Aim: Demonstrate Data Fragmentation

Objectives: To understand concept of data fragmentation.

Tools Used: MySQL Workbench

Concept:

- Fragmentation in database management systems (DBMS) refers to the division of data and indexes into non-contiguous pieces, influencing how data is stored and retrieved.
- There are two main types of fragmentation: internal and external. Internal fragmentation occurs when allocated space within data structures is not fully utilized, leading to wasted storage.
- External fragmentation happens when free space in the database is scattered throughout, hindering efficient use of available storage.
- Both types can impact system performance, causing delays in data access and retrieval.
- Managing fragmentation is crucial for optimizing database performance, often involving techniques such as defragmentation or rebuilding indexes to ensure efficient storage and retrieval of data

Problem Statement:

Demonstrate all syntax with the help on the problem statement given by instructor.

Solution:

1) **Create Two users (User1<rollno> and User2<rollno>) using parent login.**

```
-- Create Users
CREATE USER 'atharva_01'@'localhost' IDENTIFIED BY '0809';
CREATE USER 'angre_01'@'localhost' IDENTIFIED BY '0809';
```

Output

Action Output

#	Time	Action
1	11:01:45	CREATE USER 'atharva_01'@'localhost' IDENTIFIED BY '0809'
2	11:01:48	CREATE USER 'angre_01'@'localhost' IDENTIFIED BY '0809'

2) **Create a table employee with attributes empid, ename, address, salary, department in parent login.**

```sql
CREATE TABLE employee (
  empid INT PRIMARY KEY,
  ename VARCHAR(100),
  address VARCHAR(255),
  salary DECIMAL(10, 2),
  department VARCHAR(100)
);
```

Output

Action Output

#	Time	Action	Message
1	11:01:45	CREATE USER 'atharva_01'@'localhost' IDENTIFIED BY '0809'	0 row(s) affected
2	11:01:48	CREATE USER 'angre_01'@'localhost' IDENTIFIED BY '0809'	0 row(s) affected
3	11:02:23	CREATE DATABASE lab_8	1 row(s) affected
4	11:02:24	CREATE DATABASE atharva_01	1 row(s) affected
5	11:02:25	CREATE DATABASE angre_01	1 row(s) affected
6	11:02:29	USE lab_8	0 row(s) affected
7	11:02:47	CREATE TABLE employee (empid INT PRIMARY KEY, ename VARCHAR(100), address VARCHAR(2...	0 row(s) affected

3) Create link to the previously created users from the parent login.

```sql
GRANT SELECT, INSERT, UPDATE, DELETE ON employee TO 'atharva_01'@'localhost';
GRANT SELECT, INSERT, UPDATE, DELETE ON employee TO 'angre_01'@'localhost';
```

Output

Action Output

#	Time	Action	Message
1	11:01:45	CREATE USER 'atharva_01'@'localhost' IDENTIFIED BY '0809'	0 row(s) affected
2	11:01:48	CREATE USER 'angre_01'@'localhost' IDENTIFIED BY '0809'	0 row(s) affected
3	11:02:23	CREATE DATABASE lab_8	1 row(s) affected
4	11:02:24	CREATE DATABASE atharva_01	1 row(s) affected
5	11:02:25	CREATE DATABASE angre_01	1 row(s) affected
6	11:02:29	USE lab_8	0 row(s) affected
7	11:02:47	CREATE TABLE employee (empid INT PRIMARY KEY, ename VARCHAR(100), address VARCHAR(2...	0 row(s) affected
8	11:03:23	GRANT SELECT, INSERT, UPDATE, DELETE ON employee TO 'atharva_01'@'localhost'	0 row(s) affected
9	11:03:24	GRANT SELECT, INSERT, UPDATE, DELETE ON employee TO 'angre_01'@'localhost'	0 row(s) affected

4) Create table employee with attributes empid, ename, department with user1<rollno> login

```sql
USE atharva_01;

CREATE TABLE employee (
  empid INT PRIMARY KEY,
  ename VARCHAR(100),
  department VARCHAR(100)
);
```

