**Day 4 : DBT**

* In a dbms, rows are stored in a file
* In a file, the row are stored sequentially
* In a DBMS, there is concept of row number
* In RDBMS, Table is not a File, every Row is file
* In RDBMS, the rows of a table are scattered(fragmented) all over the DB Server HD
* The Reason why RDBMS does this is tp speed up the INSERT statement
* In a multi-user environment, if multiple users are inserting rows

Into the same table at the same time; if the table were to store the rows sequentially, it would be very slow

* When HD it will store the row there
* When you SELECT from a table, the order of rows in the output depends on the row address; it will always be in ascending order of row address.
* Once you INSERT a row, if the row length is increasing, and if the free HD pace is not available, then MYSQL will move the entire row to some other address.
* When you UPATE a row, the row address MAY change.
* Later when you SELECT from that table, you may see that row at some other position
* Hence is is not possible to see the first ‘N’ rows if a table or the last ‘N’ rows of a table.
* **MySQL = SQL – ORDER BY clause**
* **Used for sorting**
* **Uses :-**

1. **For presentation purposes (for reporting purposes)**

**Select deptno , job, ename, sal, hiredate from emp;**

* **Select deptno , job, ename, sal, hiredate from emp Order by ename;**
* asc -> by default
* desc
* **Select deptno , job, ename, sal, hiredate from emp Order by deptno;**
* **Select deptno , job, ename, sal, hiredate from emp Order by order by hiredate;**
* **Business Intelligenc (BT), Dtata Science, Dtata Analytics**

**Select deptno,job, ename,sal, hiredate from emp Order by deptno desc, job;**

**C Order by deptno desc, desc job;**

**\*no upper limit on the number of columns in ORDER BY Caluse**

**Select ……………………………………………….**

Ordr by country, state, city

\*if you have large number of rows in the table, and large number of colmns in ORDER BY clause, then the SELECT statement will be very slow, because that musch sorting has to take place in Server RSM.

**Select deptno,job, ename, sal, hiredate from emp where deptno = 10 order by ename;**

**\***WHERE clause is specified BEFORE the ORDER BY clause

\*ORDER BY clause is LAST clause in SELECT statement

* **Select ename, sal\*12 fom emp;**
* **Select ename, sal\*12 annual from emp order by annual;**
* **Select ename, sal\*12 annual from emp order by annual;**
* **Select ename, sal\*12 “Annual” from emp order by “Annual”;**
* **Select ename, sal\*12 “Annual” from emp order by 2;**

**Select \* from emp order by 2;**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EMPNO** | **ENAME** | **EMPSAL** | **CITY** | **DEPTNO** |
| **1** | **ADAMS** | **1000** | **Mumbai** | **10** |
| **2** | **BLAKE** | **2000** | **Delhi** | **10** |
| **3** | **ALLEN** | **2500** | **Mumbai** | **20** |
| **4** | **King** | **3000** | **Delhi** | **30** |
| **5** | **FORD** | **4000** | **Mumbai** | **40** |

* **Select \* from empwhere ename > ‘A’ and ename < ‘B’;**
* **Blank-padded comparision semantics:-**

Whenever you compare 2 strings of different lengths, the shorter of the 2 strings is temporarily padded with blank spaces on RHS such that their length become equal; then it will start the comparision character by character based on ASCII value.

* **Special Operators(Like)**

Wildcards (used for pattern matching) % any character and any number of characters.

* Select \* from emp where ename like ‘A%’;
* Select \* from emp where ename not like ‘A%’;
* Select \* from emp where ename like ‘A%S’;
* Select \* from emp where ename like ‘--A%’;
* Select \* from emp where ename like ‘----‘; 🡪 4 letters word
* Select \* from emp where ename like ‘-I - -‘;
* Select \* from emp where sal between 2000 and 3000; 🡨 inclusive
* - - - - -

Select \* from emp where sal not between 2000 and 3000; 🡨exclusive

**MYSQL- Special Operators (Between,In)**

Select \* from emp where sal between ‘A’ and ‘B’;

Select \* from emp >=’A’ andename <= ‘B’;

Select \* from emp where deptno =10 or deptno = 20 or deptno = 40;

Select \* from emp where deptno i(10,20,40);

DML 🡪 Create,Drop

DML 🡪 Insert,Update

DQL 🡪 Select

UPDATE

- - - - -- -

* Update emp set sal = 10000 where empno = 1;
* Update emp set sal = sal + sal\*0.4 where empno = 1;

Transaction Processing

* Commit will save all the DML changes since the last committed state.
* When the user issues a Commit, it is known as END of Transaction.
* Commit will make the Transaction permanent

Total Work Done = t1 + T2 +… .+Tn;

* Total work done = sum of individual work done inall transactions

Mysql>commit work;

OR

Mysql>commit;

* Work is Optional in MYSQL
* Work is ANSI SQL
* When to issue the commit is decided by the end user (Client), and it depends upon the logical scope of work.
* Only the DML commands are affected by Rollback and Coo=mmit
* Any DDL command ,it automatically commits; not nly will it commit itseif, it will commit everything before it.
* When you exit from SQL,the system automatically commits.
* Any kind of power failure, network failure,system failure, pc reboot, window.
* Savepoint is a point within your work
* Savepoint is similar to a Bookmark or aMIlestone
* You can Rollback to a Savepoint
* You Cannot commit to A Savepoint
* Commit will save all the DML changes ince the last committed state.

**Day 4**

**Read and write Consistency**

* When you SELECT from a table, you can view ONLY the committed data of all other users plus changes made by you.
* When you UPdatE or DELETE a row, thw is automatically locked for other users.
* Row LOCKING IN MYSQL IS AUTOMATIC.
* When you UPDATE or DELETE a row, that row becomes READ\_ONLY for others users.
* Other users can select from that table;they will view the old data before your changes.
* Other users can INSERT rows into the table.
* Other users can UPDATE or DELETE “other” rows from that table
* No other user can UPDATE or DELETE your locked row till you have issued a Rollback or commit.
* Locks are automatically released when you rollback or commit.
* OPTIMISTICLOCKING 🡪 Automatic locking mechanism in MySQL
* PESSIMISTIC LOCKING 🡪 Manually lock the rows in advance BEFORE issuing UPDATE or DELETE.
* To lock the rows manually, you have to use SELECT statement with a FOR UPDATE clause.
* Select \* from emp where deptno = 10 for update; 🡪 if row is available, the lock it;els wait in theRequest queue.
* Select \* from emp where deptno = 10 for update nowait; 🡪 if row is available, then lock it; els give an error message and abort the operation
* MySQL – SQL – Functions

**Concatenate**

Select concat (fname,lname) from emp;

Output

Concat(fname,lname)

* - - - -

ArunPurun

TarunArun

SirunKirun

NutanPurun

* Select concat(concat(fname, ‘ ‘),lname) from emp;

Uses:- a for presentation/reporting purposes

* Max upto 255 levels for function within function (this limit of sql can be exceeded with the help of views)
* Select lower(fname) from emp;

Arun

Tarun

Sirun

Nutan

* Uses :- For presentation/reporting purposes

Update emp set fname = lower(fname) where ……..;

- - - - - - - - - - - - - - - - - - - - - -

In Other RDBMS :-

Not available in MySQL :-

Select initcap(fname) from emp;

Arun

Tarun

Sirun

* **Select lpad(ename,25,’ ‘ ) from emp; 🡪 it put right spaces**

Uses :-

1. Right justification
2. Cheque printing
3. Masking of Account/Card/Aadhar No

* **Select rpad(ename,25,’ ‘)from emp;**
* **Select rpad(ename,25,’ \*‘)from emp;**

Uses:-

1. Left justification of numeric data.
2. Convert varchar to char (Variable length to fixed-length)
3. Cheque printing

Lpad(rpad(…..)….)

