
   Yes
4. Is th table in 2NF
   - Find Primary key
     - bkdt+showid+screenid+seatno
   - find prime attribute
     - bkdt, showed, screened, seatno
   - find non prime attributes
     - moviid,mname,showtime, custno, cname, bkrate,
   - find functional dependency

   bkdt+showed+screenid---→ moviid,mname

   showid---→ show time

   bkdt+showid+screenid+seatno--→cusno,cname,bkrate

Showid	Showtime
M	8 am

Movieid	Mname	bkdt	Showid	Screenid
123	x		M	1

bkdt	Showid	Screenid	Seatno	Customerno	Cname	rate
	M	1	1	123	fghfg	

3NF

Transitive relationship should not be there

1. Is th table in 3NF

   Is it in 2NF

   Yes

   - Find Primary key
     - bkdt+showid+screenid+seatno
   - find prime attribute
     - bkdt, showed, screened, seatno
   - find non prime attributes
     - moviid,mname,showtime, custno, cname, bkrate,

   movieid-→mname

   custno->custname

Movieid	bkdt	Showid	Screenid
123		M	1

movie

Movieid	Mname
123	x

bkdt	Showid	Screenid	Seatno	Customerno	rate
	M	1	1	156	
	M	1	2	123	
	M	1	3	155	

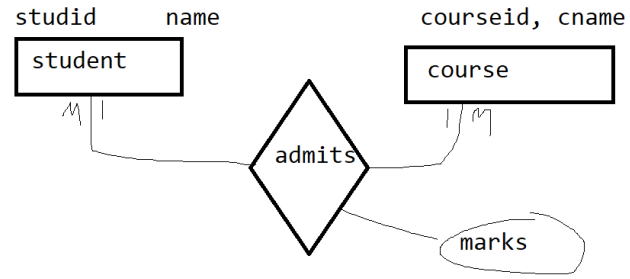
customer

Customerno	Cname
123	fghfg

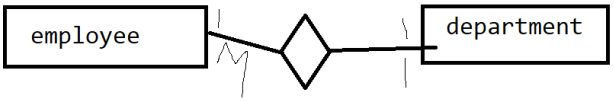
show

Showid	Showtime
M	8 am

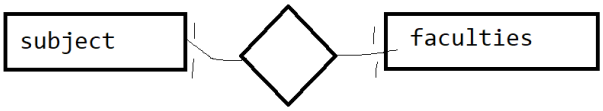
Rules for normalization



student(sid,sname)
course(cid,cname)
stud_course(sid,cid,marks)



dept(deptno,dname)
emp(empno,ename,sal,deptno)



faculty(fid,fname,subid)
subject(subid,sname)