Output

Action Output

#	Time	Action	Message
1	11:01:45	CREATE USER 'atharva_01'@'localhost' IDENTIFIED BY '0809'	0 row(s) affected
2	11:01:48	CREATE USER 'angre_01'@'localhost' IDENTIFIED BY '0809'	0 row(s) affected
3	11:02:23	CREATE DATABASE lab_8	1 row(s) affected
4	11:02:24	CREATE DATABASE atharva_01	1 row(s) affected
5	11:02:25	CREATE DATABASE angre_01	1 row(s) affected
6	11:02:29	USE lab_8	0 row(s) affected
7	11:02:47	CREATE TABLE employee (empid INT PRIMARY KEY, ename VARCHAR(100), address VARCHAR(2...	0 row(s) affected
8	11:03:23	GRANT SELECT, INSERT, UPDATE, DELETE ON employee TO 'atharva_01'@'localhost'	0 row(s) affected
9	11:03:24	GRANT SELECT, INSERT, UPDATE, DELETE ON employee TO 'angre_01'@'localhost'	0 row(s) affected
10	11:03:57	USE atharva_01	0 row(s) affected
11	11:03:59	CREATE TABLE employee (empid INT PRIMARY KEY, ename VARCHAR(100), department VARCHA...	0 row(s) affected

5) Create table employee with attributes empid, address and salary with user2<rollno>login.

```
USE angre_01;

CREATE TABLE employee (
  empid INT PRIMARY KEY,
  address VARCHAR(255),
  salary DECIMAL(10, 2)
);
```

Output

Action Output

#	Time	Action	Message
1	11:01:45	CREATE USER 'atharva_01'@'localhost' IDENTIFIED BY '0809'	0 row(s) affected
2	11:01:48	CREATE USER 'angre_01'@'localhost' IDENTIFIED BY '0809'	0 row(s) affected
3	11:02:23	CREATE DATABASE lab_8	1 row(s) affected
4	11:02:24	CREATE DATABASE atharva_01	1 row(s) affected
5	11:02:25	CREATE DATABASE angre_01	1 row(s) affected
6	11:02:29	USE lab_8	0 row(s) affected
7	11:02:47	CREATE TABLE employee (empid INT PRIMARY KEY, ename VARCHAR(100), address VARCHAR(2...	0 row(s) affected
8	11:03:23	GRANT SELECT, INSERT, UPDATE, DELETE ON employee TO 'atharva_01'@'localhost'	0 row(s) affected
9	11:03:24	GRANT SELECT, INSERT, UPDATE, DELETE ON employee TO 'angre_01'@'localhost'	0 row(s) affected
10	11:03:57	USE atharva_01	0 row(s) affected
11	11:03:59	CREATE TABLE employee (empid INT PRIMARY KEY, ename VARCHAR(100), department VARCHA...	0 row(s) affected
12	11:04:21	USE angre_01	0 row(s) affected
13	11:04:23	CREATE TABLE employee (empid INT PRIMARY KEY, address VARCHAR(255), salary DECIMAL(10,...	0 row(s) affected

6) Create trigger for inserting records into fragmented table.

```
USE lab_8;

DELIMITER $$

CREATE TRIGGER employee_insert_trigger
AFTER INSERT ON employee
FOR EACH ROW
BEGIN
    IF NEW.salary IS NOT NULL AND NEW.salary < 25000 THEN
        INSERT INTO atharva_01.employee (empid, ename, department)
        VALUES (NEW.empid, NEW.ename, NEW.department);
    ELSE
        INSERT INTO angre_01.employee (empid, address, salary)
        VALUES (NEW.empid, NEW.address, NEW.salary);
    END IF;
END $$

DELIMITER ;
```

ction Output

	Time	Action	Message
1	11:06:05	USE lab_8	0 row(s) affected
2	11:06:07	CREATE TRIGGER employee_insert_trigger AFTER INSERT ON employee FOR EACH ROW BEGIN IF...	0 row(s) affected

7) Insert minimum 5 records in employee table created in parent.

```
INSERT INTO employee (empid, ename, address, salary, department)
VALUES
(1, 'Atharva Angre', '123 Main St', 22000, 'IT'),
(2, 'Adam Ansari', '456 Elm St', 27000, 'HR'),
(3, 'Abhijeet Jadhav', '789 Pine St', 24000, 'Finance'),
(4, 'Abhishek Jha', '101 Oak St', 30000, 'IT'),
(5, 'Vineet Shinde', '202 Maple St', 23000, 'HR');
```

Output

Action Output

#	Time	Action	Message
1	11:06:05	USE lab_8	0 row(s) affected
2	11:06:07	CREATE TRIGGER employee_insert_trigger AFTER INSERT ON employee FOR EACH ROW BEGIN IF...	0 row(s) affected
3	11:06:39	INSERT INTO employee (empid, ename, address, salary, department) VALUES (1, 'Atharva Angre', '123 M...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0

