1:30 reporting time 2-2:30 mcq 2:30 to 4 lab exam 4 to 4:15 explain (word file (prn_name_surname)):- prn name Q1 : ans (qurey) with ss of out put (face should be visible)

Day 12

20-10-24

TRIGGERS (Stored Object)

Table 1:

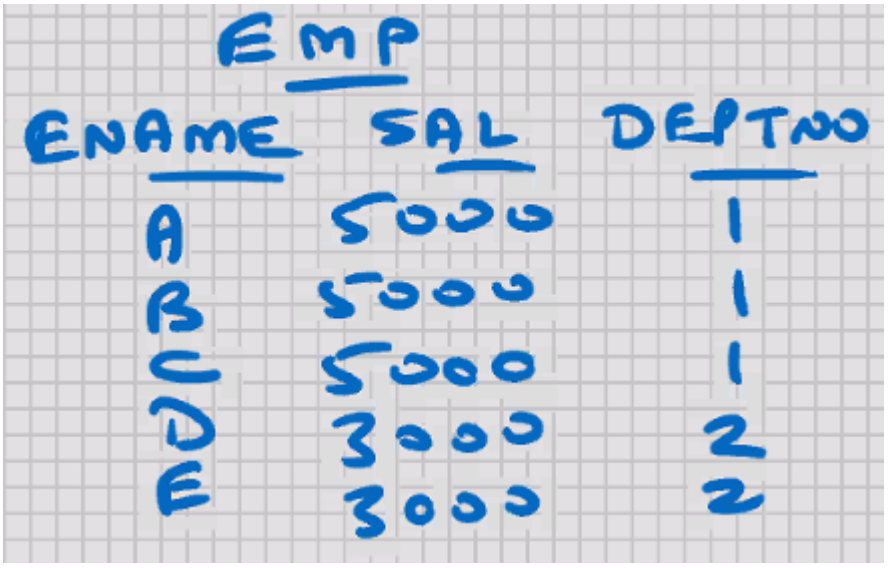
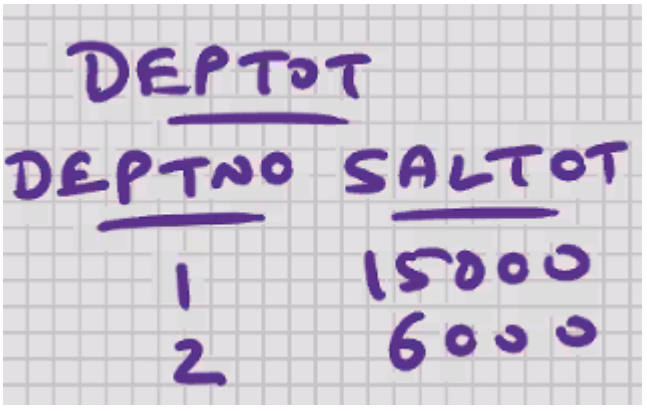
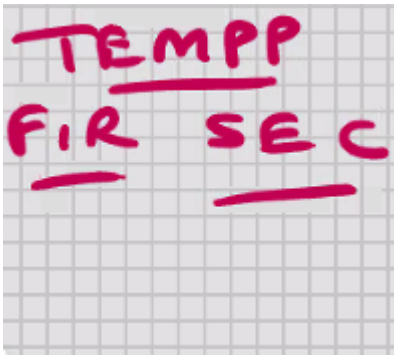


Table 2:





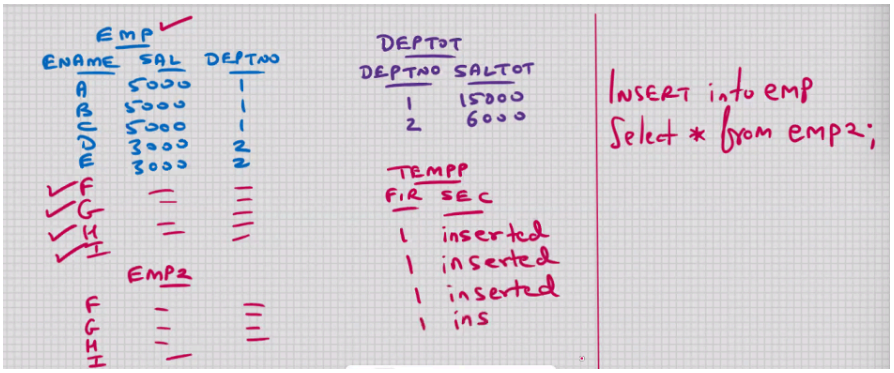
Output Table :

- Present in some of the RDBMS
- It is a rotine (set of commands) that gets executed automatically whenever some event takes place
- Triggers are written on tables
- Events are:
    - Before insert, after insert
    - Before Delete, after Delete
    - Before Update, after Update

```
delimiter //
create trigger abc  <--- creating trigger
before insert
on emp for each row
begin
    insert into tempp values(1,'Inserted');
end;//
delimiter ;
```

- Stored inb compiled fotmate
- Uses :
    - Maintain logs(Audit ttrails) of insertions
    - Automatic data duplication ,data mirroring , replication,mentain 2 or more copies of table in the event of insert
    - Maintain the shadow tables in the event of insert
    - Dynamic Default valeue Before insert
    - Data Cleansing BEFORE INSERT
    - Auto updation of relatable tables
    - AFTER DELETE is recommanded
    - maintain HISTORY table in the event of delete
    - ON delete cascad BEFORE DELETE
    - in above set child rows NULL before delete
    - to maintain log of updation (audit trails)
    - after update trigger is recommanded

    - we maintain history as well as shadow in th e event of update
- If insert operation on Table fails then it will cause the event to fail and the trigger changes are automatically rolled back
- if trigger fails then event fail and insert operation will auto rooled back
- our data will always be consistant
- After insert is recommanded
- WIthin the trigger , all MySQL statement allowed, e.g. variables , cursors ,if statements,loops, sub-blocks,etc.
- Wether you rollback or commit afterwards the data will always be consistant.
- Rollback and commit are not allowed inside the trigger
- Rollback or commit has to be specified AFTERWORDS, at the end of transaction



- in mysql all triggers are at row level(will fire once for each row)
- in my sql we can have maximum 6 triggers per table

```
delimiter //
create trigger abc  <--- creating trigger
before insert
on emp for each row
begin
    insert into tempp values(new.sal,new.ename);
end;//
delimiter ;
```

- new.emp ??

```
create trigger abc
before insert
on emp for each row
begin
        if new.deptno = 1 then
                set new.sal = 5000;
        elseif new.deptno = 2 then
                set new.sal = 3000;
        else
                set new.sal = 2500;
        end if;
end; //
delimiter ;
```

```
delimeter //
create trigger abc
before insert

on emp for each row
begin
    update deptot set saltot = saltot+new.sal where deptno =n
end; //
delimiter ;
```

show triggers;

show triggers from [db_name] show triggers from cdacmumbai;

drop triggers abc

- if you drop the tabble, then the indexes and triggers are dropped automatically

```
select * from information_schema.triggers where trigger_sche
```

- We can Stored procedure and stored functions inside the trigger

Delete Operation :

```
delimiter //
create trigger pqr
before delete
on emp for each row
begin
    insert into tempp values(1,'Deleted',user(),now());
end; //
delimiter ;
```

```
delimiter //
create trigger pqr
before delete
on emp for each row
begin
    insert into tempp values(old.sal, old.deptno);
end; //
delimiter ;
```

Update Delete :

```
delimiter //
create trigger pqr
before delete
on emp for each row
begin
    update deptot set saltot = saltot-old.sal
    where deptno = old.deptno;
end; //
delimiter ;
```

