DAY 1 Training 11/09/24

DBMS

Data - Unstructured and structured, static and dynamic

RDBMS- relations among data - stored in table form

SQL – Structured query language

DBA - database administrator

SAM database - Security

data type- int, float, numeric, date and time, binary, varchar

Structure is called schema

file system - storing and managing the info

FQDN - fully qualified domain name - ex: www.google.com

types of DBMS:

- hierarchical model ex: DNS
- network
- relational

Roles in DBMS:

- system administrator - creates system objects
- DBA - maintains database objects, backup/recovery/reorganize data
- programmer - prepare logic and functions
- end user

maintenance operator - creates test environment

Database architecture

ODBC - open data base connectivity

JDBC - java data base connectivity

OLTP – online transaction processing

OLAP – online analytical process

API – application programming interface

JDO

ODBC database administrator → user dsn →

NAME- testdb

Description- this is a testing database connectivity

Server- localhost →next→ with sql server→password→ selest readwrite next change lang→ finish okr

User level(SQL Query)→ view level→ conceptual level→ physical level→ database (dbfile stored on harddisk)

Commit – complete transaction

Rollback – incomplete transaction

SQL query→ query lang processor (query parser+ optimizer)→ dbms engine(file manager+ transaction manager) → physical database

- Optimizer connected to index

- Parser tell db engine how to use data //both executes sql commands

Exp query - Use testdb;

Select * from emp;

To create database –

Syntax: create database testdb;

To access database –

Syntax: use testdb;

To delete database -

Syntax : drop database testdb;

To create user –

Syntax : create user 'sqladmin'@'localhost' identified by 'pass@123';

To grant access -

Syntax : grant all privileges on testdb.* to 'sqladmin'@'localhost';

To check granted access -

Syntax : show grants for 'sqladmin'@'localhost';

To apply the changes we have made -

Syntax : flush privileges;

TYPES OF DATABASES

1. RDBMS (used for structured data)
2. Nosql (used to handle unstructured and semistructured data) → (not only sql)
   ex- mongodb, couchdb, amazon dynamodb
3. OODBMS → objectdb, db4o
4. Hierarchical database → IBM, IMS
5. Network database → integrated dbms

6. In-memory database → redis
7. Distributed database → apache Cassandra

For SQL ex→ rdbms

MS-SQL →instance →Database files → main database files(primary db file) → Log files(transaction log files) (*.mdf) →Secondary db files.

Mandatory→ main db files(primary db), log files(transaction log files)

Optional secondary db

Index are stored in db itself.

Datafiles → *idb for innoDB and *.MYD for MyISAM

Log files

Index files

show databases;

use testdb;

create table dept (deptno int not null primary key, dname varchar(20), location varchar(20));

select * from dept;

insert into dept values (10,'accounting', 'bangalore');

insert into dept values (20,'accounting', 'bangalore');

insert into dept values (30,'HR', 'Tamil nadu');

insert into dept values (40,'admin', null);

update dept set dname='sales' where deptno=20;

update dept set location='telangana' where deptno=40;

alter table dept add dmgr varchar(20) not null;

update dept set dmgr='varshini' where deptno=10;

update dept set dmgr='almas' where deptno=20;

update dept set dmgr='bindu' where deptno=30;

update dept set dmgr='lahari' where deptno=40;

alter table dept add email varchar(50);

update dept set email='lahari@mphasis.com' where deptno=40;

alter table dept drop column email;

insert into dept values (50,'marketing', 'mangalore', 123.123);

show tables;

describe dept;

show columns from dept;

use sys;

To alter data from output → export the output file and save → open that in notepad →alter

copy file in notepad and save as “dbscript.sql” within double quotes

To run the same file in SQL command line type-

```
source C:\Users\varshini.r1\Documents\dbscript1.sql
```

DAY 1 Training Summary

what is a database

what is sql

what is mysql

basic sql commands(use, select, insert, alter, drop, create, update, grant, show,describe)

normalization

MySQL client(command line)

MySQL workbench

connecting to MySQL workbench

creating database, deleting a database, creating tables, inserting values into the table, altering table(adding and removing a column)

updating values

datatypes

database roles, structures

exporting data

execution of a SQL script

creating users and granting permission

DAY 2 TRAINING 12/09/24

DB development life cycle

1. Requirement analysis
2. Database designing
3. Implementation

ER modeling (Entity relationship) → graphical approach to db design.

ER diagram → displays the relationships between entities.

3 basic concepts →

- entities
- Attributes
- Relationship

Types of attributes

1. Simple
2. Composite
3. Derived
4. multivalued

To maintain domain integrity in a table we need primary key, constraints

MySQL datatypes →

Int, varchar, text, date, tinyint(1), blob, decimal

Create table dept:

```
use varshinidb;

drop table dept;     // removing old dept table

create table dept (
deptno int not null primary key comment 'department number',
dname varchar(20) not null comment 'department name',
loc varchar(20) not null comment 'location of the department'
);
select * from dept;
describe dept;
show full columns from dept;      // in order to show comments column
```

Create table emp:

```
create table emp (
        empno int not null primary key comment 'employees number',
        ename varchar(20) not null comment 'employees name',
        job char(10) not null comment 'designation',
        mgr int comment 'respective managers empno',
        hiredate date not null comment 'date of joining basic',
        sal numeric(9,2) not null comment 'salary',
        comm numeric(7,2) comment 'commission',
        deptno int not null comment 'department number'
);
select * from emp;
desc emp;
show full columns from emp;
```

To add foreign key in emp table:

```
alter table emp add constraint fk_emp foreign key (deptno) references dept(deptno)
on delete cascade
on update cascade;
```

exported data from output of above lines (notepad):

```
Table,"Create Table" emp,
  "CREATE TABLE `emp` (
  `empno` int NOT NULL COMMENT 'employees number',
  `ename` varchar(20) NOT NULL COMMENT 'employees name',
  `job` char(10) NOT NULL COMMENT 'designation',
  `mgr` int DEFAULT NULL COMMENT 'respective managers empno',
  `hiredate` date NOT NULL COMMENT 'date of joining basic',
  `sal` decimal(9,2) NOT NULL COMMENT 'salary',
  `comm` decimal(7,2) DEFAULT NULL COMMENT 'commission',
  `deptno` int NOT NULL COMMENT 'department number',
   PRIMARY KEY (`empno`),
   KEY `fk_emp` (`deptno`),
   CONSTRAINT `fk_emp` FOREIGN KEY (`deptno`) REFERENCES `dept`
(`deptno`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci"
```

Note:

1. department table containing the column department number (deptno) should be already present.
2. datatype of the primary and foreign key should match.