1. Masking of Account/Card/Adhar No
2. Center-justification

Lpad(rpad(….)……..)

**Select ltrim(ename) from emp;**

Arun Purun

Tarun Arun

Sirun Kirun

Nutan Ourun

Uses:-

1. Left justification

Rpad(ltrim(………)…………);

**Select rtrim(ename)from emp;**

Uses:-

1. Convert char to varchar (convert fixed-length to variable length)
2. Right justification of char dtata

Lpad(rtrim(…….)……….)

**Select trim(ename) from emp;**

* Removes blank spaces from both the sides in MySQL

**Select substr(ename,3)from emp;**

**3** 🡪 Starting Position

Un Purun

Run Arun

Run Arun

Run Kirun

Tan Purun

**Select substr(ename,3)from emp;**

**Select ubstr(ename,3,2) from emp;**

**Select substr(ename,-3)from emp;**

**Select substr(ename,-3,2) from emp;**

ru

ru

ru

ru

**substr(‘Kharghar’,2) 🡪 harghar**

**substr(‘Kharghar’,5) 🡪ghar**

**substr(‘Kharghar’,5,3)🡪 gha**

**substr(‘Kharghar’,1,4)🡪 khar**

* Uses:-

1. Used to extract a part of the string
2. Substr(‘rajesh’1,3)-> raj
3. **select replace(ename,’un’,’xy’) from emp**;

**select replace(ename,’un’,’xy’)from emp;**

**select replace(ename,’un’,’xyz’)from emp;**

**select replace(ename,’un’,’x’from emp;**

**select replace(ename,’un’,’ ‘)from emp;**

un 🡪

Ar pur

Tar Ar

Sir Kir

Nutan Pur

Uses :

1. encoding/ Decoding
2. Spell out the number and vice versa
3. Encryption/Decryption
4. Etc.

Select insert(ename,’un’) from emp; 🡨 returns starting posn of string(frist occurrence)

3

4

4

10

0

* If string is not, then it returns 0

Uses:-

a.Check if one String exists the in another string.

Character function

Emp

Ename

AunBannerjee

Charlie\\select length(ename) from emp;

4

9

7

Select ascii(ename) from emp; (it only works for first letter.

65

66

67

Selct ascii(substr(ename,2)) from emp;

Select distinct ascii(‘z’) from emp;

Sign

1. Check if number is +ve or -ve
2. Sign(bank\_balance)
3. Sign(temperature)
4. Sign(sp-cp)
5. Sign(sensex)
6. Sign(netrunrate)
7. Sign(medicalreport)
8. Sign(sound)
9. Sign(altitude0

Lateral Thinking by Ayn

C 🡪 10

Y 🡪 20

To find out which is greater of the 2 numbers:-

Sign(x-y)

**Select mod(9,5) from dual;**

4

**Select mod(8.22,2.2)from dual;**

1.62

**Select sqrt(81) from dual;**

9

Char, number Functions

MySql- SQL – Date Functions and Formats

1. Date, Time, Datetime, Year
2. 1st Jan 1000 AD to 31st Dec 9999 AD
3. ‘YYYY-MM-DD’ or ‘YY-MM-DD’
4. 1970
5. Date1-date2
6. time1-time2
7. Datetime1-datetime2
8. Internally Date is Strored as a fixed- length number and it occupies 7 Bytes of storage (number of day since 1st jan 1000 AD)

1st Jan 1000 AD 🡪 1

2nd Jan 1000 AD 🡪

Etc.

1. Internally time is stored as a decimal(fraction of day)

1 🡪 1 day

1.5 🡪 1.5 days -> day 12 hours

1.75 🡪 1.75 days 🡪 day 18 hours

GMT 🡪 IST – 5.5 hours

GMT 🡪

SQL – Date Functions and formats

* When you enter the date, if you don’t specify the time, then
* Default value of time is 12 am idnight (00:00:00)

Select sysdate() from dual; 🡨 system date

2025-04-11 18:37:09

* Sysdate() is a function, it returns current current date
* Sysdate() returns DB Server machine date and time
* Sysdate() 🡨 returns the date and time when the statement executed
* Now() 🡨 return the date and time when the statement began to execute

**ACID Properties in RDBMS:-**

**A🡪** **Atomacity’**

Either all the operations in a transaction are completed successfully, or none of them are.

**C🡪 Consistency**

All Operations within a Transaction must follow the database’s rules and constraints, ensuring that the data remains valid and accurate.

**I 🡪 Isolation**

Concurrent Transactions sShould not interfere with each other.

**D 🡪 Durability**

Once a transaction is committed, its changes are permanently stored and will survive system failures, ensuring that the data is not lost even if the database system crashes.

**Day 5**

**MySQL – SQL – LIST Functions(ifnull)**

* Independent of datatype

Select \* from emp where comm = null;

* Any comparision done with null,returns null

Pessimistic querying 🡪 Searching for null value

**Select \* from emp where comm = null;**

**Select \* from emp where comm is null;**

**Select sal+comm from emp;**

Is null 🡪 special operator

* Any operation done with null, returns null

Select sal + ifnull(comm,0) from emp;

Ifnull(comm,0)

Ifnull(comm,100)

If(city,’Mumbai’)

Ifnull(orderdate(

|  |
| --- |
| EMP |
| Sal |
| 1000 |
| 2000 |
| 3000 |
| 4000 |
| 5000 |
|  |

Select greatest(sal,3000)from emp; 🡨 returns greater of the two values

3000

3000

3000

4000

5000

Uses:-

1. To set a lower limir on some value
2. E.g bonus = 10% sal, min bonus = 300
3. Select greatest(sal\*0.1,300) from emp;

MySql – SQL -LIST Functions(ifnull,greatest)

* Independent of datatype
* Select least(sal,3000) from emp; 🡨 returns smaller of the 2 values

