PRIME COLLEGE

Khusibun, Kathmandu

LAB REPORT

OF

ADVANCED DATABASE

Submitted by:

Name: Poem Maharjan

Faculty: BSc.CSIT

Semester: 8th

Submitted to:

Radha Krishna Gajurel

CONTENTS

Lab 1
Database Operation

1. Create Database.

CREATE DATABASE poem;

#	Time	Action	Message	Duration / Fetch
1	15:35:55	create database poem	1 row(s) affected	0.015 sec

2. Use the created database.

USE poem;

#	Time	Action	Message	Duration / Fetch
1	15:37:53	use poem	0 row(s) affected	0.000 sec

3. Create Table.

a. Create table normally:

```
CREATE TABLE student (
   id INT PRIMARY KEY,
   student_name VARCHAR(100) NOT NULL
);
```

#	Time	Action	Message	Duration / Fetch
1	15:41:32	CREATE TABLE student (id INT PRIMARY KEY,...	0 row(s) affected	0.032 sec

b. Create table using another table:

CREATE TABLE student2 AS

SELECT * FROM student;

#	Time	Action	Message	Duration / Fetch
1	15:42:03	CREATE TABLE student2 AS SELECT * FROM stu...	0 row(s) affected Records: 0 Duplicates: 0 Warning...	0.047 sec

4. Alter table

a. Add column:

ALTER TABLE student ADD roll_no DECIMAL(10,2);

#	Time	Action	Message	Duration / Fetch
1	15:43:33	ALTER TABLE student ADD roll_no DECIMAL(10,2)	0 row(s) affected Records: 0 Duplicates: 0 Warning...	0.047 sec

b. Drop column:

ALTER TABLE student2 DROP COLUMN roll_no ;

#	Time	Action	Message	Duration / Fetch
1	15:47:14	ALTER TABLE student2 DROP COLUMN roll_no	0 row(s) affected Records: 0 Duplicates: 0 Warning...	0.078 sec

c. Alter/Modify column:

ALTER TABLE student1 MODIFY COLUMN roll_no SMALLINT;

#	Time	Action	Message	Duration / Fetch
1	15:47:46	ALTER TABLE student1 MODIFY COLUMN roll_no ...	0 row(s) affected Records: 0 Duplicates: 0 Warning...	0.078 sec

5. Constraints.

a. NOT NULL: Ensures that a column cannot have a NULL value.

i. Apply NOT NULL while creating table:

```
CREATE TABLE employee (
   id INT PRIMARY KEY,
   name VARCHAR(100) NOT NULL
);
```

#	Time	Action	Message	Duration / Fetch
1	15:48:20	CREATE TABLE employee (id INT PRIMARY KE...	0 row(s) affected	0.032 sec

ii. Apply NOT NULL Using ALTER TABLE:

ALTER TABLE employee MODIFY name VARCHAR(100) NOT NULL;

#	Time	Action	Message	Duration / Fetch
1	15:49:36	ALTER TABLE employee MODIFY name VARCHAR...	0 row(s) affected Records: 0 Duplicates: 0 Warning...	0.031 sec

b. **UNIQUE:** Ensures that all values in a column are different.

i. **Unique Constraint While Creating Table:**

```
CREATE TABLE worker (
    id INT UNIQUE,
    name VARCHAR(100)
);
```

#	Time	Action	Message	Duration / Fetch
1	15:50:13	CREATE TABLE worker (id INT UNIQUE, nam...	0 row(s) affected	0.062 sec

ii. **Add Unique Constraint Using ALTER TABLE:**

```
ALTER TABLE worker ADD CONSTRAINT unique_name UNIQUE (Name);
```

#	Time	Action	Message	Duration / Fetch
1	15:50:58	ALTER TABLE worker ADD CONSTRAINT unique_...	0 row(s) affected Records: 0 Duplicates: 0 Warning...	0.032 sec

iii. **Drop a Unique Constraint:**

```
ALTER TABLE worker DROP CONSTRAINT unique_name;
```

#	Time	Action	Message	Duration / Fetch
1	15:51:54	ALTER TABLE worker DROP CONSTRAINT uniqu...	0 row(s) affected Records: 0 Duplicates: 0 Warning...	0.031 sec

c. **PRIMARY KEY:** A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table.

i. **Primary key constraint:**

```
CREATE TABLE school (
    id INT PRIMARY KEY,
    name VARCHAR(255)
);
```

#	Time	Action	Message	Duration / Fetch
1	15:57:13	CREATE TABLE school (id INT PRI...	0 row(s) affected	0.047 sec

ii. Primary Key on Multiple Columns:

```
CREATE TABLE OrderDetails (
    OrderID INT,
    ProductID INT,
    Quantity INT,
    PRIMARY KEY (OrderID, ProductID)
);
```

#	Time	Action	Message	Duration / Fetch
1	15:57:37	CREATE TABLE OrderDetails (OrderI...	0 row(s) affected	0.031 sec

iii. Add Primary Key on ALTER TABLE:

```
ALTER TABLE OrderDetails ADD PRIMARY KEY (OrderID);
```