If the tables are already created in our example emp and dept tables were already created. In that case we can use alter table statement to link these two tables with the help of a **foreign key constraints as shown below:**

```
alter table emp add constraint fk_emp foreign key (deptno) references
dept(deptno) on delete cascade on update cascade;
```

Adding data to the dept table:

```
show create table emp;

insert into dept values (10,'accounting', 'new york');

insert into dept values (20,'research', 'dallas');

insert into dept values (30,'sales', 'chicago');

insert into dept values (40,'operations', 'bostan');

select * from dept;
```

Adding data to the emp table:

```
insert into emp values (7399,'smith', 'clerk', 7902, '1980-12-17', 800, null, 20);

insert into emp values (7499,'allen', 'salesman', 7698, '1981-02-20', 1600, 300, 30);

insert into emp values (7521,'ward', 'salesman', 7698, '1981-02-22', 1250, 500, 30);

insert into emp values (7566,'jones', 'manager', 7839, '1981-04-02', 2975, null, 20);

insert into emp values (7654,'martin', 'salesman', 7698, '1981-09-28', 1260, 1400,
30);
```

insert into emp values (7698,'blake', 'manager', 7839, '1981-05-02', 2850, null, 10);

insert into emp values (7782,'clark', 'analyst', 7566, '1982-12-09', 3000, null, 20);

insert into emp values (7839,'king', 'president', null, '1981-11-17', 5000, null, 10);

insert into emp values (7844,'turner', 'salesman', 7698, '1981-09-08', 1500, null, 30);

insert into emp values (7876,'adams', 'clerk', 7788, '1983-01-12', 1100, null, 20);

insert into emp values (7900,'james', 'clerk', 7698, '1981-12-03', 950, null, 30);

insert into emp values (7902,'ford', 'analyst', 7566, '1981-12-04', 3000, null, 20);

Select statements examples:

select * from emp;

select * from dept;

select ename, sal from emp;

select deptno, empno, mgr from emp;

select dname, loc from dept;

select * from emp where deptno=20;

select ename, sal from emp where sal>1000;

select empno, ename from emp where job='manager';

select ename from emp where job='clerk' and deptno=20;

select ename from emp where job='analyst' or job='salesman';

select * from emp where hiredate<='1981-9-30';

select ename, hiredate from emp;

select ename, date_format (hiredate, '%m/%d/%y') as formatteddate from emp;

select ename, date_format (hiredate, '%m/%d/%Y') as formatteddate from emp;

```sql
select ename, date_format (hiredate, '%d/%m/%Y') as formatteddate from emp;

select ename, date_format (hiredate, '%w, %M, %d, %Y') as formatteddate from emp;

select ename, date_format (hiredate, '%W, %M, %d, %Y') as formated_date from emp;

select ename, date_format (hiredate, '%W, %M, %d, %Y, %h:%i %p') as formated_date from emp;

Select ename from emp where job<>'manager';

Select ename from emp where empno=7369 or empno=7521 or empno=7839 or

empno=7934 or empno=7788;

Select ename from emp where empno IN (7369,7521,7839,7934,7788);

Select * from emp where deptno not in (10,30,40);

Select ename, sal from emp where sal between 1000 and 2000;

Select ename from emp where hiredate not between '30-june -81' and '31-DEC -81';

Select distinct job from emp;

Select ename from emp where comm is null;

Select ename, job from emp where mgr is null;

Select ename from emp where comm is not null;


Select ename from emp where ename like 's%';

Select ename from emp where ename like '%s';

Select ename from emp where ename like '_____';

Select ename from emp where ename like '_i';
```

Select ename, sal, sal * .1 from emp;

Select ename, sal, sal * .1 “PF” from emp;

Select empno, ename, sal from emp order by sal;

Select ename, hiredate as date_of_joining from emp order by date_of_joining desc;

Select deptno,job, ename, sal from emp order by deptno desc, sal desc;

select empno, ename, sal from emp order by 3;

Select ename, sal, sal * .1 “PF” , SAL * .5 “HRA”, SAL * .3 “DA” , sal+(sal * .5) + (sal * .3) +

(sal * .1) “GROSS” from emp order by 6;

Aggregate functions examples:

Select count(*) from emp;

Select count(distinct job) from emp;

Select sum(sal) from emp;

Select max(sal) from emp where job='salesman';

Select min(sal) from emp;

Select avg(sal), count(*) from emp where deptno=20;

Select deptno, count(*) from emp group by deptno;

Select deptno, sum(sal) from emp group by deptno;

Select job, count(*) from emp group by job order by count(*);

Select job, count(*) from emp group by job order by 2 desc;

```
Select job, sum(sal), avg(sal), max(sal), min(sal) from emp group by job;

Select job, avg(sal) from emp where job !=’manager’ group by job;


Select deptno, avg(sal) from emp group by deptno having count(*) >5;

Select job, max(sal) from emp group by job having max(sal) >=5000;

select * from emp;
```

coalesce() : used to **replace null values** with other values.

```
select ename, job, coalesce(mgr, 'not applicable') from emp;

select ename, job, coalesce(comm, 0) from emp;

select ename, job, ifnull(comm, 0) as commission from emp;

select ename, job from emp where isnull(comm);

update emp set comm=coalesce(comm, 2000) where empno=7839 ;

update emp set mgr=0000 where isnull(mgr);

select ename, job, if(isnull(mgr),0000, mgr) as manager from emp;

update emp set mgr=0000 where isnull(mgr);

select ename, sal, isnull(comm) as no_commission from emp;
```