1000

2000

3000

3000

3000

Uses:-

1. to set an upper limit on some value

e.g. cashback = 10% amt, max cashback=1000

e.g bonus = 10% sal, min bonus = 300

select greatest(sal\*0.1,300) from emp;

greatest(ccol1,col2,col3,…..,colcol255) 🡨 returns the greatest

greatest(val1,val2,val3,…….,val255)

greatest(num1,num2,num3)

greatest(str1,str2,str3,str4)

greatest(date1,date2,date3)

greatest(time1,time2,time3,time4)

greatest(datetime1,datetime2,datetime3)

MySql – SQL -LIST Functions(ifnull,greatest,least)

* independent of datatype

least(col1,col2,col3,……col255) 🡨 return the least

least(val1,val2,val3,……val255)

least(str1,str2,str3,str4)

least(date1,date2,date3)

least(time1,time2,time3,time4)

least(datatime1,dtetime2,datatime3)

set x = greatest(a,b,c,d);

set y = least(a,b,c,d);

* MySQL – SQL – CASE expression

**EMP**

|  |  |
| --- | --- |
| **SAL** | **DEPTNO** |
| 1000 | 10 |
| 2000 | 20 |
| 3000 | 30 |
| 4000 | 50 |
| 5000 | 50 |

**Select case when deptno = 10 then ‘Training’**

**When deptno = 20 then ‘Exports’**

**When deptno = 30 then ‘Marketing’**

**else ‘DEPTNAME’**

**from emp;**

**Output :-**

**DEPTNAME**

------------

Training

Training

Exports

Marketing

* ELSE is optional
* If you don’t specify ELSE, and if none of the cases are satisfied, then it returns a null value.

Select

Case

When deptno = 10 then ‘Ten’

when deptno = 20 then ‘Twenty’

when deptno= 30 then ‘Thirty’

when deptno = 40 then ‘Forty’

end “DEPARTMENT CODE”

from emp;

if deptno = 10 then HRA = 40% annual

if deptno = 20 then HRA = 30% annual

if deptno = 30 then HRA= 25% annual

else HRA= 20%

Select deptno, ename, sal, sal\*12, “Annual”,

Case

When deptno = 10 then sal\*12\*0.4

when deptno = 20 then sal\*12\*0.3

when deptno= 30 then sal\*12\*0.25

else sal\*12\*0.2

end “HRA”

from emp;

order by 1;

if sal > 3000 then REMARK = ‘High Income’

if sal < 3000 then REMARK = ‘Low Income’

if sal = 300 then REMARK = ‘Middle Income’

select ename, sal,

case

when sign(sal-3000) = 1 then ‘High Income’

when sign(sal-3000) = -1 then ‘Low Income’

else “Middle Income’

end “REMARKS”

from emp

order by 2;

**MySQL – SQL – Environment Functions**

Select user() from dual;

Amit@localhost

**Uses:-**

1. to maintain logs of DML operations and other events

**show character set;**

**MySQL – SQL – Group/Aggregate Functions (V. Imp)**

Character, Number, Date, List, Environment Functions

**EMP**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EMPNO** | **ENAME** | **SAL** | **DEPTNO** | **JOB** | **MGR** |
| **1** | **Arun** | **8000** | **1** | **M** | **4** |
| **2** | **Ali** | **7000** | **1** | **C** | **1** |
| **3** | **Kirun** | **3000** | **1** | **C** | **1** |
| **4** | **Jack** | **9000** | **2** | **M** |  |
| **5** | **Thomas** | **8000** | **2** | **C** | **4** |

**Single-Row Functions:-**

* Will operate on 1 row a time
* Character, Number, Date, List, Environment Functions
* E.g.upper(ename), round(sal), etc.

Multi -RowFunctions:-

* Will operate on multiple rows at a time.

**MySQL – SQL – GROUP/Aggregate Functions**

DB Server Hard Disk

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EMPNO** | **ENAME** | **SAL** | **DEPTNO** | **JOB** | **MGR** |
| **1** | **Arun** | **8000** | **1** | **M** | **4** |
| **2** | **Ali** | **7000** | **1** | **C** | **1** |
| **3** | **Kirun** | **3000** | **1** | **C** | **1** |
| **4** | **Jack** | **9000** | **2** | **M** |  |
| **5** | **Thomas** | **8000** | **2** | **C** | **4** |

Server Ram 🡪

8000

7000

3000 =35000

9000

8000

**Select sum(sal) from emp;**

o/p :35000

* - - - - - - - - - - - - - - -

Assumption, 4th row SAL is null :-

Select sum(sal) from emp; 🡨 Null Values ARE NOT Counted BY GROUP FUNCTIONS

26000

Select sum(inull(sal,0)) from emp; 🡨 you can do this but its not required

2000

Assumption, 4th row SAL is null:-

Select avg(sal) from emp; 🡨 Null Values ARE NOT Counted BY GROUP FUNCTIONS

26000/4 🡪 6500

Select avg(ifnull(sal,0)) from emp;

26000/5 🡪 5200

* As per User/Client Requirements
* - - - - - - - - - - - - - - - -

Assumption , 4th row SAL is Null:-

Select min(sal) from emp;

3000

Select min(ifnull(sal,0)) from emp;

0

* - - - - - - - - - - - - - - - - - - - - - - - -

Assumption, 4th row SAL is null :-

Select max(sal) from emp;

8000

* - - - - - - - - - - - - - - - - - - - - - - - -

Assumption , 4th row SAL is null:-

Select count(sal) from emp; 🡨 returns a count of number of rows where SAL is not having a null value

4

* - - - - - - - - - - - - - - - - - - - - - - - -

Assumption, 4th row SAL is null :-

Select count(sal) from emp; <- returns a count of numberof rows where SAL is not having a null value

4

-------------------------------------------------------------

Assumption, 4th row SAL is null :-

Select count(\*) from emp; 🡨 returns a count of total number of rows in the table

5

Select count(\*) – count(sal) from emp;

1

Select max(sal)/min(sal) from emp;

8000/3000 🡪 2.67

Select sum(sal)/count(\*) from emp;

26000/5

Select avg9ifnull(sal,0)) from emp;

26000/5

* - - - - - - - - - - - - - - - - - - - - - - - -

Assumption 4th row SAL is 9000 :-

Select sum(sal)from emp where deptno =1; 🡨 used to used to retrive the rows

18000

Select avg(sal) from emp where job = ‘C’;

6000

Count-Query (counting the number of query hits) :-

Select count(\*) from emp

Where sal > 7000;

3

Sum(column)

avg(column)

min(column)

max(column)

count(column)

count(\*)

stddev(column)

variance(column)

Summary Report:-

Select count(\*), min(sal), max(sal), sum(sal), avg(sal) from emp;

Count(\*) MIN(SAL) MAX(SAL) SUM(SAL) AVG(SAL)

----------- --------------- --------------- -------------- --------------

14 800 5000 29025 2073

**Restriction #1 :-**

* You cannot SELECT column of table alongwith a group function

Select count(ename), min(sal) from emp;

Select ename, min(sal) from emp; 🡨 Error

**Restriction #2 :-**

Select count(ename), min(sal) from emp;

