DML operation

TCL ---- transaction control language

commit	It makes the changes permanent
rollback	It undo the changes
savepoint	It puts some marker

to turn off autocommit

set autocommit=0

to turn on autocommit

set autocommit=1

if autocommit is off

10 records are there in mytable

drop table dept; -----this is autocommit

rollback;

if we have mytable which has 5 records

commit;

insert -3

insert-4

savepoint A

delete 1

update 1

insert-3

savepoint B

insert 2

delete 1

rollback to A

window functions

Use of window function

1. To find top n values from the data
2. In aggregate function, If we want to display column which is not given in group by clause.
   Example :
   following query will give error, But it can be done in window functions

   select deptno,ename,sum(sal)

From emp
Group by deptno

All aggregate functions are window function and following are also window functions

row_number()	assign unique value to every row within window
rank()	assign number to distinct values, if the values are same then same rank will be assigned to both rows. but when more than one row gets same rank, then it will skip numbers and then the next rank will be assigned
dense_rank()	assign number to distinct values, if the values are same then same dense rank will be assigned to both rows. but when more than one row gets same dense rank, then it will not skip in between numbers and then the next rank will be assigned
lag(val, n)	it will find nth previous value
lead(val,n)	it will find nth next value
first_value(sal)	it will find first value of the given column within window
last_value(sal)	it will find last value of the given column within window for last value we need to add the extra clause ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING

1. In the following example partition by will divide the data into window and order by will arrange data within window

```
select deptno,ename,sum(sal) over (partition by deptno)

from emp


select empno,ename,sal,deptno,row_number() over (),rank() over (order by sal desc)

from emp;
```

1. to find highly paid employee

```
select *  from (

select empno,ename,sal,deptno,row_number() over () rownum,rank() over (order by sal desc) rn,dense_rank() over (order by sal desc) drn from emp) e

where e.drn=1;
```

2. to find highly paid employee in each department

```
select *  from (
select empno,ename,sal,deptno,row_number() over () rownum,rank() over (order by sal desc) rn,dense_rank() over (partition by deptno order by sal desc) drn from emp) e
where e.drn=1;
```

3. to find first sal in each window
   select empno,ename,sal,deptno,first_value(sal) over (partition by deptno order by sal) fv
   from emp;

1. find n topmost highly paid employees

```
select empno,ename,sal,rk,drk
from (select empno,deptno,ename,sal,
row_number() over (order by sal desc) rownum,
rank() over (order by sal desc) rk,
dense_rank() over (order by sal desc) drk
from emp) e

where e.drk<=3;
```

2. to find 3 topmost for every department

```
select *
from (select empno,ename,sal,deptno,
    row_number() over (partition by deptno order by sal desc) rnum,
          rank() over (partition by deptno order by sal desc) rk,
          dense_rank() over(partition by deptno order by sal desc) drk
     from emp) e
where e.drk<=3
```

3. to find topmost sal for each job

```
select *
from (select empno,ename,sal,job,
     rank() over (partition by job order by sal desc) rk
          from emp) e
where e.rk=1
```

4. to find difference between my sal, and
previous sal of my department

```
select empno,ename,sal,lagsal,sal-lagsal difference
from (select empno,ename,sal,deptno,
lag(sal,1) over (partition by deptno order by sal) lagsal
from emp) e
```

4. to find difference between my sal, and
next sal of my department

```
select empno,ename,sal,leadsal,leadsal-sal difference
from (select empno,ename,sal,deptno,
lead(sal,1) over (partition by deptno order by sal) leadsal
 from emp) e;
```

5. find difference between my sal and minimum sal of my department

```
select empno,ename,deptno,sal,fv,sal-fv difference
from (select empno,ename,deptno,sal,
first_value(sal) over (partition by deptno order by sal) fv
from emp) e
```

6. find the difference between my sal and highest salary
in my department

```
select empno,ename,deptno,sal,lv, lv-sal differnece
from (select empno,ename,deptno,sal,
last_value(sal) over
(partition by deptno order by sal) lv
ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) lv
from emp) e
```

7. find cumulative sum of sal in the table

select empno,ename,sal,sum(sal) over (order by empno)

from emp;

indexes

Indexes store key and the position of the key in index file.