8) Display records from employee table of user1<rollno> and user2<rollno>.

```
USE atharva_01;
SELECT * FROM employee;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

empid	ename	department
1	Atharva Angre	IT
3	Abhijeet Jadhav	Finance
5	Vineet Shinde	HR
NULL	NULL	NULL

employee 1

Output

Action Output

#	Time	Action	Message
1	11:06:05	USE lab_8	0 row(s) affected
2	11:06:07	CREATE TRIGGER employee_insert_trigger AFTER INSERT ON employee FOR EACH ROW BEGIN I...	0 row(s) affected
3	11:06:39	INSERT INTO employee (empid, ename, address, salary, department) VALUES (1, 'Atharva Angre', '123 ...	5 row(s) affected Records: 5 Duplicates: (
4	11:07:02	USE atharva_01	0 row(s) affected
5	11:07:06	SELECT * FROM employee LIMIT 0, 5000	3 row(s) returned

```
USE angre_01;
SELECT * FROM employee;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

empid	address	salary
2	456 Elm St	27000.00
4	101 Oak St	30000.00
NULL	NULL	NULL

employee 3

Output

Action Output

#	Time	Action	Message
4	11:07:02	USE atharva_01	0 row(s) affected
5	11:07:06	SELECT * FROM employee LIMIT 0, 5000	3 row(s) returned
6	11:07:27	SELECT * FROM employee LIMIT 0, 5000	3 row(s) returned
7	11:07:34	USE angre_01	0 row(s) affected
8	11:07:36	SELECT * FROM employee LIMIT 0, 5000	2 row(s) returned

9) Display employees whose salary is more than 25000

```
USE lab_8;
SELECT * FROM employee WHERE salary > 25000;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

empid	ename	address	salary	department
2	Adam Ansari	456 Elm St	27000.00	HR
4	Abhishek Jha	101 Oak St	30000.00	IT
NULL	NULL	NULL	NULL	NULL

employee 4

Output

Action Output

#	Time	Action	Message
6	11:07:27	SELECT * FROM employee LIMIT 0, 5000	3 row(s) returned
7	11:07:34	USE angre_01	0 row(s) affected
8	11:07:36	SELECT * FROM employee LIMIT 0, 5000	2 row(s) returned
9	11:08:12	USE lab_8	0 row(s) affected
10	11:08:15	SELECT * FROM employee WHERE salary > 25000 LIMIT 0, 5000	2 row(s) returned

10) Create table movietab in parent login using attributes movie_title, Language, Length_in_min (LENGTH IN MINUTES), lead_actor, lead_actress.

```
USE lab_8;

CREATE TABLE movietab (
  movie_title VARCHAR(255),
  language VARCHAR(100),
  length_in_min INT,
  lead_actor VARCHAR(100),
  lead_actress VARCHAR(100)
);
```

Output

Action Output

#	Time	Action	Message
8	11:07:36	SELECT * FROM employee LIMIT 0, 5000	2 row(s) returned
9	11:08:12	USE lab_8	0 row(s) affected
10	11:08:15	SELECT * FROM employee WHERE salary > 25000 LIMIT 0, 5000	2 row(s) returned
11	11:10:16	USE lab_8	0 row(s) affected
12	11:10:18	CREATE TABLE movietab (movie_title VARCHAR(255), language VARCHAR(100), length_in_min I...	0 row(s) affected

11) Create link to the users User1<rollno> & User2<rollno> from the parent login.

```
GRANT SELECT, INSERT, UPDATE, DELETE ON movietab TO 'atharva_01'@'localhost';
GRANT SELECT, INSERT, UPDATE, DELETE ON movietab TO 'angre_01'@'localhost';
```

Output

Action Output

#	Time	Action	Message
8	11:07:36	SELECT * FROM employee LIMIT 0, 5000	2 row(s) returned
9	11:08:12	USE lab_8	0 row(s) affected
10	11:08:15	SELECT * FROM employee WHERE salary > 25000 LIMIT 0, 5000	2 row(s) returned
11	11:10:16	USE lab_8	0 row(s) affected
12	11:10:18	CREATE TABLE movietab (movie_title VARCHAR(255), language VARCHAR(100), length_in_min I...	0 row(s) affected
13	11:10:58	GRANT SELECT, INSERT, UPDATE, DELETE ON movietab TO 'atharva_01'@'localhost'	0 row(s) affected
14	11:10:59	GRANT SELECT, INSERT, UPDATE, DELETE ON movietab TO 'angre_01'@'localhost'	0 row(s) affected

12) Create table movietab with same attributes in user1<rollno>.