Select upper(ename), min(sal) from emp; 🡨 ERROR

**Restriction #3 :-**

* You can use a Group function in the WHERE clause
* WHERE CLAUSE is used for searching
* Searching takes place in DB server HD
* At the time of searching in the table, min(sal) deoesnot exist
* Min(sal) is calculatedby MySQL AFTER the rows are brought into the server RAM

Select \* from emp where sal = min(sal); 🡨 ERROR

MySQL – SQL -Group By Clause (V. IMP)

* Where clause 🡪 used for Searching(used to restrict/retrieve the rows)
* Order By Clause 🡪 used for locking the rows manually
* Used for grouping

Select sum(sal) from emp;

35000

Select sum(sal) from emp

Where deptno = 1;

18000

* Sum(sal) deptwise:-

Select deptno, sum(sal) from emp

Group by deptno;

DEPTNO SUM(SAL)

------------ ---------------

1 18000

2 17000

Server Ram

1 8000

1 7000

1 3000

2 9000

2 8000

1. **Rows are retrieved from DB server HD to Server RAM**
2. **Sorting deptwise**
3. **Grouping deptwise**
4. **Summation deptwise**
5. **HAVING clause**
6. **ORDER BY clause**

Select clause 🡪 select deptno, sum(sal)

FROM clause 🡪 from emp

GROUP BY CLAUSE 🡪 group by deptno

**Rule #1 :-**

**Besides the group function, whichever column is present in SELECT clause**

It has to be present in Group BY clause

Select deptno, su(sal) from emp; 🡨 error

Select deptno, sum(sal) from emp

Group by deptno;

Select sum(sal) from emp

Group by deptno;

Sum(SAL)

**Rule #2:-**

* Whichever column is present in GROUP BY clause, it may or may not br present in SELECT CLAUSE
* SELECT sum(sal) from emp

Group by deptno;

Sum(SAL)

* - - - -

18000

17000

* Select deptno, sum(sal) from emp

Group by deptno;

* Select deptno, max(sal) from emp

Group by deptno;

* Select deptno, min(sal) from emp

Group by deptno;

* Select deptno, count(\*) from emp

Group by deptno;

* Any Select statement with a GROUP BY CLAUSE is known as a 2D query;

Because you can plot a graph from the output

* Recommended s/w:-
* Oracle Graphics.
* Select job, sum(sal) from emp

Group by job;

* Select deptno, sum(sal) from emp 🡪 used to retrieve the rows

Where sal > 7000

Group by deptno;

* WHERE clause is specified BEFORE the GROUP BY clause
* Select deptno, sum(sal) from emp

Where job = ‘C’

Group by deptno;

* Select deptno, job, sum(sal) from emp

Group by deptno, job;

* No upper limit on the number of columns in GROUP BY CLAUSE

Select …………………………..

Group by country, state, city;

* If you have a large number of rows in the table, and a large number of columns in GROUP BY CLAUSE, then that SELECT statement

Will be very slow, because that much sorting has to take place in the server RAM

1 columns in GROUP BY CLAUSE 🡪 2D query.

2 columns in GROUP BY clause 🡪 3D query.

3 columns in GROUP BY clause 🡪 4D query.

etc.

Known as Multi - Dimentional Queries .

* Select deptno, job, sum(sal) from emp

Group by deptno, job;

* Select job, deptno, sum(saal) from emp

Group by job, deptno;

* Select sum(sal), job, deptno from emp

Group by deptno, job;

* Position/order of columns in SELECT clause and the position/order

Of columns in GROUP BY clause need not be the same

* Position/order of columns in SELECT clause will determine the position of columns in the output; this you will wrte as per User/Client requirements.
* Position/Order of columns in GROUP clause will determine the sorting order, the grouping, summation order, and hence the speed of processing; this you will write as per count(distinct(columnname))

Select count(distinct deptno), count(distinct job) from emp;

25 400

Select deptno, job, sum(sal) from emp 🡨 Faster(16x)

Group by deptno,job;

Select job, deptno,sum(sal) from emp 🡨 SLOW

Group by job, deptno;

Select………………………………………….

Group by city, country, district, state; 🡨 SLOW

* Selct deptno, job, sum(sal) from emp

Group by deptno, job;

* Select deptno, sum(sal) from emp

Group by deptno, job;

* Select deptno, sum(sal) from emp

Group by deptno;

DEPTNO SUM(SAL)

* - - - ------------

1 18000

2 17000

* Select deptno, sum(sal) from emp

Group by deptno

Having sum(sal) > 17000;

DEPTNO SUM(SAL)

* - - - ------------

1 18000

2 17000

* Select deptno, sum(sal) from emp

Where sum(sal) > 17000 🡨 Error

Group by deptno;

* Select deptno, sum(sal) from emp

Where sal > 7000

Group by deptno;

Select deptno, sum(sal) from emp

Group by deptno

Having sal > 7000; 🡨 ERROR(AFTER summation, SAL does not exist in the server RAM, only DEPTNO and SUM(SAL) exist in the Server RAM)

* Whichever column is present in SELECR clause, it can be used in HAVING clause

* Select deptno, sum(sal) from emp

Group by deptno

Having deptno = 1; 🡨 This will work but it is INEFICENT; SHOULD HAVE USED WHERE DEPTNO = 1

DEPTNO SUM(SAL)

----------- -------------

1. 18000

* Select deptno, sum(sal) from emp

Where deptno = 1

Group by deptno;

DEPTNO SUM(SAL)

----------- -------------

1. 18000

* It’s recommended that only the group functions hould be used in HAVING clause
* Select deptno, sum(sal) from emp

Group by deptno

Having sum(sal) > 17000 and sum(sal) < 25000;

* Select deptno, sum(sal) from emp

Group by deptno

Having sum(sal) between 17001 and 24999;

Select deptno, sum(sal) from emp

DEPTNO SUM(SAL)

----------- ----------

1. 18000
2. 17000

* Select deptno, sum(sal) from emp

Group by deptno

Order by sum(sal);

DEPTNO SUM(SAL)

----------- ----------

1. 17000
2. 18000
3. **Rows are retrieved from DB server HD to Server RAM**
4. **Sorting deptwise**
5. **Grouping deptwise**
6. **Summation deptwise**
7. **HAVING clause**
8. **ORDER BY clause**

Select ……………………. From …………………

Where…………….

Group by………….

Having……………

Order by………….