#	Time	Action	Message	Duration / Fetch
1	16:02:05	ALTER TABLE OrderDetails ADD PRIM...	0 row(s) affected Records: 0 Duplicates:...	0.063 sec

iv. Drop a PK constraint:

```
ALTER TABLE OrderDetails DROP PRIMARY KEY;
```

#	Time	Action	Message	Duration / Fetch
1	16:03:21	ALTER TABLE OrderDetails DROP PRI...	0 row(s) affected Records: 0 Duplicates:...	0.063 sec

d. FOREIGN KEY: Uniquely identifies a row/record in another table.

i. Foreign key constraint:

```
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY, CustomerName VARCHAR(100)
);
```

#	Time	Action	Message	Duration / Fetch
1	16:03:47	CREATE TABLE Customers (Custom...	0 row(s) affected	0.031 sec

```
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    CustomerID INT,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
```

#	Time	Action	Message	Duration / Fetch
1	16:05:44	CREATE TABLE Orders (OrderID IN...	0 row(s) affected	0.093 sec

ii. Add Foreign Key on ALTER TABLE:

```
ALTER TABLE Orders
ADD CONSTRAINT fk_customer
FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID);
```

#	Time	Action	Message	Duration / Fetch
1	16:06:17	ALTER TABLE Orders ADD CONSTRA...	0 row(s) affected Records: 0 Duplicates:...	0.094 sec

iii. Drop a Foreign Key Constraint:

```
ALTER TABLE Orders DROP CONSTRAINT fk_customer;
```

#	Time	Action	Message	Duration / Fetch
1	16:06:38	ALTER TABLE Orders DROP CONSTR...	0 row(s) affected Records: 0 Duplicates:...	0.016 sec

e. DEFAULT Constraint: Sets up a default value for the field.

i. SQL DEFAULT Constraint:

```
CREATE TABLE company (
    EmployeeID INT PRIMARY KEY,
    Name VARCHAR(100),
    Department VARCHAR(50) DEFAULT 'General'
);
```

#	Time	Action	Message	Duration / Fetch
1	16:07:33	CREATE TABLE company (Employe...	0 row(s) affected	0.031 sec

ii. Add DEFAULT Using ALTER TABLE:

ALTER TABLE company ALTER COLUMN Department SET DEFAULT 'HR';

#	Time	Action	Message	Duration / Fetch
1	16:08:51	ALTER TABLE company ALTER COLU...	0 row(s) affected Records: 0 Duplicates:...	0.015 sec

iii. Drop a DEFAULT Constraint:

ALTER TABLE company ALTER COLUMN Department DROP DEFAULT;

#	Time	Action	Message	Duration / Fetch
1	16:09:25	ALTER TABLE company ALTER COLU...	0 row(s) affected Records: 0 Duplicates:...	0.016 sec

f. CHECK Constraint: This is used to enforce rules on column values to ensure data integrity.

i. SQL CHECK Constraint (While creating a Table):

```
CREATE TABLE emp (
    EmployeeID INT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Age INT CHECK (Age >= 18),
    Salary DECIMAL(10,2) CHECK (Salary > 0)
);
```

#	Time	Action	Message	Duration / Fetch
1	16:09:43	CREATE TABLE emp (EmployeeID I...	0 row(s) affected	0.031 sec

ii. CHECK Constraint for Multiple Columns:

```
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    Quantity INT,
    Price DECIMAL(10,2),
    CHECK (Quantity > 0 AND Price > 0)
);
```

#	Time	Action	Message	Duration / Fetch
1	16:10:12	CREATE TABLE Orders (OrderID IN...	0 row(s) affected	0.031 sec

iii. SQL Check on ALTER Table:

ALTER TABLE emp ADD CONSTRAINT chk_age CHECK (Age >= 18);

#	Time	Action	Message	Duration / Fetch
1	16:10:35	ALTER TABLE emp ADD CONSTRAIN...	0 row(s) affected Records: 0 Duplicates:...	0.094 sec

iv. Drop a CHECK Constraint:

ALTER TABLE emp DROP CONSTRAINT chk_age;

#	Time	Action	Message	Duration / Fetch
1	16:10:49	ALTER TABLE emp DROP CONSTRAI...	0 row(s) affected Records: 0 Duplicates:...	0.015 sec

6. SQL Index.

a. Create index:

CREATE INDEX idx_name ON emp(Name);

#	Time	Action	Message	Duration / Fetch
1	16:11:08	CREATE INDEX idx_name ON emp(Na...	0 row(s) affected Records: 0 Duplicates:...	0.031 sec

b. Drop index:

DROP INDEX idx_name ON emp;

#	Time	Action	Message	Duration / Fetch
1	16:11:24	DROP INDEX idx_name ON emp	0 row(s) affected Records: 0 Duplicates:...	0.031 sec

7. Auto Increment Field.

```
CREATE TABLE fruit (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100)
);
```

#	Time	Action	Message	Duration / Fetch
1	16:11:37	CREATE TABLE fruit (id INT AUTO_I...	0 row(s) affected	0.031 sec

8. Insert Query.

INSERT INTO fruit (Name) VALUES ('name');