```
USE atharva_01;

CREATE TABLE movietab (
  movie_title VARCHAR(255),
  language VARCHAR(100),
  length_in_min INT,
  lead_actor VARCHAR(100),
  lead_actress VARCHAR(100)
);
```

Output

Action Output

#	Time	Action	Message
7	11:07:34	USE angre_01	0 row(s) affected
8	11:07:36	SELECT * FROM employee LIMIT 0, 5000	2 row(s) returned
9	11:08:12	USE lab_8	0 row(s) affected
10	11:08:15	SELECT * FROM employee WHERE salary > 25000 LIMIT 0, 5000	2 row(s) returned
11	11:10:16	USE lab_8	0 row(s) affected
12	11:10:18	CREATE TABLE movietab (movie_title VARCHAR(255), language VARCHAR(100), length_in_min I...	0 row(s) affected
13	11:10:58	GRANT SELECT, INSERT, UPDATE, DELETE ON movietab TO 'atharva_01'@'localhost'	0 row(s) affected
14	11:10:59	GRANT SELECT, INSERT, UPDATE, DELETE ON movietab TO 'angre_01'@'localhost'	0 row(s) affected
15	11:11:32	USE atharva_01	0 row(s) affected
16	11:11:34	CREATE TABLE movietab (movie_title VARCHAR(255), language VARCHAR(100), length_in_min I...	0 row(s) affected

13) Create table movietab with same attributes in user2<rollno>.

```sql
USE angre_01;

CREATE TABLE movietab (
  movie_title VARCHAR(255),
  language VARCHAR(100),
  length_in_min INT,
  lead_actor VARCHAR(100),
  lead_actress VARCHAR(100)
);
```

Output

Action Output

#	Time	Action	Message
9	11:08:12	USE lab_8	0 row(s) affected
10	11:08:15	SELECT * FROM employee WHERE salary > 25000 LIMIT 0, 5000	2 row(s) returned
11	11:10:16	USE lab_8	0 row(s) affected
12	11:10:18	CREATE TABLE movietab (movie_title VARCHAR(255), language VARCHAR(100), length_in_min I...	0 row(s) affected
13	11:10:58	GRANT SELECT, INSERT, UPDATE, DELETE ON movietab TO 'atharva_01'@'localhost'	0 row(s) affected
14	11:10:59	GRANT SELECT, INSERT, UPDATE, DELETE ON movietab TO 'angre_01'@'localhost'	0 row(s) affected
15	11:11:32	USE atharva_01	0 row(s) affected
16	11:11:34	CREATE TABLE movietab (movie_title VARCHAR(255), language VARCHAR(100), length_in_min I...	0 row(s) affected
17	11:11:51	USE angre_01	0 row(s) affected
18	11:11:53	CREATE TABLE movietab (movie_title VARCHAR(255), language VARCHAR(100), length_in_min I...	0 row(s) affected

14) Create trigger in main user to insert data into fragmented table based on Length_in_min. (if Length_in_minare <60 then insert into movietab table of user1<rollno>otherwise inuser2<rollno>movietab table).

```sql
USE lab_8;

DELIMITER $$

CREATE TRIGGER movietab_insert_trigger
AFTER INSERT ON movietab
FOR EACH ROW
BEGIN
    IF NEW.length_in_min < 60 THEN
        INSERT INTO atharva_01.movietab (movie_title, language, length_in_min, lead_actor, lead_actress)
        VALUES (NEW.movie_title, NEW.language, NEW.length_in_min, NEW.lead_actor, NEW.lead_actress);
    ELSE
        INSERT INTO angre_01.movietab (movie_title, language, length_in_min, lead_actor, lead_actress)
        VALUES (NEW.movie_title, NEW.language, NEW.length_in_min, NEW.lead_actor, NEW.lead_actress);
    END IF;
END $$

DELIMITER ;
```

Output

Action Output

#	Time	Action	Message
15	11:11:32	USE atharva_01	0 row(s) affected
16	11:11:34	CREATE TABLE movietab (movie_title VARCHAR(255), language VARCHAR(100), length_in_min I...	0 row(s) affected
17	11:11:51	USE angre_01	0 row(s) affected
18	11:11:53	CREATE TABLE movietab (movie_title VARCHAR(255), language VARCHAR(100), length_in_min I...	0 row(s) affected
19	11:12:22	USE lab_8	0 row(s) affected
20	11:12:24	CREATE TRIGGER movietab_insert_trigger AFTER INSERT ON movietab FOR EACH ROW BEGIN ...	0 row(s) affected

15) Display data from both the horizontal fragments.