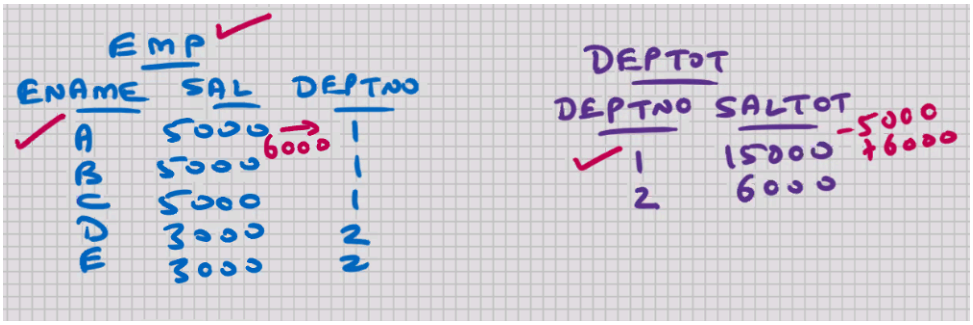
- all thiggers are at server level
- you can perform the DML operatin using mySql command line client or mysql workbench or java or MS

```
delimiter //
create ttrigger xyz
before update
on emp for each row
begin
    insert into tempp values(1,'updated ');
end; //
delimiter ;
```

```
delimiter //
create trigger xyz
before update
on emp for each row
begin
        insert into tempp values(old.sal,old.ename);
        insert into tempp values(new.sal,new.ename);
end; //
delimiter ;
*       old.ename, old.sal, old.deptno, new.ename, new.sal, new.deptno
        are MySQL created variables
```

Cascading trigger

- one trigger causes to second trigger to execute
- what is ment by mutating tables ?(in case of recursion)
- --> if some cascading trigger causes one of the privious triggers to execute , then it will not go into infinite loop; you will get an error that the table is undergoing mutation and the entire transaction is automatically rolled back



-->

```
delimiter //
create trigger xyz
before update
on emp for each row
begin                                   (1500)    (5000)    (6000)
        update deptot set saltot = saltot - old.sal + new.sal
        where deptno = old.deptno;
end; //                  (1)
delimiter ;
```

for:

```
update emp
set deptno = 2
where ename = 'A';
```

```
delimiter //
create trigger xyz
before update
on emp for each row
begin
        if old.sal <> new.sal or old.deptno <> new.deptno then
                if old.deptno <> new.deptno then
                     update deptot set satot = saltot - old.sal
                     where deptno = old.deptno;
                     update deptot set satot = saltot + new.sal
                     where deptno = new.deptno;
                else
                     /* if you are UPDATING THE SAL COLUMN ONLY */
                     update deptot set saltot = saltot - old.sal + new.sal
                     where deptno = old.deptno;
                end if;
        end if;
end; //
delimiter ;
```