Example program:

```
create table orders (

orderid int primary key auto_increment comment 'order number',

ordername varchar(20) not null comment 'order name',

ordervalue int not null comment 'order value',
orderdate date not null comment 'date on which order was placed',

check (ordervalue>0)

);

select * from orders;

insert into orders values (1,'cake', 2, '24-09-12');

insert into orders (ordername, ordervalue, orderdate) values('biscute', 10, '24-09-13');

insert into orders (ordername, ordervalue, orderdate) values('chocolate', 1, '24-09-20');

insert into orders (ordername, ordervalue, orderdate) values('bread', 2, '24-09-23');


show databases;

use classicmodels;

show tables;

select * from customers;
```

Day2 training Summary

1. Datatypes
2. Keys in SQL
3. CONSTRAINTS
4. Column alias
5. ERD and its components
6. Degree of relationships
7. Cardinality of relationships
8. Relational database model
9. DDL commands

DAY 3 TRAINING 13/9/24

Differenet ways

- STANDARD tcp/ip
- Named pipes
- Shared memory

MySQL default port : 3306

To get ERD :

database→ reverse engineering → store pswd and next → select required db →next and finish.

Database schema in MySQL :

MySQL schema refers to the structure of the database which includes tables, primary key, foreign key other constraints and indexes

Adding extra column to dept:

```
use varshinidb;

show tables;

alter table dept add email varchar(50);

select * from dept;

alter table dept add constraint unique_email unique(email) ;

update dept set email='accounting@mphasis.com' where deptno=10;

update dept set email='research@mphasis.com' where deptno=20;

update dept set email='sales@mphasis.com' where deptno=30;

update dept set email='operations@mphasis.com' where deptno=40;
```

creating indexes:

create index idx_dname on dept(dname);

create index idx_loc on dept(loc);

create index idx_email on dept(email);

drop index idx_email on dept;   // since by default index is created while adding unique constraint

select * from emp;

View : used to hide the actual data from end-user and display only required/ selected data (column) from a table.

create view user_view as select ename, job, sal from emp;

select * from user_view;

C:\ProgramData\MySQL\MySQL Server 8.0\Uploads\
Path for

Step to import tables from other db :

Select db where we have to import → right clk select table data import wizard →

browse the file path of the particular **exported table file** (.csv extn) of other db →

next and can alter the file name → click next until finish → file gets imported.

To check the imported table → refresh the database you'll find table.

While querying we need to specify the following:

1. Select: specifies the column to retrieve.
2. From: specifies the tables to query.
3. Where: filters rows based upon certain conditions.
4. Group by : groups rows with the same values.

5. Having: filters groups after group by.
6. Order by : orders the result set based on one or more columns (default is ascending)
7. Limit: restricts the number of rows returned.

```
use classicmodels;

select * from employees;

select firstname, lastname, concat_ws( ',' , firstname, lastname) from employees;

select firstname, lastname, concat_ws( ',', firstname, lastname) as fullname from employees;
```

concat_ws :

```
select concat_ws( ',', firstname, lastname) as fullname from employees;

select concat_ws( ' ', firstname, lastname) as fullname from employees;

select concat_ws( ' ', firstname, lastname) as fullname from employees order by fullname;

select concat_ws( ' ', firstname, lastname) as fullname from employees order by fullname desc;

select * from orderdetails;

select ordernumber 'orderno', sum(priceeach * quantityordered) total from orderdetails group by orderno;

select ordernumber 'orderno', sum(priceeach * quantityordered) total from orderdetails group by orderno having total>60000;

select ordernumber as 'orderno', sum(priceeach * quantityordered) from orderdetails group by orderno having sum(priceeach * quantityordered)>60000;
```

select o.ordernumber as 'o.no', o.productcode as 'pcode', o.quantityordered as 'qty', e.employeenumber, e.lastname, e.firstname from orderdetails o, employees e;

select * from orders, orderdetails;

use varshinidb;

create table **members**(

memberid int auto_increment,

name varchar(50),

primary key (memberid)

);

create table **committe**(

committeid int auto_increment,

name varchar(50),

primary key (committeid)

);

insert into members(name) values ('john'), ('jack'), ('king'), ('queen'), ('prince'), ('princess'), ('raja');

insert into committe (name) values ('john'),('king'),('prince'),('raja');

select * from members;

select * from committe;

Joins example :

select m.memberid, m.name as member, c.committeid, c.name as committe

```
    from members m inner join committe c on c.name = m.name;

select * from members, committe;

    insert into committe (name) values ('jill'),('joker');

select m.memberid, m.name as member, c.committeid, c.name as committe

    from members m left join committe c using(name);

select m.memberid, m.name as member, c.committeid, c.name as committe

    from members m left join committe c using(name) where c.committeid is null;

select m.memberid, m.name as member, c.committeid, c.name as committe

    from members m right join committe c using(name);

select m.memberid, m.name as member, c.committeid, c.name as committe

    from members m right join committe c on c.name=m.name;

select m.memberid, m.name as member, c.committeid, c.name as committe

    from members m right join committe c on c.name=m.name where
    m.memberid is null;

select m.memberid, m.name as member, c.committeid, c.name as committe

    from members m cross join committe c;
```

concatenation:

```
select ifnull(concat(m.lastname, ',' , m.firstname), 'mymanager') as manager,
    concat(e.lastname, ',' , e.firstname) as 'reportingperson' from employees e left
    join employees m on m.employeeNumber= e.reportsTo order by manager
    desc;

select * from customers order by city;

select c1.city, c1.customername, c2.customername from customers c1 inner
join customers c2 on c1.city=c2.city and c1.customername <>
c2.customername order by c1.city;
```

select m.name, c.name from members m cross join committe c order by m.name, c.name;

show tables;

select * from orders;

insert into orders values (**3**, 'milk', 5, '2024-09-13')as **new on duplicate key** update

ordername = new.ordername, ordervalue = new.ordervalue, orderdate=new.orderdate;

