Name :- Prashant Suresh Shirgave

Roll No:-3 **Batch:**T1

Class: TY(CSE-AIML)

Experiment No. 2

Title :- Design and implement the Fragmentation schema
Aim : To partition the relations into horizontal & vertical fragments**.**

Theory:

Storing data in a distributed DBMS:

1. Fragmentation :

It consists of breaking a relation into smaller relations or fragments and storing the fragments possibly at different sites.

There are two types of fragmentation:

a. Horizontal fragmentation: Each fragment consists of a subset of rows of the original relation.

A fragment can be defined as a selection on the global relation r.

$$\mathbf{r_i = \sigma\, p_i (r)}$$

Original relation (r) can be reconstructed by taking union of all fragments.

$$\mathbf{r = r_1 \cup r_2 \cup r_3 \cup \ldots\ldots \cup r_n}$$

b. Vertical Fragmentation: Each fragment consists of a subset of columns of the original relation.

Each fragment r_i of r is defined by

$$\mathbf{r_i = \Pi\, R_i (r)}$$

Original relation (r) can be reconstructed by taking natural join of all fragments.

$$\mathbf{r = r_1 \times r_2 \times r_3 \times \ldots\ldots \times r_n}$$

2. Replication :

It means storing of several copies of a relation or relation fragments.

//Program for Horizontal Fragmentation (Client).

```
import socket

# Configuration
SERVER_HOST = "127.0.0.1"  # Use localhost for
testing SERVER_PORT = 1520  # Match this with
server.py

try:
    # Setup Client Socket
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((SERVER_HOST, SERVER_PORT))
    print(f"✅ Connected to server at {SERVER_HOST}:{SERVER_PORT}")
    # Send SQL Query
    query = input("Enter SQL query: ").strip()
    client_socket.sendall(query.encode())

    # Receive number of rows
    row_count =
    int(client_socket.recv(1024).decode().strip())
    print(f"📥 Received {row_count} rows.")

    # Receive and print each
    row for _ in
    range(row_count):
        row_data = client_socket.recv(1024).decode().strip()
        print("⚡", row_data)

    print("🟩Data received successfully!")

except ConnectionRefusedError:
    print("+ Error: Could not connect to the server. Make sure the server is running.")

except Exception as e:
    print(f"+ Client error:
    {e}")

finally:
    # Cleanup
    if 'client_socket' in locals():
    client_socket.close() print("🔴 Client closed.")
```

OUTPUT:

```
C:\Users\saqla\OneDrive\Desktop\adbs> & C:/Python312/python.exe
c:/Users/saqla/OneDrive/Desktop/adbs/client.py
🟩 Connected to server at 127.0.0.1:1520
Enter SQL query: select * from employeekop where ecity='Mumbai';
📥 Received 4 rows.
⚡1,Rajesh
Kumar,60000,Mumbai
2,Anita Sharma,75000,Mumbai
3,Vikram Rao,50000,Mumbai
5,Priya Iyer,70000,Mumbai
⚡
⚡
⚡
🟩 Data received successfully!
🔴 Client closed.
```

//Program for Horizontal Fragmentation (Server).

```
import socket
import pymysql

# Configuration
HOST = "127.0.0.1"  # Use "0.0.0.0" if accepting connections from other
machines PORT = 1520  # Change if needed
def handle_client(client_socket):
    """Function to handle client connection and process SQL
    queries.""" try:
        # Connect to MySQL
        conn =
        pymysql.connect(
            host="localhost",
            user="root",
            password="12345",  # Update if necessary
            database="employee",
            port=3306,
            autocommit=True
        )
        cursor = conn.cursor()
        print(" 🟩 Connected to MySQL successfully!")

        # Receive SQL Query from Client
        query = client_socket.recv(1024).decode().strip()
        print(f"📥 Received Query: {query}")

        try:
            cursor.execute(query
            ) rows =
            cursor.fetchall()

            # Send row count
            row_count =
            len(rows)
            client_socket.sendall(f"{row_count}\n".encode())

            # Send each row as a comma-separated
            string for row in rows:
                formatted_data = ",".join(str(i) for i in row) + "\n"
                client_socket.sendall(formatted_data.encode())

            print(" 🟩 Data sent

        successfully!") except

        pymysql.MySQLError as db_error:
            print(f"+ Database error: {db_error}")
            client_socket.sendall(f"ERROR:
            {db_error}\n".encode())
        finally:
            cursor.close()
            conn.close()

    except Exception as e:
        print(f"+ Server error:
        {e}")
        client_socket.sendall(f"ERROR: {e}\n".encode())

    finally:
        client_socket.close()
        print("🔴 Client disconnected.")
# Start the server
def
start_server():
    """Function to initialize the server and accept multiple clients."""
    server_socket = socket.socket(socket.AF_INET,
    socket.SOCK_STREAM)
    server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1) #
 Allow reusing the port

    try:
        server_socket.bind((HOST, PORT))
        server_socket.listen(5)  # Allow up to 5 clients in the queue
        print(f"🚀 Server is running on {HOST}:{PORT}, waiting for clients...")

        while True:
            client_socket, client_address =
            server_socket.accept() print(f" 🟩Connection
            established with {client_address}")
            handle_client(client_socket)

    except KeyboardInterrupt:
        print("\n🔴 Server shutting down due to manual interruption.")

    except Exception as e:
        print(f"+ Critical Server Error: {e}")

    finally:
        server_socket.close()
        print("🔴 Server
        closed.")

 if __name__== "__main__
    ": start_server()
```