* ORDER BY CLAUSE is the LAST cause in SELECT statement
* select deptno, sum(sal) from emp

Where job = ‘C’

Group by deptno

Having sum(sal) > 9000

Order by 1;

* select deptno, count(\*), min(sal), sum(sal) from emp

group by deptno

order by 1;

DEPTNO COUNT(\*) MIN(SAL) MAX(SAL) SUM(SAL)

------------------------------------------------------------------- ------------------

10 3 1300 5000 8750

20 5 800 3000 10875

30 6 950 2850 9400

In Other RDBMS :-

* Select deptno, sum(Sal) from emp

Group by deptno;

DEPTNO SUM(SAL)

------------ ---------------

1. 18000
2. 17000

* Select sum(sal) from emp

Group by deptno;

SUM(SAL)

-------------

18000

17000

* Select max(sum(sal)) from emp

Group by deptno; 🡨 NESTING OF GROUP FUNCTIONS IS SUPPORTED ONLY IN ORACLE RDBMS;

NOT SUPPORTED BY ANY OTHER RDBMS

* Select sum(sal) from emp

Group by deptno;

**Sum\_sal**

-------------

18000

17000

* (select sum(sal) sum\_sal from emp

Group by deptno) abcd;

Sum\_Sal

----------

18000

1700

In MySQL:-

Select sum(sal) from emp

Group by deptno;

Sum(sal)

----------

18000

17000

Select sum(sal) sum\_sal from emp

Group by deptno;

Sum\_sal

----------

18000

17000

Select max(sum\_sal) from

(select sum(sal) sum\_sal from emp

Group by deptno) abcd;

Max(sum\_sal)

-----------------

18000

-------------------------------------------

**Day 6**

**MySQL – SQL – Joins (V. V. IMP) (Most IMP)**

**\* 5 Types of Joins**

**EMP**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EMPNO** | **ENAME** | **SAL** | **DEPTNO** | **JOB** | **MGR** |
| 1 | Arun | 8000 | 1 | M | 4 |
| 2 | Ali | 7000 | 1 | C | 1 |
| 3 | Kirun | 3000 | 1 | C | 1 |
| 4 | Jack | 9000 | 2 | M |  |
| 5 | Thomas | 8000 | 2 | C | 4 |

**DEPT**

|  |  |  |
| --- | --- | --- |
| **DEPTNO** | **DNAME** | **LOC** |
| 1 | TRB | Bby |
| 2 | EXP | Dlh |
| 3 | MKTG | Cal |

* **Data Redundancy** 🡪 Unnecessary duplication of data, and that’s a wastage of HD space.
* **Normalisation**
* Select dname,ename from emp, dept

where dept.deptno = emp.deptno;

Dept.deptno 🡪 tablename.columnname

Emp.deptno 🡪 tablename.columnname

* Select ename, dname from emp,dept

Where dept.deptno = emp.deptno;

* To make the output more Presentable:-
* Select dname, ename from emp, dept

Where dept.deptno = emp.deptno order by 1;

* The position of columns in SELECT, eill determine the position of columns in the output; this you will write as per User/client requiremets
* Select dname, ename from emp, dept

Where dept.deptno = emp.deptno;

* Select ename, dname from emp, dept

Where dept.deptno = emp.deptno;

* You may write tablename.columnnme in the WHERE clauses in any order, the output will be the same
* Select dname, ename from emp, dept

Where dept.deptno = emp.deptno;

OR

* Select dname, ename from emp, dept

Where emp.deptno = dept.deptno;

Dept 🡪 driving table

Emp 🡪 driven table

<🡨----

* Select dname, ename from emp, dept

Where dept.deptno = emp.deptno; 🡨 FAST

Emp 🡪 driven table

Dept 🡪 driven table

<🡨

* Select dname, ename from dept, emp 🡨 SLOW

Where dept.deptno = emp.deptno;

* IN ORDER FOR THE JOIN TO WORK FASTER; PREFERABLY THE DRIVING TABLE SHOULD BE TABLE WITH LESSIER NUMBER OF ROWS
* The commeon column in both the tables(DEPTNO); the columnname need not be the same in both the tables, because the same column can have a different meaning elsewhere; e.g. EXPORT I]in one table = IMPORT in another table, EXPENSE in one table = INCOME in another table, etc.
* Select dname, ename from emp, dept

Where dept.x = emp.y;

* What matters is the datatype of the common column should match in both the tables, and there should be some meaningful relation on whose basis you are writing the join; e.g. the below SELECT statement will work but the output would be meaningless:-
* Select dname, ename from emp, dept

Where dept.deptno = emp.empno;

* To make The output more presentable:-
* Select dname, ename from emp, dept

Where dept.deptno = emp.deptno

Order by 1;

* It’s good to SELECT lots of columnsin the output; the output becomes richer and more meaningful:-
* Select dname, loc, ename, job, sal from emp, dept

Where dept.deptno = emp.deptno

Order by 1;

* To see ALL the columns from BOTH the tables:-
* Select \* from emp, dept

Where dept.deptno = emp.deptno

Order by 1;

* Common in both the tables (DEPTNO) will be displayed twice
* To see the common column only once in the output, you will to SELECT all the columns explicitly:-

(\*) 🡨 column ambiguously defined

* Select deptno, dname, loc, ename, job, sal from emp, dept

Where dept.deptno = emp.deptno

Order by 1;

To make the above work:-

* Select dept.depto, dname, loc, ename, job, sal from emp, dept where dept.deptno = emp.deptno

Order by 1;

OR

* Select emp.deptno, dname, loc, enmae, job, sal from emp, dept

Where dept.deptno = emp.deptno

Order by 1;

* It’s a good programming practice to use tablename.columnname for all the column; it makes the SELECT statement more Readable
* Select dept.deptno, dept.dname, dept.loc, emp.ename, emp.job, emp.sal

From emp, dept

Where dept.deptno = emp.deptno

Order by 1;

---------------------------------------------------------------

* Select deptno, sum(sal/0 from emp

Group by deptno;

DEPTNO SUM(SAL)

------------ --------------

1. 18000

2 17000

* Select dname, sum(sal) from emp, dept

Where dept.deptno = emp.deptno

Group by dname;

DNAME SUM(SAL)

TRN 18000

EXP 17000

* Select upper(dname), sum(sal) from emp, dept

Where dept.deptno = emp.deptno

Group by upper(dname);

* Select dname, sum(sal) from emp, dept

Where dept.deptno = emp.deptno

Group by dname

Having sum(sal) > 10000

Order by 1;

* **Types of joins :-**

1. **Equijoin :-**

* Join based on equality condition.
* Shows matching rows of both the tables

\* Uses:-

All the data is not stored in one table,the data is stored in multiple tables; if you want to see the columns of 2 or more tables.

1. Dname, ENAME
2. CNAME, SNAME

etc.

\* this is the most frequentlyused join (>90%), and therefore it’s also known as Natural join

* Select dname, ename from emp, dept

Where dept.deptno = emp.deptno;

DNAME ENAME

----------- -----------

TRN ARUN

TRN ALI

TRN KIRUN

EXP JACK

EXP THOMAS

1. **Inequijoin**

\* join based on inequality condition

\* shows non-matching rows of both the tables

\* Uses:-

a. EXCEPTION REPORTS

Who are the employees who don’t belong to TRN?

Who are the Customers who have not made the payments?