2 types of indexes

1. clustered index
    a. there will be only one clustered index
    b. it does not require extra space because it is stored along with data in the table
2. non clustered index
    a. there can be many non clustered index
    b. these are stored outside table, and hence need extra space

indexes speed up the search action , but reduces the speed of DML statements, so donot create unnecessary indexes.

Types of indexes

1. primary key index-----indexes on primary key gets create automatically
2. unique index
    a. indexes on column with unique constraint gets created automatically
    b. the fields on which unique index is created, then duplicate values are not allowed in that column.

    create unique index idx_passport

    on emp(passport desc)

3. regular index

    create index idx_sal

    on emp(sal desc)

4. full text indexes
    a. These are usually used on text type column,
    b. it stores phrases or words and their position

    create full text index idx_profile

    on emp(profile)

5. geospatial index – It is used when the field store geographical location.

to drop the index

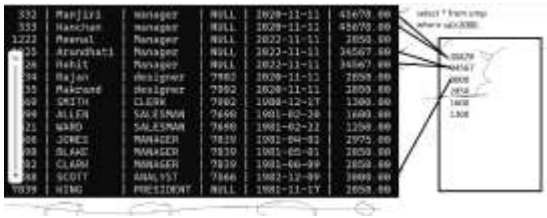
drop index idx_passport on emp

show indexes from emp;

To find, query uses which index

Explain select * from emp where sal>2000;

Salary index



Views in mysql

There are 2 types views in database

1. view---→ simple views or complex views
2. materialized view---- when you are working on static data, then it is good to create materialized view.
   Once you fire the query 1 st time data will be retrieved, it will get save in RAM for the current session.
   It speeds up the job of retrieval of the data.

- for every view, separate table will not get created, only base query gets stored for view.
- if we fire select statement on view, then the base query will get executed
- while creating view, if we use with check option, if view based on single table, and if view contains all not null columns, then one can perform DML operations on the view. and only valid data for which where condition is satisfied, can be added or removed or updated
- To stop all DML operations on the view, use with read only option

why to use views

1. to give limited access to the table
2. hide complex queries under the view
3. we may hide table names, which increases security.

to create view

create <materialized> view < name of the view>

as

<base query>

to create a view for manager of dept 10 to give access to all the records of dept 10

create view mgr10

as

(select * from emp

where deptno=10

with check otion)

```
create view mgr10
as
(select * from emp
where deptno=10
with read only option)
```

```
create view testview1
   -> as
   -> select empno,ename,e.deptno edeptno,d.deptno ddeptno,d.dname
   -> from emp e, dept d
   -> where e.deptno=d.deptno;
```

```
create view testview
   -> as
   -> select deptno,job,sum(sal+ifnull(comm,0)) sumsal,count(*) cnt
   -> from emp
   -> group by deptno;
```

DCL ----Data control language

grant	It is used for assigning the permission
revoke	It is used for revoking the permission

to assign permission to all database, all table, all previledges

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

to assign inser, create, select permissions on test database emp table to newuser

```
Grant select,create,insert on test.emp to 'newuser'@'localhost'
```

to make the changed permission permanent

1. FLUSH PRIVILEGES;

How To Grant Different User Permissions

Here is a short list of other common possible permissions that users can enjoy.

- ALL PRIVILEGES- as we saw previously, this would allow a MySQL user full access to a designated database (or if no database is selected, global access across the system)
- 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 provide a specific user with a permission, you can use this framework:

- GRANT type_of_permission ON database_name.table_name TO 'username'@'localhost';
- Grant select,insert on 'mydb.dept' to 'user2'@'localhost'
- Grant select,insert on 'mydb.dept' to 'user3'@'localhost' with grant option

If you need to revoke a permission, the structure is almost identical to granting it:

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

Revoke insert on test.dept from 'username'@'localhost';

Note that when revoking permissions, the syntax requires that you use FROM, instead of TO as we used when granting permissions.