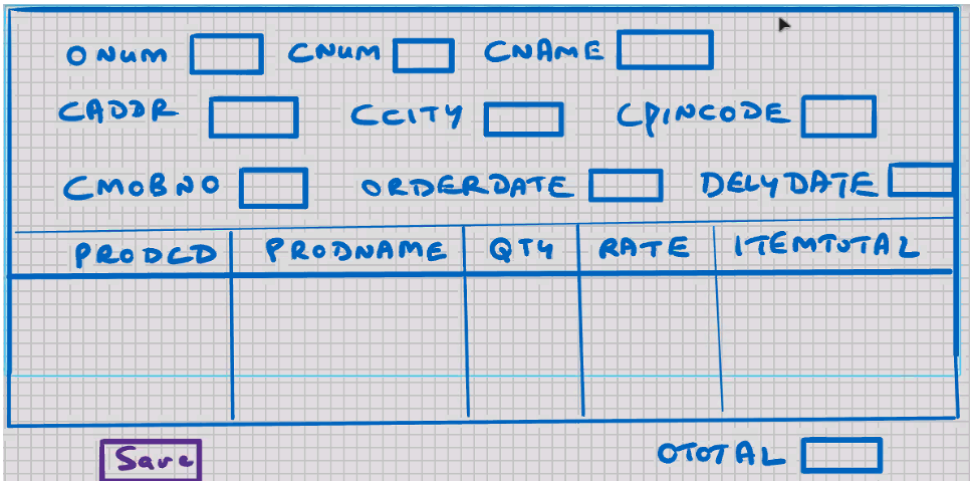
NORMALIZATION

- Applicable for RDBMS only
- coz it is concept of table design
- Upto 9th normal form in ORDBMS
- Upto 4th normal form in RDBMS
- What tables to create , Structure ,columns,datatypes,widths,Constrints
- Based on User requirments
- part of design phase (min1/6)
- Aim of normalization is to have efficiant structure
- Aim of normalization is to avoid the redundancy (avoid the duplication)
- Secondary aim of Normalisation is to reduce the problem of insert, update and delete
- Normalization is done from an input prespective
- Normalization is done from a Forms prespective

1. VIEW THE ENTIRE APPLICATIONS ON A PER-TRANSACTION BASIS , AND YOU NORMALISE EACH TRANSACTION SEPARATELY eg. Customer places an order, customer cancels the order

NORMALISATION STEPS

(ONLINE SHOPPING EXAMPLE : CUSTOMER PLACES AN ORDER)



1. For a given transactions make a list of fields
2. Ask client some sample data
3. With the permision and involvement of client , strive for atomicity
    1. Column is divided into sub columns and sub columns are divided into sub -sub-columns (Like address)
4. For every column make a list of column properties (like PINCODE,DATE)
5. Get the Client Sign-off
6. End of Client Involvement
7. Assign the Datatype for each column
8. Assign width for each column
9. Assign not-null,Unique& check constrints
10. For all practical purposes you can have a single table with all these columns
11. Remove computed columns
12. Key element Will be primary key of this table

AT this point data is in Un-normalised form

STEP 1

- It's a term: single row block / multi row block
- The group of column thayt constitute your multi row block are repeting group

1. remove Reapeting Columns remove and create new

Onum	
Cnum	Prodcd
Cname	Prodname
Caddr	Qty
Ccity	Rate
Cpincode	
Cmobno	
Orderdate	
Delydate	

STEP 2

- key element will be primary key of new table

STEP 3

Onum	Onum
Cnum	Prodcd
Cname	Prodname
Caddr	Qty
Ccity	Rate
Cpincode	
Cmobno	
Orderdate	
Delydate	

(This step may or may not be required)

- Add the primary key of the original table to new table to give you a composite primary key

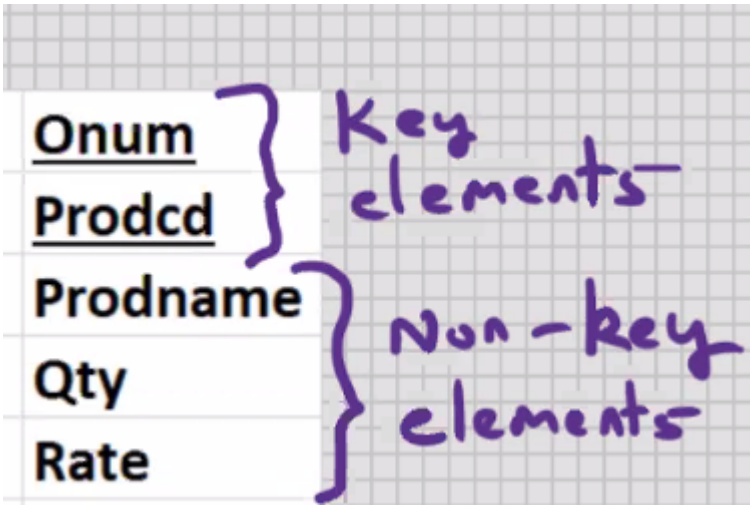
The above are to be repated again and again infinitely till you can not normalise any further FIRST NORMAL FORM (FNF/1NF/Single Normal Form)