#	Time	Action	Message	Duration / Fetch
1	16:11:52	INSERT INTO fruit (Name) VALUES ('na...	1 row(s) affected	0.016 sec

9. Update Table.

UPDATE fruit SET Name = 'name' WHERE id = 1;

#	Time	Action	Message	Duration / Fetch
1	16:12:08	UPDATE fruit SET Name = 'name' WHE...	0 row(s) affected Rows matched: 1 Cha...	0.000 sec

10. SQL Delete.

DELETE FROM fruit WHERE Name = 'name';

#	Time	Action	Message	Duration / Fetch
1	16:15:25	DELETE FROM fruit WHERE Nam...	1 row(s) affected	0.000 sec

11. SQL truncate table.

TRUNCATE TABLE fruit;

#	Time	Action	Message	Duration / Fetch
1	16:15:44	TRUNCATE TABLE fruit	0 row(s) affected	0.078 sec

12. DROP table.

DROP TABLE fruit;

#	Time	Action	Message	Duration / Fetch
1	16:16:04	DROP TABLE fruit	0 row(s) affected	0.016 sec

Lab 2
Data Control Language

1. **Creating User**

CREATE USER 'username'@'localhost' IDENTIFIED BY 'password';

#	Time	Action	Message	Duration / Fetch
1	16:25:48	CREATE USER 'username'@'local...	0 row(s) affected	0.032 sec

2. **Grant privilege**

user	host
mysql.infoschema	localhost
mysql.session	localhost
mysql.sys	localhost
root	localhost
username	localhost

- Individual privilege like CREATE, DELETE, INSERT, UPDATE, SELECT, etc.

GRANT READ ON poem.student TO 'username'@'localhost';

#	Time	Action	Message	Duration / Fetch
1	16:34:07	GRANT SELECT, INSERT, UPDA...	0 row(s) affected	0.000 sec

- Provide all the privileges to the created user.

GRANT ALL PRIVILEGES ON *.* TO 'username'@'localhost';

#	Time	Action	Message	Duration / Fetch
1	16:34:31	GRANT ALL PRIVILEGES ON *.* ...	0 row(s) affected	0.016 sec

3. **Connect to User**

First, exit from the DBMS and then in the command prompt, login using credentials.

```
C:\Users\HP>mysql -u username -p
Enter password: ********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 37
Server version: 8.0.42 MySQL Community Server - GPL

Copyright (c) 2000, 2025, 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>
```

4. Remove permission

Remove the permission to create from all the tables for the user ‘<username>’;

REVOKE READ ON *.* FROM ‘username’@’localhost’;

#	Time	Action	Message	Duration / Fetch
1	16:40:46	REVOKE SELECT ON *.* FROM 'u...	0 row(s) affected	0.016 sec

Grants for username@localhost
GRANT INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, SHUTDOWN, PROCES...
GRANT APPLICATION_PASSWORD_ADMIN,AUDIT_ABORT_EXEMPT,AUDIT_ADMIN,...
GRANT SELECT, INSERT, UPDATE ON `mysql`.`user` TO `username`@`localhost`

As we can see, the ***READ*** permission has been revoked.

5. Drop User

DROP USER ‘username’@‘localhost’;

#	Time	Action	Message	Duration / Fetch
1	16:41:38	DROP USER 'username'@'localhost'	0 row(s) affected	0.015 sec

Lab 3

Active Database and Triggers

Step 1: Create Tables

```
CREATE TABLE main_table (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255),
    active BOOLEAN DEFAULT TRUE
);
CREATE TABLE active_table (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255),
    active BOOLEAN DEFAULT TRUE
);
```

1	07:08:52	CREATE TABLE main_table (	id ...	0 row(s) affected	0.141 sec
2	07:08:52	CREATE TABLE active_table (	i...	0 row(s) affected	0.016 sec

Step 2: Define Trigger

```
DELIMITER //
CREATE TRIGGER update_active_flag
AFTER UPDATE ON main_table
FOR EACH ROW
BEGIN
    IF OLD.active != NEW.active THEN
```

```
        UPDATE active_table
        SET active = NEW.active
        WHERE id = NEW.id;
    END IF;
END //
DELIMITER ;
```

#	Time	Action	Message	Duration / Fetch
1	07:11:17	CREATE TRIGGER update_active...	0 row(s) affected	0.047 sec

Step 3: Insert Data and Update

Insert some data into the main_table.

INSERT INTO main_table (name) VALUES ('Item 1'), ('Item 2');

#	Time	Action	Message	Duration / Fetch
1	07:11:51	INSERT INTO main_table (name) ...	2 row(s) affected Records: 2 Dupli...	0.015 sec

Insert corresponding data into the active_table.