You can review a user's current permissions by running the following:

- SHOW GRANTS FOR 'username'@'localhost';

PLSQL (Procedural Language Structured Query Language)

Why we use PL SQL

1. we can hide table names from the developer of the middleware application, which increases the security of the database.
2. For a particular task, if we need to execute many queries, then we may wrap these queries in a procedure, and call the procedure from middleware application, once,

execute all the queries, complete the task and go back, this will reduce the network traffic, also improves performance efficiency of the middleware application. so it reduces the interaction between middleware program and database.

3. If any of the query is complex, the we may hide the query inside the procedure
4. Procedures will also reduce the network traffic.

in plsql we can write 3 types of blocks

procedure	If you want to write business logic and do not want to use return statement, then use procedure
function	If you want to return single value as output, then use functions
triggers	if you want to write procedures which gets executed automatically, is called as triggers

in procedure we can pass 3 types of parameters

in	these types of parameters are used for passing the value as i/p these are readonly parameters, its value cannot be changed inside the procedure this is default type of parameter
out	these types of parameters are used for getting the output these are writeonly parameters, its value can be assigned or changed inside the procedure
inout	these types of parameters are used for passing input as well as getting the output these are read and write parameters, using these parameters we may pass the value and we may change its value

While writing procedures or function we need to change the delimiter

delimiter //

1. to write procedure to insert record in department table

```
delimiter //
create procedure insertdept(in dno int,in dnm varchar(20),
in dloc varchar(20))
begin
  insert into dept values(dno,dnm,dloc);
end//
delimiter ;
```

to check whether procedure works correctly or not

```
mysql> call insertdept(200,'admin','pune');
```

2. to get number of employees in each department

delimiter //

```
create procedure getdata(in dno int,out cnt int,
out minsal float(9,2), out maxsal float(9,2))
begin
  select count(*),min(sal),max(sal) into cnt,minsal,maxsal
  from emp
  where deptno=dno;
  #to print all values
  -- this is comment
  select cnt,minsal,maxsal;
end//
delimiter ;
```

to check whether procedure works correctly or not

```
mysql> call getdata (10,@cnt,@min,@max);
```

to check values @cnt,@min,@max

3. increment cnt by val

```
delimiter //
create procedure incrementcnt(inout cnt int,in val int)
begin
  set cnt=cnt+val;

end//
delimiter ;
```

to check whether procedure works correctly or not

```
mysql> set @c=5
call incrementcnt(@c,12)
select @c;
```

4. write a procedure to find sal+comm of a employee whose empno is given

```
delimiter //
create procedure getadetails(in eno int,out netsal float(9,2))
begin
  select sal+ifnull(comm,0) into netsal
  from emp
  where empno=eno;

end//
delimiter ;
```

to check whether procedure works correctly or not

```
call getadetails(7902,@s)
select @s;
```

in above example, select ... into statement can be used only inside pl sql blocks, the select query should return single row as output. number of column names before into and number of variables after into should be same.
@s is session variables. these variables will remain

Syntax for if—else

IF `expression` THEN
  `statements;`
ELSE
  else-`statements;`
END IF;
	Using If ----else--------
	
	IF expression THEN
	
	  statements;
	
	ELSEIF elseif-expression THEN
	
	  elseif-statements;
	
	...
	
	ELSE
	
	  else-statements;
	
	END IF;

if comm is null or 0 “poor performance”
 comm <=300 “ok performance”
comm >300 and <= 500 “good performance”
otherwise “excellent performance”

```
delimiter //
create procedure getfeedback(in eno int,out response varchar(20))
begin
```

```
  declare vcomm float(9,2);
  select comm into vcomm
  from emp
  where empno=eno;
  if vcomm is null or vcomm=0 then
     set response='poor performance';
  elseif vcomm<=300 then
     set response='ok performance';
  elseif vcomm<=500 then
            set response='good performance';
  else
    set response='excellent performance';
  end if;
  select vcomm,response;
end//
delimiter ;
```

write a procedure using product table pass pid as parameter
if the price of product <100 cheap product
else if the price is in range 100 to 500 moderate price
else if it is in range 500 to 1500 ok price
else expensive

```
delimiter //
create procedure getProuctRemark(pno int,out remark varchar(30))
begin
    declare vprice double(9,2);
    select price into vprice
    from product
    where pid=pno;
    if vprice <100 then
      set remark='cheap product';
    elseif vprice<500 then
      set remark='moderate product';
    elseif vprice<1500 then
      set remark='ok product';
     else
       set remark='expenssive product';
     end if;
     select pno,remark;
end//
```

loops in plsql

while expression do
statements;
end while;
	It is a top tested loop, statements inside loops will get executed till the condition is true.