* Select dname, ename from emp,dept

Where.deptno != emp.deptno;

DNAME ENAME

----------- -----------

TRN JACK

TRN Thomas

EXP Arun

EXP Ali

EXP Kirun

MKTG Arun

MKTG Ali

1. **Outerjoin**

\* join with (+) sign

\* shows matching rows of both the tables

Plus

Non-matching rows of “Outer” table

\* Outer table 🡪 table which is on Outer side of (+) sign.

* select dname, ename from emp, dept

Where dept.deptno = emp.deptno (+);

* Select dname, ename from emp, dept

Where dept.deptno(+) = emp.deptno;

**DNAME ENAME**

----------- -----------

TRN Arun

TRN Ali

TRN Kiran

EXP Jack

EXP Thomas

* Select dname, ename from emp, dept

Where dept.deptno = emp.deptno (+);

**DNAME ENAME**

**------------ ------------**

TRN Arun

TRN Ali

TRN Kiran

EXP Jack

EXP Thomas

---------------------------------------------------------------------------

1. **Half Outerjoin**

\* (+) sign on any 1 side, i.e LHS or RHS

b.**Full Outerjoin**

\* (+) sign on both the sides (which is not allowed)

\* based on Nested Do-while loop

* Select dname,ename from emp.dept

Where dept.deptno = emp.deptno(+) <= right outerjoin

* Select dname, ename from emp,dept 🡸 left outerjoin

Where dept.deptno(+) = emp.deptno;

* **Full Outerjoin**

\* shows matching rows of both the tables

Plus

Non-matchingrows of both the tables

* Select dname,ename from emp.dept

Where dept.deptno = emp.deptno(+) <=

union

Select dname, ename from emp,dept

Where dept.deptno(+) = emp.deptno;

ANSI syntax for Full Outerjoin:-

* Select dname, ename from full outer join dept

On (dept.deptnp = emp.deptno);

**ANSI syntax for RightOuterjoin:-**

* Select dname, ename from emp right outer join dept

On (dept.deptno = emp.deptno);

**ANSI syntax for left Outerjoin:-**

* Select dname, ename from emp left outer join dept

On (dept.deptno = emp .deptno)

* (+) sign for Outerjoin is Supported ONLY by Oracle RDBMS; not

Supported by any other RDBMS.

* ANSI syntax for Right outerjoin is supported by all RDBMS including

MySQL.

* ANSI syntex for Left Outerjoin is supported by all RDBMS including MySQL.
* ANSI syntax for FULL Outerjoin is supported by all RDBMS except
* For MySQL

To implement Full Outerjoin in MySQL, take a UNION of ANSI syntax for Right Outerjoin and ANSI syntax for left Outerjoin:-

* Select dname, ename from emp emp right outer join dept

On(dept.deptno = emp.deptno)

Select dname, ename from emp left outer join dept

On (dept.deptno = emp.deptno);

* Inner Join :-

\* DO NOT MENTION IN INTERVIEWS UNLESS EXPLICITY AKSED BY INTERVIWER

\* By default every join is an Inner join; using a(+) sign or using

The keyword “Outer” is what makes it an Outerjoin

1. **Cartesian join (Cross Join)**

* Join without a WHERE clause
* Every rowof driving table is combined with each and every row of

Driven table

* Cross product of 2 tables, and therefore it’s also known as a

Cross join.

* **Uses:-**

1. Returns all the combinations

e.g ::

in STUDNTS table you have ll thestudents names, I SUBJECTS table you haveall the subjects names; when you are printing the mark-sheets for the students, every student name is combined with each and every subject name

Dept 🡪 driving table

Emp🡪driven table

<< -----

Select dname, ename from emp , dept; 🡨 FAST (The lesser the I/O between the DB Server HD and Server RAM, the faster it will be)

<< --------

Select dname, ename from dept, emp; 🡨 SLOW (The more the I/O between the DB Server HD and Server RAM, the slower it will be)

**DNAME ENAME**

**------------ ------------**

TRN Arun

TRN Ali

TRN Kirun

TRN Jack

TRN Thomas

EXP Arun

EXP Ali

EXP Kiran

EXP Jack

EXP Thomas

MKTG Arun

MKTG Ali

MKTG Jack

MKTG Thomas

1. **Self Join**

* Joining a table to itself
* Used when parent column and child, both are present in the

same table.

* Slowest join
* Based on Recursion

All Joins are slow:-

* Comparatively Self join is theslowest join

(because two copies of entire table are brought into Server RAM)

* Comparativelycartesian join is the fastest join

(there is WHERE clause and therefore there is no searching

Involved)

**<< -----**

* Select a.ename, b.ename from emp as b, emp a

Where a.mgr = b.empno;

**Server Ram:**

**A**

**ENAME MGR**

-------------- -------------

Arun 4

Ali 1

Kiran 1

Jack 1

Thomas 4

**B**

**ENAME MGR**

----------- -----------

Arun 1

Ali 2

Kiran 3

Jack 4

Thomas 5

**A.Ename B.Ename**

----------- -----------

Arun Jack

Ali Arun

Kiran Arun

Thomas Jack

**Day7**

**Joins :-**

|  |  |  |
| --- | --- | --- |
| **DEPTNO** | **DNAME** | **LOC** |
| **1** | **TRN** | **Bby** |
| **2** | **EXP** | **Dlh** |
| **3** | **MKTG** | **Cal** |

**DEPTHEAD DEPT**

|  |  |
| --- | --- |
| **DRPYNO** | **DHEAD** |
| 1 | **Arun** |
| **2** | **Jack** |

**EMP**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EMPNO** | **ENAME** | **SAL** | **DEPTNO** | **JOB** | **MGR** |
| 1 | Arun | 8000 | 1 | M | 4 |
| 2 | Ali | 7000 | 1 | C | 1 |
| 3 | Kirun | 3000 | 1 | C | 1 |
| 4 | Jack | 9000 | 2 | M |  |
| 5 | Thomas | 8000 | 2 | C | 4 |

* **Joining 2 or more tables :-**
* select dname, ename, dhead from emp, dept, depthead

where depthead.deptno = dept.deptno

and dept.deptno = emp.deptno;

**DNAME ENAME DHEAD**

----------- ----------- -----------

TRN Arun Arun

TRN Ali Kiran

TRN Kiran Arun

EXP JACK JACK

EXP Thomas Jack

**EMP**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EMPNO** | **ENAME** | **SAL** | **DEPTNO** | **JOB** | **MGR** |
| 1 | Arun | 8000 | 1 | M | 4 |
| 2 | Ali | 7000 | 1 | C | 1 |
| 3 | Kirun | 3000 | 1 | C | 1 |
| 4 | Jack | 9000 | 2 | M |  |
| 5 | Thomas | 8000 | 2 | C | 4 |

**PROJECTS**

|  |  |  |
| --- | --- | --- |
| **PROJNO** | **CLIENT\_NAME** | **DESCRIP** |
| P1 | Deloitte | CGS |
| P2 | Morgan Stanley | AMS |
| P3 | BNP Paribas | Mcros |
| P4 | ICICI Bank | PPS |
| P5 | AMFI | Website Devl |