Ignore : ignore if already present and insert values if not present

insert **ignore** into orders values (**3**, 'milk', 5, '2024-09-13');

insert ignore into orders values (5, 'chocolate', 4, '2024-09-13');

insert into orders values (6, 'lollipop', 4, now());   //to include date and time

insert into orders values (7, 'popcorn', 8, now());

insert into orders values (8, 'gems', 8, current_date());   //to get only date

insert into orders values (9, 'cadberies', 9, utc_date());

insert into orders values (10, 'kitkat', 25, str_to_date('10/8/2024', '%d/%m/%Y'));

DAY 3 SUMMARY

Joins

Graphical representation of ER diagrams

Indexes

Views

Importing tables from other db

Duplicate key update

Various formats to display date

DAY 4 TRAINING 14/9/24

ACID theory(properties)

1. Atomicity
2. Consistency
3. Isolation - intermediate transaction is not visible.
4. Durability - after transaction, changes are permanently recorded even system crashes or power failures.

DBCC - Database consistency checker

```
use varshinidb;

select * from emp;

select hiredate from emp;

select empno, cast(hiredate as date) as dateofjoining from emp;

select empno, cast(hiredate as datetime) from emp;

select empno, cast( date_format(hiredate, '%Y-%M-%D')as char) as doj from
    emp;

select empno, cast(year(hiredate)as char) as dateofjoining from emp;
```

case and cast:

```
select empno, ename,

case

when cast(year(hiredate)as char)> 1980 then cast(year(hiredate)as char)

else'olderemployee' end as ename from emp;
```

Datatypes supported by Convert:

It will support binary, characters, date, datetime, decimal, signed and unsigned and we can also use ‘using’ keyword for character sets. This convert is MySQL SPECIFIC

Datatypes supported by cast:

It supports portability with other databases and It will support binary, characters, date, datetime, decimal, signed and unsigned. Signed and unsigned are used only for integers.

```
select empno, convert(year(hiredate) using utf8mb4) as dateofjoining from emp;

select empno, convert(year(hiredate), unsigned) as dateofjoining from emp;

select empno, convert(hiredate, unsigned) as dateofjoining from emp;

select empno, convert(hiredate using utf8mb3) as dateofjoining from emp;

select empno, convert(ename using utf16) as empname from emp;


select convert('mphasis' using utf16le);

select cast('12345' as unsigned);

select cast('123.45' as unsigned);

select cast(123.45 as signed);

select cast(123.45 as unsigned);

select convert(123.45, unsigned);

select convert(123.45, signed);

show tables;
```

```
rename table employees to person;

select * from person;

alter table person rename column employeenumber to empid;

describe person;
```

changing datatypes or properties:

```
alter table person change lastname lastname varchar(20);

alter table person change firstname fname varchar(20);

alter table person change email email varchar(50);

alter table person modify email varchar(50) null;

alter table person modify empid int not null primary key;


insert into person values (1006 , 'tom', 'jerry', 'x5678', null, 8, 1005, 'cartoon');

update person set email='johnwick@classicmodelcars.com' where empid= 1005;

update person set email='tomjerry@classicmodelcars.com' where empid= 1006 and email is null;

delete from person where empid=1006;

insert into person values (1008, 'chota', 'bheem', 'x5679', 'chota', 8, 1005, 'cartoon');

update person set email='chotabheem@classicmodelcars.com' where empid= 1008 and email is not null;

select * from orders where ordervalue = 2;

delete from orders where ordervalue='2';

desc orders;

delete from orders where orderid in (1,4);
```

select * from orders;

start transaction;

delete from orders where orderid=10;

commit;

rollback;

MySQL Functions:

1. **Aggregate functions**
2. **String functions**
3. **Date functions**
4. Comparison functions
5. Control flow
6. **Numeric functions**
7. Mathematical functions
8. Bit functions

Aggregate functions → Sum , avg, count, max, min

Comparison functions → coalesce, is null, greatest, least
Set operators → union, union all, except, except all, intersect, intersect all, intersect distinct

use classicmodels;
select * from employees, customers;
create table emp_cust as select lastname, firstname, email, jobtitle, customername, addressline1, city from employees, customers;
select * from emp_cust;
desc emp_cust;
create table empdata as select * from employees;

```
select * from empdata;
create table emp_structure as select * from employees where 1=0;
select * from emp_structure;
desc emp_structure;
select * from products;
use varshinidb;
create table mydata(
slno int primary key,
name char(10) not null,
marks int not null
);
insert into mydata values (1,'a', 10),(2,'a',20), (3,'b',30), (4,'c',40), (5,'c',50), (6,'b',90);
select * from mydata;
select name,avg(marks) from mydata group by name;

select * from products;
select * from productlines;
select * from orderdetails;
select productline, avg(buyprice) as averageprice from products group by
productline order by productline;
select count(*) as totalrows from products;
select productline, count(*) as totalcount from products group by productline
order by productline;
select productcode, sum(priceeach * quantityordered) total from orderdetails
group by productcode order by total desc;
select productcode, productname, sum(priceeach * quantityordered) total from
orderdetails inner join products using(productcode) group by productcode order
by total desc;
select max(buyprice) highestprice from products;
select min(buyprice) highestprice from products;
select * from customers;
select customername,city, coalesce(state, 'not provided')as state, country from
customers;
select customername, city, state, country from customers where isnull(state);
select customername, city, state, country from customers where state is not null;
select * from products;
select greatest(10,20,30), least(10,20,30,40);

select upper(customername), upper(contactlastname) from customers;
select lower(customername), lower(contactlastname) from customers;
select contactlastname, contactfirstname, concat(contactlastname, ' ',
contactfirstname) as fullname from customers;
select contactlastname, contactfirstname,city, state, concat_ws(','
,contactlastname, contactfirstname, city, state) as fullname from customers;
show character set;
select length('varshini') as numofcharacters;
use varshinidb;
select * from person;
update person set email= replace(email,'classicmodelcars','mphasis') where
empid>1000 or email like '%classicmodelcars.%';
desc person;

set sql_safe_updates =0;

select curdate(), current_time(), current_date(), now(), sysdate(), utc_date(),
utc_time();
select convert_tz( '2024-12-9 16:47:00', 'UTC', 'America/New_York');
use classicmodels;
select * from customers;
select * from employees;
select contactfirstname, contactlastname from customers union select
lastname, firstname from employees;
```

```sql
select lastname, firstname from employees union select contactfirstname,
contactlastname from customers;
select lastname, firstname from employees union all select contactfirstname,
contactlastname from customers;
select concat(lastname,' ', firstname) as fullname, 'employee' as contacttype from
employees
union
select concat(contactfirstname,' ', contactlastname), 'customers' as contacttype
from customers order by 2;
select firstname from employees except all select contactfirstname from
customers order by firstname;
select firstname from employees intersect select contactfirstname from
customers order by firstname;
select firstname from employees intersect all select contactfirstname from
customers order by firstname;
    select firstname from employees intersect distinct select contactfirstname
    from customers order by firstname;
```