INSERT INTO active_table (name) VALUES ('Item 1'), ('Item 2');

#	Time	Action	Message	Duration / Fetch
1	07:12:18	INSERT INTO active_table (name)...	2 row(s) affected Records: 2 Dupli...	0.016 sec

Update the active flag in main_table

UPDATE main_table SET active = FALSE WHERE id = 1;

#	Time	Action	Message	Duration / Fetch
1	07:12:34	UPDATE main_table SET active = ...	1 row(s) affected Rows matched: 1...	0.031 sec

Step 4: View Data

SELECT * FROM active_table;

id	name	active
1	Item 1	0
2	Item 2	1
NULL	NULL	NULL

Step 5: Clean Up (Drop the tables)

DROP TABLE IF EXISTS main_table;

DROP TABLE IF EXISTS active_table;

2	07:14:04	DROP TABLE IF EXISTS main_t...	0 row(s) affected	0.031 sec
3	07:14:04	DROP TABLE IF EXISTS active_...	0 row(s) affected	0.031 sec

Lab 4

Deductive Database

Step1: Create Database and Tables

CREATE DATABASE IF NOT EXISTS dbase;

2	07:17:49	CREATE DATABASE IF NOT EXI...	1 row(s) affected	0.016 sec

USE dbase;

#	Time	Action	Message	Duration / Fetch
1	07:18:15	USE dbase	0 row(s) affected	0.000 sec

```
CREATE TABLE IF NOT EXISTS parents (
   parent VARCHAR(255),
   child VARCHAR(255)
);
```

#	Time	Action	Message	Duration / Fetch
1	07:18:29	CREATE TABLE IF NOT EXISTS ...	0 row(s) affected	0.032 sec

```
CREATE TABLE IF NOT EXISTS ancestors (
   ancestor VARCHAR(255),
   descendant VARCHAR(255),
   UNIQUE KEY unique_ancestor_descendant (ancestor, descendant)
);
```

2	07:18:40	CREATE TABLE IF NOT EXISTS ...	0 row(s) affected	0.062 sec

Step 2: Insert data

```
INSERT INTO parents (parent, child) VALUES
    ('a', 'b'),
    ('b', 'c'),
    ('c', 'd');
```

#	Time	Action	Message	Duration / Fetch
1	07:19:15	INSERT INTO parents (parent, chil...	3 row(s) affected Records: 3 Dupli...	0.016 sec

Step 3: Stored Procedure

```
DELIMITER &&
CREATE PROCEDURE populate_ancestors()
BEGIN
    DECLARE rows_inserted INT DEFAULT 0;
    TRUNCATE TABLE ancestors;
    INSERT INTO ancestors (ancestor, descendant)
    SELECT parent, child FROM parents;
    SET rows_inserted = ROW_COUNT();
    label: REPEAT
        INSERT INTO ancestors (ancestor, descendant)
        SELECT a.ancestor, p.child
        FROM ancestors a
        JOIN parents p ON a.descendant = p.parent
        ON DUPLICATE KEY UPDATE ancestor = a.ancestor;
        SET rows_inserted = ROW_COUNT(); -- Get number of rows inserted
    UNTIL rows_inserted = 0 -- Stop when no new rows are inserted
    END REPEAT label;
END&&
DELIMITER ;
```

#	Time	Action	Message	Duration / Fetch
1	07:19:32	CREATE PROCEDURE populate_...	0 row(s) affected	0.016 sec

Step 4: Procedure Execution

CALL populate_ancestors();

#	Time	Action	Message	Duration / Fetch
1	07:19:50	CALL populate_ancestors()	0 row(s) affected	0.094 sec

Step 5: Querying the results

SELECT * FROM ancestors;

ancestor	descendant
a	b
a	c
a	d
b	c
b	d
c	d

Lab 5

Multimedia Database

Step 1: Create Database and Tables

CREATE DATABASE IF NOT EXISTS mdb;

	#	Time	Action	Message	Duration / Fetch
✔	2	07:23:29	CREATE DATABASE IF NOT EX...	1 row(s) affected	0.000 sec

USE mdb;

	#	Time	Action	Message	Duration / Fetch
✔	1	07:23:53	USE mdb	0 row(s) affected	0.000 sec

```
CREATE TABLE IF NOT EXISTS images (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255),
    image_data LONGBLOB
);
```

	#	Time	Action	Message	Duration / Fetch
✔	1	07:27:41	CREATE TABLE IF NOT EXISTS i...	0 row(s) affected	0.047 sec

```
CREATE TABLE IF NOT EXISTS audio (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255),
    audio_data LONGBLOB
);
```

#	Time	Action	Message	Duration / Fetch
1	07:27:59	CREATE TABLE IF NOT EXISTS ...	0 row(s) affected	0.047 sec

```
CREATE TABLE IF NOT EXISTS videos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255),
    video_data LONGBLOB
);
```

#	Time	Action	Message	Duration / Fetch
1	07:28:14	CREATE TABLE IF NOT EXISTS ...	0 row(s) affected	0.031 sec

Step 2: Insert media files

1. Insert an image.

```
INSERT INTO images (name, image_data)
VALUES ('Poem', LOAD_FILE('C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/poem.jpg'));
```

#	Time	Action	Message	Duration / Fetch
1	07:42:54	INSERT INTO images (name, imag...	1 row(s) affected	0.094 sec

2. Insert an audio file.

```
INSERT INTO audio (name, audio_data)
VALUES ('Audio', LOAD_FILE('C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/Dirty - KSI.webm'));
```