repeat statements until expression end repeat	It is a bottom tested loop, and statements inside this loop will get executed until the condition is false.

label1: loop
if condition then
leave label1
end if;
end loop;
	This is infinite loop, and will continue execution till leave statement gets executed, leave statement is same as break statement, it stops the loop forcefully
	
	In the loop you may use iterate statement, it is similar to continue statement.
	It will transfer the control at the beginning of the loop

example while loop

```
delimiter //
create procedure displaywhile(in num int)
begin
  declare str varchar(200) default '';
  declare n int default 1;
  while n<=num do
   set str=concat(str,n,',');   #1,2,3,4,5,
          set n=n+1;
  end while;
  #remove last , from str
  set str=substr(str,1,length(str)-1);
  select str;
end//
delimiter ;
```

repeat----until example

```
delimiter //
mysql> create procedure repeatdemo(in cnt int)
   -> begin
   ->  declare str varchar(100) default '';
   ->  declare n int default 1;
   ->  repeat
   ->    set str=concat(str,n,',');
   ->    set n=n+1;
   ->  until n>cnt
   ->  end repeat;
   -> set str=substr(str,1,length(str)-1);
   -> select str;
  -> end//
```

loop-endloop example

```
delimiter //
create procedure loopdemo(in num int)
begin
 declare cnt int default 1;
 declare str varchar(100) default '';
```

```
        label1: loop
          set str=concat(str,cnt,',');
          set cnt=cnt+1;
          if cnt>num then
            leave label1;
          end if;

          end loop;
          set str=left(str,length(str)-1);
          select str;
        end//
delimiter ;
```

In mysql the select statement which returns multiple rows is allowed to be written inside the procedure, but in other databases like oracle, allows only select....into statement inside the procedure
hence if you need multiple rows from a table inside procedure, then we use cursor.

step by step procedure to use cursor

1. declare the cursor
2. declare continue handler to stop the loop
3. open cursor -→ the data will be populated in the cursor
4. fetch the next row in the cursor
5. check if it is last row, then stop the loop and goto step 8
6. process the row
7. repeat steps 4 to 6 till the data is available in the cursor
8. close cursor

```
delimiter //
create procedure displayallemp()
begin
  declare vname,vjob varchar(20);
        declare v_finished,vempno int default 0;
        declare vsal float(9,2);
  declare empcur cursor for select empno,ename,job,sal from emp;
  declare continue handler for NOT FOUND set v_finished=1;
  open empcur;
  label1: loop
    fetch empcur into vempno,vname,vjob,vsal;
          if v_finished=1 then
                  leave label1;
          end if;
          select vempno,vname,vjob,vsal;
   end loop;
        close empcur;

end//
```

```
delimiter ;
```

```
create procedure updateempsal()
  -> begin
  ->   declare vname,vjob varchar(20);
  -> declare v_finished,vempno int default 0;
  -> declare vsal,vnewsal float(9,2);
  ->   declare empcur cursor for select empno,ename,job,sal from emp;
  ->   declare continue handler for NOT FOUND set v_finished=1;
  ->   open empcur;
  ->   label1: loop
  ->     fetch empcur into vempno,vname,vjob,vsal;
  ->   if v_finished=1 then
  -> leave label1;
  ->   end if;
  ->   #select vempno,vname,vjob,vsal;
  ->   if vjob='Manager' then
  ->     set vnewsal=vsal*1.10;
  ->
  ->   elseif vjob='Analyst' then
  ->     set vnewsal=vsal*1.12;
  ->
  ->   elseif vjob='Clerk' then
  ->     set vnewsal=vsal*1.15;
  ->
  -> else
  ->   set vnewsal=vsal*1.08;
  ->
  ->      end if;
  ->  update emp
  ->  set sal=vnewsal
  ->  where empno=vempno;
  -> select vempno,vname,vsal,vjob,vnewsal;
  ->    end loop;
  -> close empcur;
  -> end//
```