DAY 5 TRAINING 16/9/24

In command prompt:

(c) Microsoft Corporation. All rights reserved.

C:\Users\varshini.r1>**cd \**

C:\\>**cd "Program Files**"   // after cd can clk tab to get options

C:\Program Files>**cd MySQL**

C:\Program Files\MySQL>**cd "MySQL Server 8.0"**

C:\Program Files\MySQL\MySQL Server 8.0>**cd bin**

To create a backup file of our database:

C:\Program Files\MySQL\MySQL Server 8.0\bin>**mysqldump -u root -p varshinidb > C:\Users\varshini.r1\Documents\uploads\varshini.sql**

Enter password: ****   → (root)

To create different user accounts:

create user 'tom'@'localhost' identified by 'tom@123';

create user 'jerry'@'localhost' identified by 'jerry@123';

Granting permissions:

grant all privileges on varshinidb.* to varshini@localhost;

grant select on varshinidb.* to tom@localhost;

grant select on varshinidb.* to jerry@localhost;

grant insert on varshinidb.dept to tom@localhost;

grant update on varshinidb.dept to tom@localhost; **// can update in cmd**

revoke all on varshinidb.* from jerry@localhost;

grant select on varshinidb.dept to jerry@localhost;

select * from varshinidb.emp;

grant select (ename, job, sal, comm), update (comm) on varshinidb.emp to jerry@localhost;

select * from emp;

To check grant and revoke commands, ex- be in tom account and run grant/revoke to jerry account commands in workbench and then come to cmd and try to execute.

In cmd:

To come out of present screen and change to other user account:

mysql> **exit**

bye

C:\Program Files\MySQL\MySQL Server 8.0\bin>**mysql -u jerry -p**

Enter password: ********* →(jerry@123)

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 23

Server version: 8.0.38 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its

```
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| performance_schema |
| varshinidb_schema |
+--------------------+
3 rows in set (0.00 sec)

mysql> select * from varshinidb.dept;
+--------+------------+----------+------------------------+
| deptno | dname      | loc      | email                  |
+--------+------------+----------+------------------------+
|     10 | accounting | new york | accounting@mphasis.com |
|     20 | research   | dallas   | research@mphasis.com   |
|     30 | sales      | chicago  | sales@mphasis.com      |
|     40 | operations | bostan   | operations@mphasis.com |
|     50 | manager    | paris    | manager@mphasis.com    |
+--------+------------+----------+-----------------------+
5 rows in set (0.00 sec)
```

```
mysql> show grants;
+----------------------------------------------------+
| Grants for jerry@localhost                    |
+----------------------------------------------------+
| GRANT USAGE ON *.* TO `jerry`@`localhost`          |
| GRANT SELECT ON `varshinidb`.* TO `jerry`@`localhost` |
+----------------------------------------------------+
2 rows in set (0.00 sec)
```

// After revoking privileges, checking again →

1. mysql> **show grants;**

   ```
   +-----------------------------------------+
   | Grants for jerry@localhost           |
   +-----------------------------------------+
   | GRANT USAGE ON *.* TO `jerry`@`localhost` |
   +-----------------------------------------+
   1 row in set (0.00 sec)
   ```

2. mysql> **select * from varshinidb.dept;**

```
ERROR 1142 (42000): SELECT command denied to user 'jerry'@'localhost' for
table 'dept'
```

After granting checking again:

```
mysql> select * from varshinidb.dept;
+--------+------------+----------+------------------------+
| deptno | dname      | loc      | email                  |
+--------+------------+----------+------------------------+
|     10 | accounting | new york | accounting@mphasis.com |
|     20 | research   | dallas   | research@mphasis.com   |
|     30 | sales      | chicago  | sales@mphasis.com      |
|     40 | operations | bostan   | operations@mphasis.com |
|     50 | manager    | paris    | manager@mphasis.com    |
+--------+------------+----------+------------------------+
5 rows in set (0.00 sec)

mysql> select * from varshinidb.emp;
ERROR 1142 (42000): SELECT command denied to user 'jerry'@'localhost' for table
'emp'
mysql> use varshinidb;
Database changed
mysql> show tables;
+----------------------+
| Tables_in_varshinidb |
+---------------------+
| dept                |
+---------------------+
1 row in set (0.00 sec)
```

mysql> **show tables;**

```
+---------------------+
| Tables_in_varshinidb |
+---------------------+
| dept                |
| emp                 |
+---------------------+
2 rows in set (0.00 sec)
```

mysql> **select * from varshinidb.emp;**

```
ERROR 1143 (42000): SELECT command denied to user 'jerry'@'localhost' for column 'empno' in table 'emp'
```