```sql
-- Switch to the Parent Database (lab_8) and insert data
USE lab_8;
-- Insert a movie with length < 60 minutes (should go to User1's movietab)
INSERT INTO movietab (movie_title, language, length_in_min, lead_actor, lead_actress)
VALUES ('Short Movie', 'English', 45, 'Short Actor', 'Short Actress');

-- Insert a movie with length >= 60 minutes (should go to User2's movietab)
INSERT INTO movietab (movie_title, language, length_in_min, lead_actor, lead_actress)
VALUES ('Long Movie', 'English', 120, 'Long Actor', 'Long Actress');
```

Output

Action Output

#	Time	Action	Message
16	11:11:34	CREATE TABLE movietab (movie_title VARCHAR(255), language VARCHAR(100), length_in_min I...	0 row(s) affected
17	11:11:51	USE angre_01	0 row(s) affected
18	11:11:53	CREATE TABLE movietab (movie_title VARCHAR(255), language VARCHAR(100), length_in_min I...	0 row(s) affected
19	11:12:22	USE lab_8	0 row(s) affected
20	11:12:24	CREATE TRIGGER movietab_insert_trigger AFTER INSERT ON movietab FOR EACH ROW BEGIN ...	0 row(s) affected
21	11:13:28	USE lab_8	0 row(s) affected
22	11:13:30	INSERT INTO movietab (movie_title, language, length_in_min, lead_actor, lead_actress) VALUES ('Sho...	1 row(s) affected
23	11:13:32	INSERT INTO movietab (movie_title, language, length_in_min, lead_actor, lead_actress) VALUES ('Lon...	1 row(s) affected

```sql
-- Display data from User1's movietab table
USE atharva_01;
SELECT * FROM movietab;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

movie_title	language	length_in_min	lead_actor	lead_actress
Short Movie	English	45	Short Actor	Short Actress

movietab 5

Output

Action Output

#	Time	Action	Message
18	11:11:53	CREATE TABLE movietab (movie_title VARCHAR(255), language VARCHAR(100), length_in_min I...	0 row(s) affected
19	11:12:22	USE lab_8	0 row(s) affected
20	11:12:24	CREATE TRIGGER movietab_insert_trigger AFTER INSERT ON movietab FOR EACH ROW BEGIN ...	0 row(s) affected
21	11:13:28	USE lab_8	0 row(s) affected
22	11:13:30	INSERT INTO movietab (movie_title, language, length_in_min, lead_actor, lead_actress) VALUES ('Sho...	1 row(s) affected
23	11:13:32	INSERT INTO movietab (movie_title, language, length_in_min, lead_actor, lead_actress) VALUES ('Lon...	1 row(s) affected
24	11:14:35	USE atharva_01	0 row(s) affected
25	11:14:37	SELECT * FROM movietab LIMIT 0, 5000	1 row(s) returned

```
USE angre_01;
SELECT * FROM movietab;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

movie_title	language	length_in_min	lead_actor	lead_actress
Long Movie	English	120	Long Actor	Long Actress

movietab 6

Output

Action Output

#	Time	Action	Message
20	11:12:24	CREATE TRIGGER movietab_insert_trigger AFTER INSERT ON movietab FOR EACH ROW BEGIN ...	0 row(s) affected
21	11:13:28	USE lab_8	0 row(s) affected
22	11:13:30	INSERT INTO movietab (movie_title, language, length_in_min, lead_actor, lead_actress) VALUES ('Sho...	1 row(s) affected
23	11:13:32	INSERT INTO movietab (movie_title, language, length_in_min, lead_actor, lead_actress) VALUES ('Lon...	1 row(s) affected
24	11:14:35	USE atharva_01	0 row(s) affected
25	11:14:37	SELECT * FROM movietab LIMIT 0, 5000	1 row(s) returned
26	11:14:53	USE angre_01	0 row(s) affected
27	11:14:55	SELECT * FROM movietab LIMIT 0, 5000	1 row(s) returned

Observation:

In MySQL, fragmentation can degrade query performance by increasing disk I/O and inefficient use of storage. Frequent inserts, updates, or deletes can cause data to become scattered across disk storage, leading to slower queries. However, MySQL provides corrective measures such as the OPTIMIZE TABLE command, which reorganizes the table and rebuilds indexes, improving query performance and reclaiming disk space. Proper index management and table optimization help reduce fragmentation, enhancing both disk space efficiency and query speed, making the system more efficient overall.