To find constraints on the table

Show create table course;

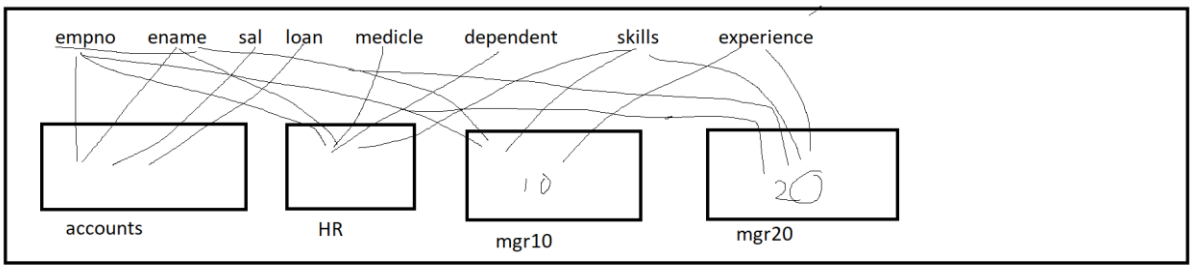
Views and indexes

Views

- It is a logical table
- It does not occupy any space
- It used to give restricted access to the existing table
- If a view is simple view(based on single table) and if view contains all not null columns from the base table then you can perform DML operations on the view
- But if view is complex view(based on more than one table), or it contains group by clause or having clause, or any complex union query, or joins, then view are read only view by default
- To avoid complexity of queries we can use views
  create view allemp
  -> as
  -> select * from emp_us
  -> union
  -> select * from emp_japan
  -> union
  -> select * from emp_india;

To find views

Show full tables in <database_name> where table_type like 'VIEW'



Create view mgr10

As(

Select empno,ename,skills,experience

From emp)

Select * from mgr10;

To put restrictions on vew

1. To create read only view (this does not work in mysql, but available in oracle)
   create view mgr20
   -> as

```
-> select *
-> from emp
-> where deptno=20
-> with read only;
```

In certain scenarios we need constant data then use materialized view(works in oracle but not in mysql)

```
Create materialized view item_inventory
As
Select *
From  stock_items
Where category='non perishable';
```

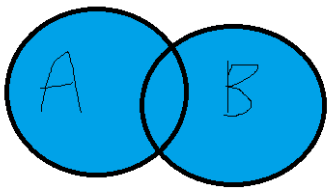
To drop view

```
 Drop view mgr10
```

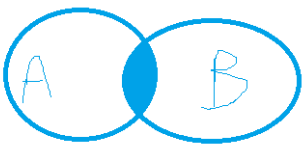
In mysql you may create temporary table, these tables will be available; only in current session

```
Create temporary table mytab1
(
Id int, name varchar(20));
)
```

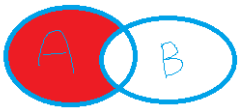
Set Opeartors (in mysql only union works)




```
select * from A
union
select * from B
```




```
select * from from A
intersect
select  * from B
```




```
select * from A
minus
select * from B
```