#	Time	Action	Message	Duration / Fetch
1	07:44:07	INSERT INTO audio (name, audio...	1 row(s) affected	0.016 sec

3. Insert a video file.

INSERT INTO videos (name, video_data)

VALUES ('Video', LOAD_FILE('C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/abc.mp4'));

#	Time	Action	Message	Duration / Fetch
1	07:44:46	INSERT INTO videos (name, vide...	1 row(s) affected	0.016 sec

Step 3: Retrieve media files

1. Retrieve an image.

SELECT image_data FROM images WHERE id = 1;

image_data
BLOB

2. Retrieve an audio file.

SELECT audio_data FROM audio WHERE id = 1;

audio_data
BLOB

3. Retrieve a video file.

SELECT video_data FROM videos WHERE id = 1;

video_data
BLOB

Lab 6

Spatial Database

Step 1: Verify Spatial Extensions

SHOW VARIABLES LIKE 'have_%';

Variable_name	Value
have_compress	YES
have_dynamic_loading	YES
have_geometry	YES
have_openssl	YES
have_profiling	YES
have_query_cache	NO
have_rtree_keys	YES
have_ssl	YES
have_statement_timeout	YES
have_symlink	DISABLED

Step 2: Create the Database and Table

```
CREATE DATABASE IF NOT EXISTS spatial_database;

USE spatial_database;

CREATE TABLE locations (

    id INT AUTO_INCREMENT PRIMARY KEY,

    name VARCHAR(255), coordinates POINT

);
```

#	Time	Action	Message	Duration / Fetch
1	07:50:52	CREATE DATABASE IF NOT EXI...	1 row(s) affected	0.016 sec
2	07:50:53	USE spatial_database	0 row(s) affected	0.000 sec
3	07:50:53	CREATE TABLE locations (id I...	0 row(s) affected	0.063 sec

Step 3: Insert Spatial Data

INSERT INTO locations (name, coordinates) VALUES

('Location A', POINT(1.2345, 2.3456)),

('Location B', POINT(3.4567, 4.5678)),

('Location C', POINT(5.6789, 6.7890));

#	Time	Action	Message	Duration / Fetch
1	07:51:09	INSERT INTO locations (name, co...	3 row(s) affected Records: 3 Dupli...	0.015 sec

Step 4: Perform a Spatial Query

SELECT name

FROM locations

WHERE ST_DISTANCE(coordinates, POINT(1.0, 2.0)) < 1.0;

name
Location A

Lab 7

Temporal database

1. Create a new database.

CREATE DATABASE IF NOT EXISTS tdd;

```
CREATE DATABASE IF NOT EXISTS tdd;
/* Affected rows: 1  Found rows: 0  Warnings: 0  Duration for 1 query: 0.000 sec. */
```

2. Use the created database.

USE tdd;

```
USE tdd;
/* Affected rows: 0  Found rows: 0  Warnings: 0  Duration for 1 query: 0.000 sec. */
```

3. Create Table.

CREATE TABLE a_table (
id INT AUTO_INCREMENT PRIMARY KEY,
data VARCHAR(255),
valid_from TIMESTAMP(6) GENERATED ALWAYS AS ROW START,
valid_to TIMESTAMP(6) GENERATED ALWAYS AS ROW END,
PERIOD FOR SYSTEM_TIME(valid_from, valid_to)
) WITH SYSTEM VERSIONING;

```
CREATE TABLE a_table (      id INT AUTO_INCREMENT PRIMARY KEY,      data VARCHAR(255),
/* Affected rows: 0  Found rows: 0  Warnings: 0  Duration for 1 query: 0.016 sec. */
```

4. Insert Data

INSERT INTO a_table (data) VALUES ('Data 1');

```
INSERT INTO a_table (data) VALUES ('Data 1');
/* Affected rows: 1  Found rows: 0  Warnings: 0  Duration for 1 query: 0.016 sec. */
```

INSERT INTO a_table (data) VALUES ('Data 2');

```
INSERT INTO a_table (data) VALUES ('Data 2');
/* Affected rows: 1  Found rows: 0  Warnings: 0  Duration for 1 query: 0.015 sec. */
```

5. Update Data

UPDATE a_table SET data = 'Updated Data' WHERE id = 1;

```
UPDATE a_table SET data = 'Updated Data' WHERE id = 1;
/* Info: Rows matched: 1  Changed: 1  Inserted: 1  Warnings: 0 */
/* Affected rows: 1  Found rows: 0  Warnings: 0  Duration for 1 query: 0.016 sec. */
```

6. Delete Data

DELETE FROM a_table WHERE id = 2;

```
DELETE FROM a_table WHERE id = 2;
/* Affected rows: 1  Found rows: 0  Warnings: 0  Duration for 1 query: 0.000 sec. */
```

7. View Current Data

SELECT * FROM a_table;

a_table (1r × 4c)

#	id	data	valid_from	valid_to
1	1	Updated Data	2025-05-19 10:58:11.572614	2106-02-07 12:13:15.999999

8. View Historical Data

SELECT * FROM a_table FOR SYSTEM_TIME ALL;

a_table (3r × 4c)