**Projects\_Emp**

|  |  |
| --- | --- |
| **ProjectNo** | EmpNo |
| P1 | 1 |
| P1 | 2 |
| P1 | 4 |
| P2 | 1 |
| P2 | 3 |
| P3 | 2 |
| p3 | 4 |
| P3 | 5 |
| P3 | 5 |

* **Types of Relationships between tables** :-

**1 : 1** (Dept : DeptHead) or (Depthead : head)

**1 : Many** (Dept :Emp) and (Depthead : Emp)

**Many : 1** (Emp : Dept) and (Emp : Depthead)

**Many : Many** (Emp : Projects) or (Projects: Emp)

* **Interection table is required for Many : Many Relationship**
* Display the Ename Whois receiving SAL = min(SAL):\_
* Select ename from 🡨 Main Query (Outer) (Parent)

Where sal =

(select min(sal) from emp); 🡨Sub Query (Inner) (Child)

* Max upto 255 levels for sub-queries(common for all RDBMS)

(this limit of SQL can be exceeded with the help of Views)

* JOIN IS FASTER THAN SUB-QUERY
* When you solve the problem using a join, you solve the problem using one SELECT statement.
* The more the number of SELECT statements, the slower it will be
* Display all the rows belongs to the DEPTMO as ‘Thomas’ :-

Select \* from emp

Where deptno = (2)

(selectename = deptno from emp

Where ename = ‘Thomas’);

-------------------------------------------

* Select \* from emp

Where deptno !=

(select depot fromemp

Where ename = ‘Thomas’);

* **Display 2nd largest SAL** :-
* Select max(sal) from emp

Where sal <

(select max(sal) from emp);

* Display all the rows who are doing the same JOB as ‘kiran’:-
* Select \* from emp

Where job =

(select job from emp

Where eame = ‘Kirun’);

* Select \* from emp

Where deptno =

(select deptno from dept

Where dname = ‘TRN’);

-------------------------------------------------------------

Using sub-queries with DML commands:-

* Delete from emp

Where deptno= (2)

(select deptno from dept

Where dname =’TRN’);

* update emp set sal = 1000

where deptno = (2)

(select deptno from dept

Where dname =’TRN’);

--------------------------------------------------------------

**In MySQL , you cannot UPDATE or DELETE from a table from which you aee currently Selecting**

* Delete from emp

Where deptno=

(select abcd.deptno from

(select deptno from emp

Where ename = ‘Thomas’)abcd);

* Update empset sal =10000

Where job=

(select abcd.job from

(selet job from emp

Where ename = ‘Kirun’)abcd);

-----------------------------------------------------------------------------------------

Multi-Row sub-queries

* Sub-query retuns multiple rows

Select \* from emp

Where sal = any (8000,9000)

(select sal from emp

Where job=’M’);

Select \* from emp

Where sal in (8000,9000)

(select sal from emp

Where job = ‘M’)

* To make it work faster:-

1,. Try to solve the problem using s join; because join is faster than sub- query.

2.Try to reduce the levels for sub-queries; the more the number of levels

For sub-queries, te slower it will be.

3. Try to rreduce the number of rows returned by sub-query; the more the number of rows returned by sub-query, the slower it will be.

* IN Operator is faster than ANY operator(because ANY operator is Overloaded)
* ANY operator is more powerful than IN operator
* With the IN operator, you can check for = ANY, >Any, >=Any,<Any, <=Any
* If you want to check for inequality, then the
* 9n operator
* If you eant check for >, >=, N, r <= thenuse ANY Operator

Any🡪 Logical OR

IN 🡪 Logical OR

* Uni0n, union all, Intersect, except, oder by

RDBMS-supplied pseudocolumns

**Day 8**

**EMP**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ROWID | EMPNO | SAL | SAL | DEPTNO |
| X001 | 5 | A | 5000 | 1 |
| X002 | 4 | A | 6000 | 1 |
| X003 | 1 | C | 7000 | 1 |
| X004 | 2 | D | 9000 | 2 |
| X005 | 3 | E | 8000 | 2 |

**MySQL- INDEXES**

* Present in all the RDBMS, all DBMS and some of the programming languages also
* To speed up the search operations(for faster success)
* To speed up SELECT statement with a WHERE clause.
* Indexes are automatically invoked by MySql a and when required.
* Duplicate values are stored in an index
* Null vaues are not stored in an index
* Indexes are automatically updated by MySQL for all the DML operations.
* MySQLis a self-managing RDBMS
* No upper limit on the number of indexes per table
* Larger the number of indexes, slower would be the DML operations.
* You cannot index Text and Blob columns
* If you have 2 or more INDEPENDENT columns in the WHERE clause, then create separate indexes for each column and MYSQL will use all the necessary indexes as and then when required.
* COMPOSITE INDEX -> combine 2 or more INTER-DEPENDENT columns in a single index.
* INDEX KEY 🡪 column or set of columns on whose basis the index has been created
* In MySQL, you can combine upto 32 columns in a composite index
* Conditions when index should be created

\* to speed up SELECT statement with a WHERE clause, ORDER BY clause, GROUP BY CLAUSE, DISTINCT, UNION, INTERSECT, EXCEPT.

If select retrives <25% of table data

Primary key nd unique column should be indexed

Common columns in join operations should always be indexed.

IN Other RDBMS:-

Insert/update/delete……………………………

REINDEX;

Types of Indexes:-

1. Normal Index
2. Unique Index
3. Normal Composite Index
4. Composite Unique in

-----------------------------------------------------------------------------------------------

**EMP**

|  |  |  |  |
| --- | --- | --- | --- |
| **EMPNO** | **ENAME** | **SAL** | **DEPTNO** |
| 1 | A | 5000 | 1 |
| 2 | B | 6000 | 1 |
| 3 | C | 7000 | 1 |
|  | D | 8000 | 2 |
|  | E | 9000 | 2 |

SQL :-

CONSTRANTS

* Limitations/resreiction imposed on a table
* DATATYPE is also a type of constraint; it’s known as Domain Inregrity

X char(10)

Y int unsigned(0 to 4,294,967,295)

\* PRIMARY KEY

* Primary column
* Column or set of columns that uniquely identifies a row (e.g.EMPNO)
* Duplicate values are not allowed; it has to be unique.
* Null values are not allowed; it is mandatory column.
* It’s recommended that every table should have a primary key; it helps from along-term prespective.
* Purpose of primary key is row uniqueness; with the help of primary .
* Unique index automatically created.
* COMPOSITE PRIMARY KEY 🡪 combine2 or more columns together to serve the purpose.
* IN MySQL, you can combine upto 32 columns ina composite primary key.