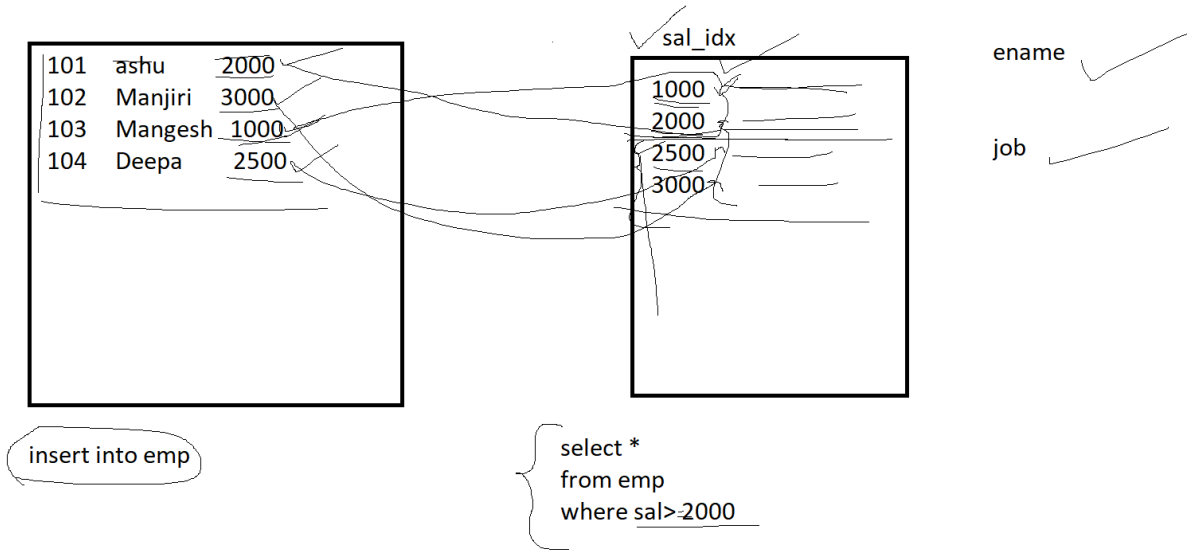
Indexes in MySQL

clustered index

- for one table there is always only one clustered index

non clustered index

- for one table there can be many non clustered indexes
- needs extra space to store index files



In a table, indexes get created automatically for primary key constraint and unique constraint

To create index

Create index sal_idx on emp(sal)

To drop index

Drop index sal_idx;

Full text index

When in query we use like or REGEXP then internally fulltext indexes can be used

Fulltext index while creating table

create table mytext(

-> col1 text,

Fulltext(col1));

To create fulltext indexes manually

Create fulltext index text_idx

On book(lesson)

To see the list of all indexes

To see the index for a specific table use SHOW INDEX:

```
SHOW INDEX FROM yourtable;
```

To view indexes on emp table

```
SHOW INDEX FROM emp
```

To see indexes for all tables within a specific schema you can use the STATISTICS table from INFORMATION_SCHEMA:

```
SELECT DISTINCT
   TABLE_NAME,
   INDEX_NAME
FROM INFORMATION_SCHEMA.STATISTICS
WHERE TABLE_SCHEMA = 'your_schema';
```

To see indexes for all tables within test database you can use the STATISTICS table from INFORMATION_SCHEMA:

```
SELECT DISTINCT
   TABLE_NAME,
   INDEX_NAME
FROM INFORMATION_SCHEMA.STATISTICS
WHERE TABLE_SCHEMA = test';
```

Grant and revoke

To assign permission

Syntax:

```
GRANT privileges_names ON object TO user;
```

Parameters Used:

- **privileges_name**: These are the access rights or privileges granted to the user.
- **object:**It is the name of the database object to which permissions are being granted. In the case of granting privileges on a table, this would be the table name.
- **user:**It is the name of the user to whom the privileges would be granted.

Privileges: The privileges that can be granted to the users are listed below along with the description:

To grant select and insert previledges on emp table to user1

```
GRANT SELECT,insert ON emp TO 'user1'@'localhost;
```

To grant select privilege on emp table to user1 with grant option

```
GRANT SELECT ON emp TO 'user1'@'localhost with grant option;
```

To remove select and insert previledges on emp table from user1

```
Revoke SELECT,insert ON emp from 'user1'@'localhost;
```

To assign all previleges

```
GRANT All ON emp TO 'user1'@'localhost;
```

To grant select permission to all users

```
GRANT SELECT,insert ON emp TO '*'@'localhost;
```

To grant permissions to functions and procedures

```
GRANT EXECUTE ON [ PROCEDURE | FUNCTION ] object TO user;
```

To assign execute permission to a function calculatesal

```
GRANT EXECUTE ON FUNCTION calculatesal TO user;
```

acno	custid	name	address	type	balance	branch
11	100	Kishori	Aundh	saving	3456	Aundh
12	100	Kishori	Baner	current	44444	Aundh
13	100	Kishori	Baner	demat	55555	Aundh
14	101	Rajan	Aundh	saving	6666	Aundh
null	103	Revati	Baner			

Then due to redundancy of data we are facing

1. updation anamoly: if we try to change address of customer for account 11 and forgot to change the address for account 12, 13 then it is updation anamoly

2. insertion anamoly:

   if any customer enquire about account and did not open the account then we will not be able to add new customer information for future reference, because acno cannot be kept null, is called as insertion anamoly.

3. Deletion anamoly

If rajan wants to close the account then we need to delete the row
But then we will loose Rajan’s information for further reference, it is called as deletion anamoly

To avoid these problems we will divide the tables into 2 tables

acno	custid	type	balance	branch
11	100	saving	3456	Aundh
12	100	current	44444	Aundh
13	100	demat	55555	Aundh
14	101	saving	6666	Aundh
null	103			

custid	name	address
100	Kishori	Aundh
101	Rajan	Aundh
103	Revati	Baner