- All repating groups are removed from table design
- 1: many is always encountered here
- DEPT and EMP tables are in First normal form

25%

STEP 4

- Only the tables with composite primary key are examined



STEP 5

- Those non key elements that are not dependent on the entire composite primary key they are to be removed into a new table

Onum
Cnum
Cname
Caddr
Ccity
Cpincode
Cmobno
Orderdate
Delydate

Onum	
Prodcd	Prodname
Qty	Rate

STEP 6

Onum						Onum			Prodcd
Cnum						Prodcd ✓			Prodname
Cname						Qty			Rate
Caddr									
Ccity									
Cpincode									
Cmobno									
Orderdate									
Delydate									

- on which they are dependent in privious table bring that column in new table and make that primary key

(above 3 steps are to be repeted infinitly till you can not normalize any furthor)

SECOND NORMAL FORM(SNF)/2NF

Every column is functionly dependant on primary key FUNCTIONAL DEPENDANCY -> without primary key , that column cannot function

67%

STEP 7

- only the non key elements are examined for inter-dependent

STEP 8

- Inter - dependent columns are removed into a new table

Onum			Cnum			Onum			Prodcd
Orderdate			Cname			Prodcd			Prodname
Delydate			Caddr			Qty			Rate
			Ccity						
			Cpincode						
			Cmobno						

STEP 9

- key element will be the primary key of the new table , and the primary key of new table , that column it is to be retained in the original table for relationship purposes

Onum	Cnum	Onum	Prodcd
Orderdate	Cname	Prodcd	Prodname
Delydate	Caddr	Qty	Rate
Cnum	Ccity		
	Cpincode		
	Cmobno		

Onum	Cnum	Onum	Prodcd
Orderdate	Cname	Prodcd	Prodname
Delydate	Caddr	Qty	Rate
Cnum	Ccity		
	Cpincode		
	Cmobno		

(above 3 steps are to be repeted infinitly till you can not normalize any furthor)

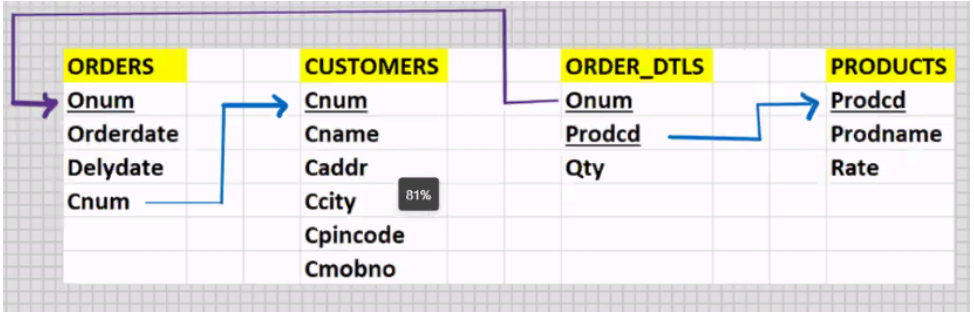
THIRD NORMAL FORM(SNF)/3NF

- Transitive dependancies are removed from table design

Final Output:

ORDERS	CUSTOMERS	ORDER_DTLS	PRODUCTS
Onum	Cnum	Onum	Prodcd
Orderdate	Cname	Prodcd	Prodname
Delydate	Caddr	Qty	Rate
Cnum	Ccity		
	Cpincode		
	Cmobno		

Foregin Key :



- Post Normalisation

```
Post-Normalisation
*       implement Extension columns
*       reserve some columns for logs of DML operations
```