mysql> **select ename, job, sal, comm from varshinidb.emp;**

```
+--------+----------+---------+---------+
| ename  | job      | sal     | comm    |
+--------+----------+---------+---------+
| smith  | clerk    |  800.00 |    NULL |
| allen  | salesman | 1600.00 |  300.00 |
| ward   | salesman  | 1250.00 |  500.00 |
| jones  | manager   | 2975.00 |    NULL |
| martin | salesman  | 1260.00 | 1400.00 |
| blake  | manager   | 2850.00 |    NULL |
| clark  | analyst   | 3000.00 |    NULL |
| king   | president | 5000.00 | 2000.00 |
| turner | salesman  | 1500.00 |    NULL |
| adams  | clerk     | 1100.00 |    NULL |
| james  | clerk     |  950.00 |    NULL |
| ford   | analyst   | 3000.00 |    NULL |
+--------+-----------+---------+---------+
12 rows in set (0.00 sec)
```

mysql> **update dept set comm= 1000 where ename='adams';**

```
ERROR 1142 (42000): UPDATE command denied to user 'jerry'@'localhost' for table 'dept'
```

mysql> **update emp set comm= 1000 where ename='adams';**

```
Query OK, 1 row affected (0.04 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

mysql> **select ename, job, sal, comm from varshinidb.emp;**

```
+--------+-----------+---------+---------+
| ename  | job       | sal     | comm    |
+--------+-----------+---------+---------+
| smith  | clerk     |  800.00 |    NULL |
| allen  | salesman  | 1600.00 |  300.00 |
| ward   | salesman  | 1250.00 |  500.00 |
| jones  | manager   | 2975.00 |    NULL |
| martin | salesman  | 1260.00 | 1400.00 |
| blake  | manager   | 2850.00 |    NULL |
| clark  | analyst   | 3000.00 |    NULL |
| king   | president | 5000.00 | 2000.00 |
| turner | salesman  | 1500.00 |    NULL |
| adams  | clerk     | 1100.00 | 1000.00 |
| james  | clerk     |  950.00 |    NULL |
| ford   | analyst   | 3000.00 |    NULL |
+--------+-----------+---------+---------+
12 rows in set (0.00 sec)
```

In workbench:

```
alter user 'tom'@'localhost' identified by 'tom@123' account lock;
select user, host, account_locked from mysql.user;
alter user 'tom'@'localhost' account unlock;
```

In cmd:

```
mysql> exit
Bye
```

```
C:\Program Files\MySQL\MySQL Server 8.0\bin>mysql -u tom -p
Enter password: *******
ERROR 3118 (HY000): Access denied for user 'tom'@'localhost'. Account is locked.
```

After granting:

```
C:\Program Files\MySQL\MySQL Server 8.0\bin>mysql -u tom -p
Enter password: *******
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 27
Server version: 8.0.38 MySQL Community Server - GPL
Copyright (c) 2000, 2024, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| performance_schema |
| varshinidb         |
+--------------------+
3 rows in set (0.00 sec)


mysql> use varshinidb;
Database changed
mysql> show tables;
+----------------------+
| Tables_in_varshinidb |
+----------------------+
| committe             |
| dept                 |
| emp                  |
| members              |
| mydata               |
| orders               |
| person               |
| user_view            |
+----------------------+
8 rows in set (0.00 sec)


mysql> select * from dept;
+--------+------------+----------+------------------------+
| deptno | dname      | loc      | email                  |
+--------+------------+----------+------------------------+
|    10 | accounting | new york | accounting@mphasis.com |
|    20 | research   | dallas   | research@mphasis.com   |
|    30 | sales      | chicago  | sales@mphasis.com      |
|    40 | operations | bostan   | operations@mphasis.com |
+--------+------------+----------+------------------------+
4 rows in set (0.00 sec)

mysql> insert into dept values (50, 'manager', 'India', 'manager@mphasis.com');
ERROR 1142 (42000): INSERT command denied to user 'tom'@'localhost' for table 'dept'
mysql> insert into dept values (50, 'manager', 'India', 'manager@mphasis.com');
Query OK, 1 row affected (0.04 sec)
```

mysql> **select * from dept;**

```
+--------+------------+----------+------------------------+
| deptno | dname      | loc      | email                  |
+--------+------------+----------+------------------------+
|    10 | accounting | new york | accounting@mphasis.com |
|    20 | research   | dallas   | research@mphasis.com   |
|    30 | sales      | chicago  | sales@mphasis.com      |
|    40 | operations | bostan   | operations@mphasis.com |
|    50 | manager    | India    | manager@mphasis.com    |
+--------+------------+----------+------------------------+
5 rows in set (0.00 sec)

mysql> update dept set loc='paris' where deptno=50;
ERROR 1142 (42000): UPDATE command denied to user 'tom'@'localhost' for table 'dept'
mysql> update dept set loc='paris' where deptno=50;
Query OK, 1 row affected (0.09 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from dept;
+--------+------------+----------+------------------------+
| deptno | dname      | loc      | email                  |
+--------+------------+----------+------------------------+
|     10 | accounting | new york | accounting@mphasis.com |
|     20 | research   | dallas   | research@mphasis.com   |
|     30 | sales      | chicago  | sales@mphasis.com      |
|     40 | operations | bostan   | operations@mphasis.com |
|     50 | manager    | paris    | manager@mphasis.com    |
+--------+------------+----------+------------------------+
5 rows in set (0.00 sec)
```

In workbench:

```
create role datawriter@localhost;
```

```sql
show grants for datawriter@localhost;
grant select, insert, update, delete on varshinidb.* to datawriter@localhost;
show grants for datawriter@localhost;
grant datawriter@localhost to tom@localhost;
show grants for tom@localhost;
show grants for tom@localhost using datawriter@localhost;

use classicmodels;
select * from payments;
select max(amount) from payments;
select customerNumber, checkNumber, amount from payments
where amount=(select max(amount) from payments);
select customerNumber, checkNumber, amount from payments
where amount>(select avg(amount) from payments);
select * from customers;
select * from orders;
select distinct customernumber from orders;
select customername from customers where customernumber not in (select
distinct customernumber from orders);
select customername, customerNumber from customers
where customernumber not in (select distinct customernumber from orders);
select * from orders where customerNumber=125;
select * from orderdetails;
select max(items), min(items), floor(avg(items))
from (select ordernumber, count(ordernumber) as items from orderdetails
group by ordernumber) as lineitems;


select ordernumber, count(ordernumber) as items from orderdetails group by ordernumber;
select productname, buyprice from products p1;


select productname, buyprice from products p1 where buyprice >
(select avg(buyprice) from products where productline=p1.productline);
select * from products;
select * from productlines;
select * from orderdetails;
select ordernumber, sum(priceeach*quantityOrdered) as totalamount from orderdetails group by orderNumber;
select ordernumber, sum(priceeach*quantityOrdered) as totalamount from orderdetails
inner join orders using (ordernumber) group by orderNumber having totalamount > 60000;


select customernumber, customername from customers where exists
(select ordernumber, sum(priceeach*quantityOrdered) as totalamount from orderdetails
inner join orders using (ordernumber) where customerNumber= customers.customerNumber
group by orderNumber having totalamount > 60000);


select customernumber, customername from customers where not exists
(select ordernumber, sum(priceeach*quantityOrdered) as totalamount from
orderdetails
inner join orders using (ordernumber) where customerNumber=
customers.customerNumber
group by orderNumber having totalamount > 60000);


select customernumber, customername from customers where exists
(select ordernumber, sum(priceeach*quantityOrdered) as totalamount from
orderdetails
inner join orders using (ordernumber) where customerNumber=
customers.customerNumber
group by orderNumber having totalamount < 60000);


select customername, checknumber, paymentdate, amount from customers
inner join payments using (customernumber);
create view customerpayments as select customername, checknumber,
paymentdate, amount from customers
inner join payments using (customernumber);
select * from customerpayments;
create view dayofweek (day) as select 'Mon' union select 'Tue'
union select 'Wed' union select 'Thur' union select 'Fri' union select 'Sat' union
select 'Sun' ;
select * from dayofweek;


create view bigsalesorder as

select customernumber, customername from customers where exists

(select ordernumber, sum(priceeach*quantityOrdered) as totalamount from orderdetails

inner join orders using (ordernumber) where customerNumber= customers.customerNumber

group by orderNumber having totalamount > 60000);

select * from bigsalesorder;


select ordernumber, sum(priceeach*quantityOrdered) as totalamount from orderdetails group by orderNumber order by 2;

create view salesperorder as select ordernumber, sum(priceeach*quantityOrdered) as totalamount from orderdetails group by orderNumber order by 2;

select * from salesperorder;

select ordernumber, round(totalamount, 2)as total from salesperorder where totalamount>60000;

create view bigsales as select ordernumber, round(totalamount, 2)as total from salesperorder where totalamount>60000;

select * from bigsales;


show tables;

show full tables where table_type='VIEW';
```

RENAME TABLE salesperorder to orderpersales;

drop view dayofweek;

DAY 6 TRAINING   17/09/2024

Creating & Dropping Index in Emp & Dept table:-

```
use classicmodels;

show tables;

show indexes from employees;

select * from employees;

create index jobtitle on employees(jobtitle);

select employeenumber, lastname, firstname from employees where jobtitle='sales rep';

explain select employeenumber, lastname, firstname from employees where jobtitle='sales rep';

drop index jobtitle on employees;

drop index `PRIMARY` on employees;

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

use testdb;

show tables;

drop index `PRIMARY` on dept;

desc dept;

show indexes from dept;
```

```
select * from dept;

insert into dept values (10,'accounting', 'new york','accounting@mphasis.com');

insert into dept values (20,'research', 'dallas', 'reasearch@mphasis.com');

insert into dept values (30,'sales', 'chicago', 'sales@mphasis.com');

insert into dept values (40,'operations', 'bostan', 'operations@mphasis.com');

alter table dept add primary key (deptno);

alter table dept add primary key (deptno, email);

select email from dept where dname='sales';

explain select email from dept where dname='sales';

select dname from dept where email='sales@mphasis.com';

explain select dname from dept where email='sales@mphasis.com';

create index d_index_name on dept(dname);

show indexes from dept;

drop index `PRIMARY` on dept;

alter table dept add constraint email_constraint unique key(email);

show index in dept from testdb;

show keys from dept in testdb;
```

```
use classicmodels;

check table customers, orders, orderdetails, payments, productlines;

repair table customers, orders, orderdetails, payments, productlines;
```

```
C:\Program Files\MySQL\MySQL Server 8.0\bin>mysqlcheck -u root -p --check classicmodels
```

```
C:\Program Files\MySQL\MySQL Server 8.0\bin>mysqlcheck -u root -p --repair --all-databases
```

```
C:\Program Files\MySQL\MySQL Server 8.0\bin>mysqlcheck -u root -p --check --all-databases
```

```
mysql> check table classicmodels.orders \G;
```

```
mysql> check table classicmodels.orders;
```

```
use classicmodels;

select * from customers;

select customername, city, state, postalcode, country from customers order by customerName;
```

Creating stored procedure:-

1. delimiter $$

```
    create procedure getcustomers()

    begin

        select customername, city, state, postalcode, country from customers order by customerName;

    end $$

    delimiter ;

    call getcustomers();
```

```
        show procedure status;

        show procedure status where db= 'classicmodels';

        CALL `varshinidb`.`getempdata` ();
```

2. delimiter $$

```
        create procedure getdept()

        Begin

        select deptno, loc from dept order by deptno;

        end $$

        delimiter ;

        call getdept();
```

--

```
show warnings;

use classicmodels;

drop procedure getdept;

drop procedure if exists getdept;
```

--

```
select * from orders;

select count(*) from orders;
```

3. delimiter $$

```
create procedure gettotalorders()

begin
```

```
        declare totalorder int default 0;

        select count(*) into totalorder from orders;

  select totalorder;

end $$

delimiter ;

call gettotalorders();

select * from customers;

select customernumber, creditlimit from customers where creditLimit > 50000
order by creditLimit desc;
```

4. delimiter $$