#	id	data	valid_from	valid_to
1	1	Data 1	2025-05-19 10:57:27.629832	2025-05-19 10:58:11.572614
2	1	Updated Data	2025-05-19 10:58:11.572614	2106-02-07 12:13:15.999999
3	2	Data 2	2025-05-19 10:57:48.711292	2025-05-19 10:58:33.097374

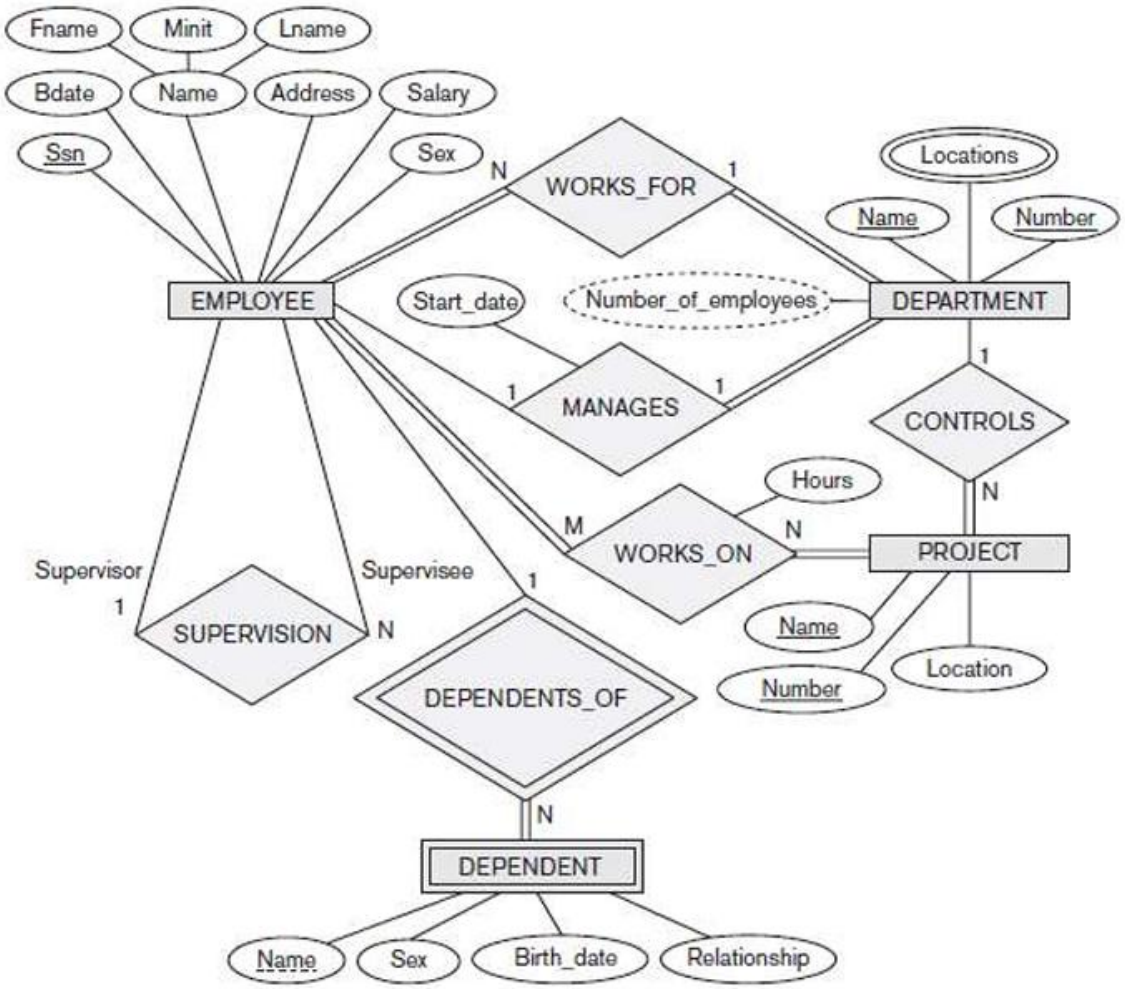
9. Drop the database

DROP DATABASE IF EXISTS tdd;

```
DROP DATABASE IF EXISTS tdd;
/* Affected rows: 1  Found rows: 0  Warnings: 0  Duration for 1 query: 0.063 sec. */
```

Lab 8
Converting EER model to Relational model

Consider the following EER.