OUTPUT:

```
PS C:\Users\saqla\OneDrive\Desktop\adbs> & C:/Python312/python.exe
c:/Users/saqla/OneDrive/Desktop/adbs/server.py
Server is running on 127.0.0.1:1520, waiting for clients...
🟩 Connection established with ('127.0.0.1', 55714)
🟩 Connected to MySQL successfully!
Received Query: select * from employeekop where ecity='Mumbai';
🟩 Data sent successfully!
🔴 Client disconnected.
```

//Program for Vertical Fragmentation(Client).

```
import socket
import struct
# Server details
HOST =
"127.0.0.1"
PORT = 1520

# Connect to server
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect((HOST, PORT))
print(f" 🟩 Connected to server at {HOST}:{PORT}")

# Ask user for SQL query
query = input(" Enter the SQL query to
perform: ") client_socket.send(query.encode())
print(" Query sent to server!")

# Receive number of rows
count = struct.unpack('i', client_socket.recv(4))[0]
print(f" Rows received: {count}")

# Receive and print data
print("\n Query
Results:") for _ in
range(count):
    # Receive ID
    eid = struct.unpack('i', client_socket.recv(4))[0]

    # Receive Name Length
    name_length = struct.unpack('i', client_socket.recv(4))[0]

    # Receive Name (Based on received length)
    ename = client_socket.recv(name_length).decode()

    print(f"{eid}\t{ename}")

print(" 🟩Data received successfully!")
client_socket.close()
```

```
print("🔴 Disconnected from server.")
```

OUPUT:

```
PS C:\Users\saqla\OneDrive\Desktop\adbs> & C:/Python312/python.exe
c:/Users/saqla/OneDrive/Desktop/adbs/client1.py
🟩 Connected to server at 127.0.0.1:1520
📝 Enter the SQL query to perform: select eid,ename from employee;
📤 Query sent to server!
📥 Rows received: 6

📊Query Results:
101    Shweta
102    Arjun
103    Shlok
104    Riya
105    Ahana
106    Tara
🟩 Data received successfully!
🔴 Disconnected from server.
```

//Program for Vertical Fragmentation(Server).

```
import socket
import mysql.connector
import struct

# Database
connection try:
    conn = mysql.connector.connect(
        host="localhost",
        user="root",
        password="12345",
        database="employee"
    )
    cursor = conn.cursor()
   print("🟩 Connected to MySQL
successfully!") except mysql.connector.Error
as err:
    print(f"Error: {err}")
    exit(1)

# Setting up the server
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(("127.0.0.1", 1520))
server_socket.listen(1)
print("🚀 Server is running on 127.0.0.1:1520, waiting for clients...")


client_socket, addr = server_socket.accept()
print(f"🟩 Connection established {addr}")
```

```
try:
    # Receiving query from client
    query =
    client_socket.recv(1024).decode()
    print(f"📥 Received Query: {query}")

    # Execute the query
    cursor.execute(query)
    results =
    cursor.fetchall()

    # Sending number of
    rows count = len(results)
    client_socket.send(struct.pack('i',  count))

    # Sending data row by
    row for row in results:
        eid = row[0]
        ename = row[1].encode()  # Convert string to bytes
        name_length = len(ename) # Send ID and Name
        Length client_socket.send(struct.pack('i', eid))  #
        Send integer ID
        client_socket.send(struct.pack('i', name_length))  # Send name length
        client_socket.send(ename)  # Send name bytes

    print(" 🟩Data sent successfully!")

except Exception as e:
    print("Error:",
e) finally:
    client_socket.close()
    print("🔴 Client
    disconnected.")
    server_socket.close()
    cursor.close()
    conn.close()
```

OUTPUT:

```
PS C:\Users\saqla\OneDrive\Desktop\adbs> & C:/Python312/python.exe
c:/Users/saqla/OneDrive/Desktop/adbs/server1.py
🟩 Connected to MySQL successfully!
🚀 Server is running on 127.0.0.1:1520, waiting for clients...
🟩 Connection established with ('127.0.0.1', 3857)
📥Received Query: select eid,ename from employee;
🟩 Data sent successfully!
🔴 Client disconnected.
```

Conclusion:- As per requirement, it is possible to fragment the relation in distributed database systems using horizontal & vertical fragmentation .