```
create procedure getcustomerlevel(

        in pcustomernumber int, out pcustomerlevel varchar(20))

begin

        declare credit decimal(10,2) default 0;

        select creditlimit into credit from customers

  where customernumber = pcustomernumber;

  if credit > 50000 then

                set pcustomerlevel='diamond';

  end if;

end $$

delimiter ;

call getcustomerlevel(128, @level);
```

```
select @level;

call getcustomerlevel(103, @level);

select @level;
```

Altering stored procedure:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `getcustomerlevel`(
        in pcustomernumber int, out pcustomerlevel varchar(20))
begin
        declare credit decimal(10,2) default 0;
        select creditlimit into credit from customers
  where customernumber = pcustomernumber;
  if credit > 50000 then
                set pcustomerlevel='diamond';
        elseif credit <=50000 and credit >20000 then
                set pcustomerlevel= 'silver';
  else
                set pcustomerlevel = 'bronze';
  end if;
end
call getcustomerlevel(223, @level);
```

Using case:

```
delimiter $$
```

```
create procedure getcustomershipping(
        in pcustomernumber int, out pshipping varchar(100))
begin
        declare customercountry varchar(50);
        select country into customercountry from customers
  where customernumber = pcustomernumber;
  case customercountry
                when 'USA' then
                        set pshipping = 'shipping will be delayed by 1 week';
                when 'Canada' then
                        set pshipping = 'shipping will be delayed by 2 week';
                when 'Australia' then
                        set pshipping = 'shipping will be delayed by 2 days';
                else
                        set pshipping = 'shipping will be done immediately';
        end case;
end $$
delimiter ;
call getcustomershipping(124, @shipping);
select @shipping;
call getcustomershipping(114, @shipping);
select @shipping;
call getcustomershipping(103, @shipping);
```

```
select @shipping;
```

---

creating table and inserting using stored procedures:

```
use varshinidb;
 create table calender(
 date date primary key,
 month int not null,
 quarter int not null,
 year int not null
 );
 select * from calender;

 delimiter $$
create procedure filldates(
        in startdate date, in enddate date)
begin
        declare currentdate date default startdate;
    insert_date: loop
                -- insert increase date by one day
        set currentdate=date_add(currentdate, interval 1 day);
        -- come out of the loop if the current date is after the end date
        if currentdate > enddate then
```

```
                leave insert_date;
           end if;

    -- inserting the data into the calender table
    insert into calender (date, month, quarter, year)
    values (currentdate, month(currentdate), quarter(currentdate),
year(currentdate));
      end loop;
end $$
delimiter ;

call filldates('2024-09-01', '2024-09-30');
select count(*) from calender;
select * from calender;

set autocommit = off;     -- or
set autocommit = 0;
use varshinidb;
show tables;
select * from dept;
start transaction;
insert into dept values (60, 'electrical', 'japan', 'electrical@mphasis.com');
insert into dept values (70, 'mechanical', 'german',
'mechanical@mphasis.com');
```

```
commit;



set autocommit = off;

start transaction;

insert into dept values (80, 'civil', 'london', 'civil@mphasis.com');

insert into dept values (90, 'chemical', 'china', 'chemical@mphasis.com');

select * from dept;

insert into dept values (100, 'banking', 'USA', 'banking@mphasis.com');

rollback;


CREATE TABLE accounts (

   account_id INT AUTO_INCREMENT  PRIMARY KEY ,

   account_holder VARCHAR(255) NOT NULL,

   balance DECIMAL(10, 2) NOT NULL

);



CREATE TABLE transactions (

   transaction_id INT AUTO_INCREMENT PRIMARY KEY,

   account_id INT NOT NULL,

   amount DECIMAL(10, 2) NOT NULL,

   transaction_type ENUM('DEPOSIT', 'WITHDRAWAL') NOT NULL,
   FOREIGN KEY (account_id) REFERENCES accounts(account_id)
);


INSERT INTO accounts (account_holder, balance) VALUES ('raja', 1000.00),
('rani', 500.00);

DELIMITER //
CREATE PROCEDURE transfer(
   IN sender_id INT,
   IN receiver_id INT,
   IN amount DECIMAL(10,2)
)
BEGIN
   DECLARE rollback_message VARCHAR(255) DEFAULT 'Transaction rolled
back: Insufficient funds';
   DECLARE commit_message VARCHAR(255) DEFAULT 'Transaction
committed successfully';

         -- Start the transaction
   START TRANSACTION;

   -- Attempt to debit money from account 1
```

```
    UPDATE accounts SET balance = balance - amount WHERE account_id = sender_id;

    -- Attempt to credit money to account 2
    UPDATE accounts SET balance = balance + amount WHERE account_id = receiver_id;

    -- Check if there are sufficient funds in account 1
    -- Simulate a condition where there are insufficient funds
    IF (SELECT balance FROM accounts WHERE account_id = sender_id) < 0 THEN
        -- Roll back the transaction if there are insufficient funds
        ROLLBACK;
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = rollback_message;
    ELSE
        -- Log the transactions if there are sufficient funds
        INSERT INTO transactions (account_id, amount, transaction_type) VALUES (sender_id, -amount, 'WITHDRAWAL');
        INSERT INTO transactions (account_id, amount, transaction_type) VALUES (receiver_id, amount, 'DEPOSIT');

        -- Commit the transaction
        COMMIT;
        SELECT commit_message AS 'Result';
    END IF;
END //
DELIMITER ;



SELECT * FROM accounts;
call transfer(1, 2, 1000);
show warnings;
SELECT * FROM transactions;
lock table accounts read;
insert into accounts values(3, 'minister', 500.20);
insert into accounts values(4, 'soldier', 10000.99);
unlock table;
start transaction;
insert into accounts values(5, 'king', 2000.20);
insert into accounts values(6, 'queen', 5000.99);
savepoint sp1;

insert into accounts values(9, 'prince', 2500);
insert into accounts values(10, 'princess', 5600);
savepoint sp2;
rollback to savepoint sp1;
commit;
```