Now, write the SQL code to convert ***EMPLOYEE***, ***DEPARTMENT*** and ***PROJECT*** into relational model. (Just use CREATE statement)

Employee:

```
CREATE TABLE EMPLOYEE (
    Ssn CHAR(9) PRIMARY KEY,
    Fname VARCHAR(30),
    Minit CHAR(1),
    Lname VARCHAR(30),
    Bdate DATE,
    Address VARCHAR(100),
```

```
    Sex CHAR(1),
    Salary DECIMAL(10, 2)
);
```

Department

```
CREATE TABLE DEPARTMENT (
    Number INT PRIMARY KEY,
    Name VARCHAR(30),
    Locations VARCHAR(100),
    Number_of_employees INT
);
```

Project

```
CREATE TABLE PROJECT (
    Number INT PRIMARY KEY,
    Name VARCHAR(30),
    Location VARCHAR(100)
);
```

Lab 9
Object Oriented Database Management System

Source Code:

- Download DB4o jar file and add to the library
- Step 1 (Import necessary class and interfaces)

```
import com.db4o.Db4o;
import com.db4o.ObjectContainer;
import com.db4o.config.Configuration;
import com.db4o.Db4oEmbedded;
import com.db4o.ObjectSet;
import com.db4o.config.EmbeddedConfiguration;
```

- Step 2 (Create a class)

```
class Student {
  private int sid;
  private String name;
  public Student(int s, String n) {
    sid = s;
    name = n;
  }
  public String toString() {
    return sid + "," + name;
  }
}
```

- Step 3 (Now store and retrieve the object of Student class)

```
public class ADBMS_OODB {

  public static void main(String[] args) {

    EmbeddedConfiguration conf = Db4oEmbedded.newConfiguration();

    ObjectContainer db = Db4oEmbedded.openFile(conf, "student.db4o");

    createFewStudents(db);

    printStudents(db);

    db.close();

  }
```

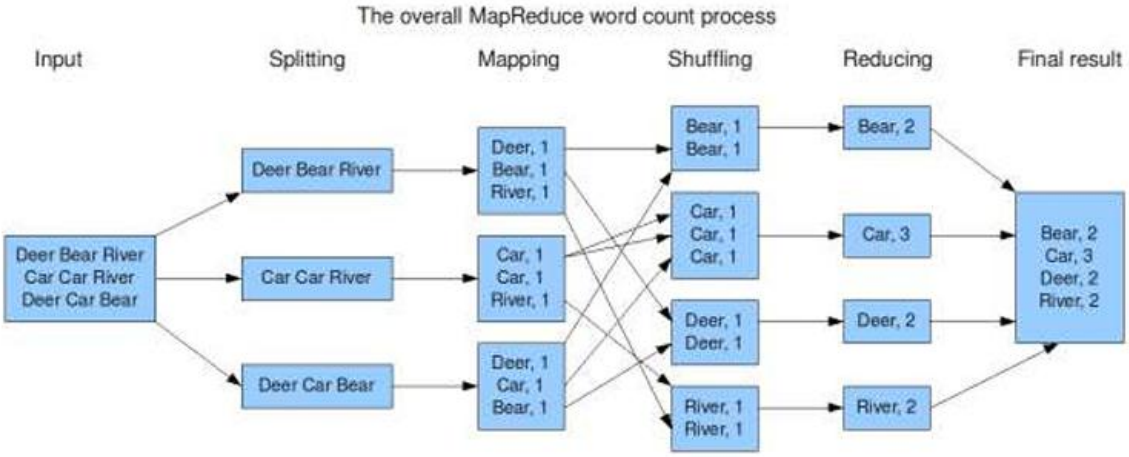
```
    public static void createFewStudents(ObjectContainer db) {
      Student S1 = new Student(1, "Ram");
      Student S2 = new Student(2, "Hari");
      db.store(S1);
      db.store(S2);
    }
    public static void printStudents(ObjectContainer db) {
      ObjectSet < Student > result = db.queryByExample(Student.class);
      System.out.println("Number of students = " + result.size() + "\n");
      while (result.hasNext()) {
        System.out.println(result.next());
      }
    }
  }
```

Output:

```
run:
Number of students = 2

1,Ram
2,Hari
BUILD SUCCESSFUL (total time: 1 second)
```

Lab 10
Map Reduce



Create a class MapReduce and perform all the tasks as described above for the corpus you will be provided by the instructor.

<u>Source Code:</u>

```
import multiprocessing
from collections import defaultdict

class MapReduce:
    def __init__(self, num_workers=2):
        self.num_workers = num_workers

    def map(self, chunk):
        word_counts = defaultdict(int)
        for word in chunk.split():
            word_counts[word] += 1
        return list(word_counts.items())

    def shuffle_sort(self, mapped_data):
        shuffled_data = defaultdict(list)
        for sublist in mapped_data:
            for word, count in sublist:
                shuffled_data[word].append(count)
        return shuffled_data
```

```
    def reduce(self, shuffled_data):
        return {word: sum(counts) for word, counts in shuffled_data.items()}

    def execute(self, text):
        chunks = text.split("\n")
        # Step 1: Map Phase (Parallel Processing)
        with multiprocessing.Pool(self.num_workers) as pool:
            mapped_data = pool.map(self.map, chunks)
        # Step 2: Shuffle and Sort Phase
        shuffled_data = self.shuffle_sort(mapped_data)
        # Step 3: Reduce Phase
        final_result = self.reduce(shuffled_data)
        return final_result

if __name__ == "__main__":
    text_corpus = """Deer Bear River
                    Car Car River
                    Deer Car Bear"""
    mr = MapReduce(num_workers=3)
    result = mr.execute(text_corpus)
    print("Final Word Count:", result)
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
[Running] python -u "c:\Users\HP\mapreduce.py"
Final Word Count: {'Deer': 2, 'Bear': 2, 'River': 2, 'Car': 3}
```