--------------------------------------------------------------------------------------------

Create table emp

(

Empno char(4) primary key,

Sal float,

Deptno int

);

Mysql>insert into emp emp values(‘5’, ‘F’, 5000,2); 🡨 Error

Mysql>insert into emp emp values(null, ‘F’, 5000,2); 🡨 Error

* All constraints are at Server level; you can perform DML operations using MySQL command Line Client, MySQL workbetch, c++, java, MS .net, tec. Or any front-end s/w; you cannot break the constraint
* This is known as Data Integrity
* Internally a constraint is MySQL created function, it perform the validations

Select \* from information\_schema.table\_constraints;

Select \* from information\_schema.table\_constraints

Where table\_schema = ‘mumbai’;

Select \* from information\_schema.key\_column\_usage

Where table\_schema = ‘emp’;

Constraints are of 2 types;-

a. column level constraints (specified on one column)

b. Table level constraint (specified on combination of two or more columns) (has tobe specified at the end of the table structure)

\* composite Unique index automatically created

Show indexes from emp;

Alter table emp drop primary key;

Alter table emp add primary key(deptno, empno);

* **Not Null Constraint**
* Null values are not allowed(mandatory column) (same as pk)
* Duplicate values are allowed (unlike pk)
* You can specify as many not null constarints as you want (unlike PK)
* You cannot have a composite not null constraint (unlike PK)
* You will have to specify a separate not null constraint for each column.
* Text and Blob can be not null (unlike PK)
* Index is not created automatically (unlike PK)
* Always a column level constraint; you cannot havea table level not null constraint.

\* **Nullability 🡪 Property of column whether null value is allowedor not**

**\* Nullability is a part of the datatype.**

To drop the constraint afterwards:-

Alter table emp modify ename varchar(25) null;

To add the not null constraint to an exiting table:-

Alter table emp modify ename varchar(25) not null;

**Day 10**

|  |  |  |  |
| --- | --- | --- | --- |
| **EMPNO** | **ENAME** | **SAL** | **DEPTNO** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**VIEWS**

Amit\_mysql > create view v1

As

Select empno, ename from emp;

root\_mysql> grant select, insert on mumbai.v1 to scott@localhost;

root\_mysql> flush privileges;

---------------------------------------------------------------------------

Scott\_mysql> insert into Mumbai.v1 values(6, ‘F’);i

* DML operations can be performed on aviews
* DML operations done on a view will affect the base table
* ENTIRE APPLICATION is BUILT ON VIEWS
* Constraints that are specified on the base table will be enforced
* Even if you INSERT via the view.

Drop view v1; (if you wan to drop view

* If you drop the table , then views remain.

Amit\_mysql > Create view v2

As select \* from emp where deptno =1

Root\_mysql> grant select,insert on Mumbai.v2 to King@localhost;

------------------------------------------------------------------------------------------

King\_mysql> select \* from Mumbai.v2;

**EMPNO ENAME SAL DEPTNO**

**------------ ----------- ------ ------------**

**1 A 5000 1**

**2 B 6000 1**

**3 C 7000 1**

--------------------------------------------------------------------------------------------

* View with checkoption can be used to enforce different check for different users.
* Can only select from this View
* DML operations are not allowed
* Common for all RDBMS.

--------------------------------------------

Role 🡪 set of Privileges.

**MySQL-PL**

* Product of MySQL
* MySQL Programming Language
* Programming language from MySQL.
* Used for database programming. E.g . HRA\_CALC, TAX\_CALC, ATTENDANCE\_CALC
* Every RDBMS HAS ITS OWN ‘NATIVE” programming language

e.g.

Oracle -> PL/SQL -> Procedural SQL.

Used for Server-side data processing.

* Has few 4 GL features
* Block level language.

Benefits:-

1. Modularity
2. Control scope of variable

* SQL commands allow in MySQL-PL:-

DDL, DML,DQL, DTL/TCL

Delete from emp where deptno =

(select deptno from emp where ename = ‘Thomas’);

* Can use SELECT statement inside the block but it’s not recommended
* Screen input and screen output is not allowed.
* DCL commands not allowed inside MySQL-PL
* MySQL – STORED OBJECTS
* Objects that are stored in the database
* E.g. CREATE….tables, indexes, views
* Anything that you do with CREATE command is a stored object

STORED PROCEDURES

* Routine (set of commands) that has to be called explicitely
* Stored in the database

**Day 11**

Loop, Leave, and Iterate Statements:-

* Leave statement allows you to exit the loop(similar to ‘break’ statement of ‘C’ Programming)
* Iterate statement slows you to skip the entire code under it and start a new iteration (similar to ‘continue’ statement of ‘C’ programming)
* Loop statement execute a block of code repeatedly with an additionaly flexibility of using a loop label.

Delimiter//

Create procedure abc()

Begin

Declare x int default 1;

Pqr\_loop:loop

If x > 10 then

Leave pqr\_loop;

End if;

Set x = x+1;

If mod(x,2) != 0 then

Iterate pqr\_loop;

Else

Insert into tempp values(x,’inside loop’);

End if;

End loop;

End; //

Delimiter ;

* In deeply nested loops, going from innermost loop a point outside the outermost loop, LEAVE is the fastest way of doing it.

**MYSQL-PL – CURSORS (MOST IMP) :-**

EMP

|  |  |  |  |
| --- | --- | --- | --- |
| **EMPNO** | **ENAME** | **SAL** | **DEPTNO** |
| 1 | A | 5000 | 1 |
| 2 | B | 6000 | 1 |
| 3 | C | 7000 | 1 |
| 4 | D | 9000 | 2 |
| 5 | E | 8000 | 2 |

**FIR**

|  |  |
| --- | --- |
| FIR | SEC |

* Present in all RDBMS, some of the DBMS, and some of thefront-ends also
* Cursor is atype of s variable.

Declare x int;

Declare hra float;

Select sal into x from emp where empno = 1;

Set hra = x\*0.4 ;

Insert………………….;

Set u = y+1;

* Present in all RDBMS, some of the DBMS, and some of the front-ends also
* Cursor is a type of s variable.
* Cursor can store multiple rows
* Cursor is similar to 2D array
* Cursor used for storing multiple rows
* Cursor used for processing multiple rows
* Cursor used for handling multiple rows
* Cursor used for storing the data temporarily
* Cursor is based on SELECT STATEMENT
* Cursor has to be declared AFTWR ALL the variables.
* Cursor is a READ\_ONLY variable
* The data that is present in the cursor, it cannot be manipulated
* You will have to fetch 1 row at a time into some intermediate variables, and do your processing with those varables
* You can only fetch sequentially (topo bottom)
* YOU CANNOT FETCH BACKWARDS IN MYSQL CURSOR.

delimiter//

Create procedure abc()

begin

declare a int;

declare b varchar(15);

declare c int;

declare d int;

declare x int default 0;

declare c1 cursor for select \* from emp; **🡨 CURSOR declaration/definition (AT THE POINT THE CURSOR DOES NOT CONTAIN ANY DATA**

open c1;

while x < 5 do

fetch c1 into a,b,c,d;

insert into tempp values(a,b\_;

set x = x+1;

end while;

close c1;

end;//

delimiter;