Lab 11
NoSQL Operation

13. Open MongoDB Shell.

mongosh;

```
C:\Users\HP>mongosh
Current Mongosh Log ID: 682aa63156c01f9b646c4bcf
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2
.5.1
Using MongoDB:          8.0.9
Using Mongosh:          2.5.1

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

------
   The server generated these startup warnings when booting
   2025-05-19T09:15:52.592+05:45: Access control is not enabled for the database. Read and write access to data and conf
iguration is unrestricted
------

test>
```

14. Create a database

use office;

```
test> use office
switched to db office
office>
```

15. Create a Collection (table).

db.createCollection("employees");

```
office> db.createCollection("employees");
{ ok: 1 }
office>
```

16. Show existing databases and collections.

a. Show database:

show dbs;

```
office> show dbs;
admin   40.00 KiB
config  12.00 KiB
local   40.00 KiB
office   8.00 KiB
office>
```

b. Show collections:

show collections;

```
office> show collections;
employees
office>
```

17. Insert Data into Collections.

a. Insert a single document:

db.employees.insertOne({
  name: "Sammy Basnet",
  age: 30,
})

```
office> db.employees.insertOne({
...     name: "Sammy Basnet",
...     age: 30,
... })
...
{
  acknowledged: true,
  insertedId: ObjectId('682aa71356c01f9b646c4bd0')
}
office>
```

b. Insert multiple documents:

db.employees.insertMany([
  { name: "Sita Manandhar", age: 25},
  { name: "Habit Rai", age: 34 }
])

```
office> db.employees.insertMany([
...     { name: "Sita Manandhar", age: 25},
...     { name: "Habit Rai", age: 34 }
... ])
...
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('682aa72956c01f9b646c4bd1'),
    '1': ObjectId('682aa72956c01f9b646c4bd2')
  }
}
office>
```

18. Querying Data.

a. Retrieve all documents:

db.employees.find().pretty();

```
office> db.employees.find().pretty();
[
  {
    _id: ObjectId('682aa71356c01f9b646c4bd0'),
    name: 'Sammy Basnet',
    age: 30
  },
  {
    _id: ObjectId('682aa72956c01f9b646c4bd1'),
    name: 'Sita Manandhar',
    age: 25
  },
  {
    _id: ObjectId('682aa72956c01f9b646c4bd2'),
    name: 'Habit Rai',
    age: 34
  }
]
office>
```

b. Find an employee where age > 25:

db.employees.find({ age: { $gt: 25 } });

```
office> db.employees.find({ age: { $gt: 25 } });
[
  {
    _id: ObjectId('682aa71356c01f9b646c4bd0'),
    name: 'Sammy Basnet',
    age: 30
  },
  {
    _id: ObjectId('682aa72956c01f9b646c4bd2'),
    name: 'Habit Rai',
    age: 34
  }
]
office>
```

c. Find an employee by name:

db.employees.find({ name: "Habit Rai" });

```
office> db.employees.find({ name: "Habit Rai" });
[
  {
    _id: ObjectId('682aa72956c01f9b646c4bd2'),
    name: 'Habit Rai',
    age: 34
  }
]
office>
```

19. Updating Data.

a. Update an employee's grade:

db.employees.updateOne({ name: "Sita Manandhar" }, { $set: { age: 22 } });

```
office> db.employees.updateOne({ name: "Sita Manandhar" }, { $set: { age: 22 } });
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
office>
```

b. Update multiple documents (increase age by 1 for all employees):

db.employees.updateMany({},{ $inc: { age: 1 } });

```
office> db.employees.updateMany({},{ $inc: { age: 1 } });
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 3,
  modifiedCount: 3,
  upsertedCount: 0
}
office>
```

20. Deleting Data.

a. Delete an employee based on name:

db.employees.deleteOne({ name: "Sammy Basnet" });

```
office> db.employees.deleteOne({ name: "Sammy Basnet" });
{ acknowledged: true, deletedCount: 1 }
office>
```

b. Delete all employees with age = 30:

db.employees.deleteMany({ age: 30 });

```
office> db.employees.deleteMany({ age: 30 });
{ acknowledged: true, deletedCount: 0 }
office>
```

21. Indexing for Performance.

a. Create an index on the 'name' field:

db.employees.createIndex({ name: 1 });

```
office> db.employees.createIndex({ name: 1 });
name_1
office>
```

b. View all indexes:

db.employees.getIndexes();

```
office> db.employees.getIndexes();
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { name: 1 }, name: 'name_1' }
]
office>
```

22. Dropping Collection and Databases.

a. Drop the employees collection:

db.employees.drop();

```
office> db.employees.drop();
true
office>
```

b. Drop the 'office' database:

use office;

db.dropDatabase();

```
office> use office;
already on db office
office> db.dropDatabase();
{ ok: 1, dropped: 'office' }
office>
```