Name :- Prashant Suresh Shirgave

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

Class: TY(CSE-AIML)

Experiment No. 3

Title :- Implement Semi join in distributed DBMS.

Aim : To demonstrate the semi join operation of distributed environment.

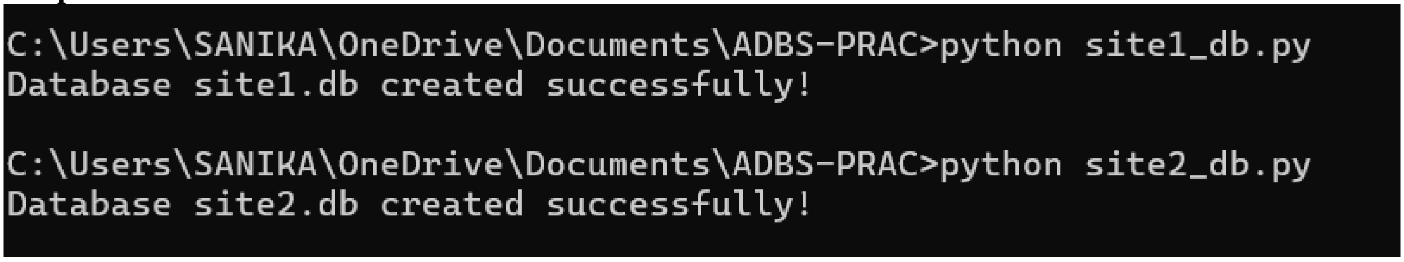
site1_db.py:

```
import sqlite3

# Connect to Site 1 database
conn = sqlite3.connect('site1.db')
cursor = conn.cursor()

# Create table1
cursor.execute('''
    CREATE TABLE IF NOT EXISTS table1 (
        accno INTEGER PRIMARY KEY,
        cname TEXT
    )
''')

# Insert sample data
cursor.execute("INSERT OR IGNORE INTO table1 VALUES (100, 'Sanika')")
cursor.execute("INSERT OR IGNORE INTO table1 VALUES (101, 'Prashant')")
cursor.execute("INSERT OR IGNORE INTO table1 VALUES (102, 'Nikhil')")
cursor.execute("INSERT OR IGNORE INTO table1 VALUES (201, 'Omkar')")
cursor.execute("INSERT OR IGNORE INTO table1 VALUES (202, 'Sushant')")

conn.commit()
conn.close()
print("Database site1.db created successfully!")
```

site2_db.py:

```
import sqlite3

# Connect to Site 2 database
conn = sqlite3.connect('site2.db')
cursor = conn.cursor()

# Create table2
cursor.execute('''
    CREATE TABLE IF NOT EXISTS table2 (
        accno INTEGER PRIMARY KEY,
        bname TEXT,
        bal INTEGER
    )
''')

# Insert sample data
cursor.execute("INSERT OR IGNORE INTO table2 VALUES (100, 'Bawada', 1000)")
cursor.execute("INSERT OR IGNORE INTO table2 VALUES (101, 'Shahupuri', 2000)")
cursor.execute("INSERT OR IGNORE INTO table2 VALUES (102, 'Laxmipuri', 3000)")
cursor.execute("INSERT OR IGNORE INTO table2 VALUES (103, 'Rajarampuri', 3000)")
cursor.execute("INSERT OR IGNORE INTO table2 VALUES (104, 'Papachi', 3000)")

conn.commit()
conn.close()
print("Database site2.db created successfully!")
```

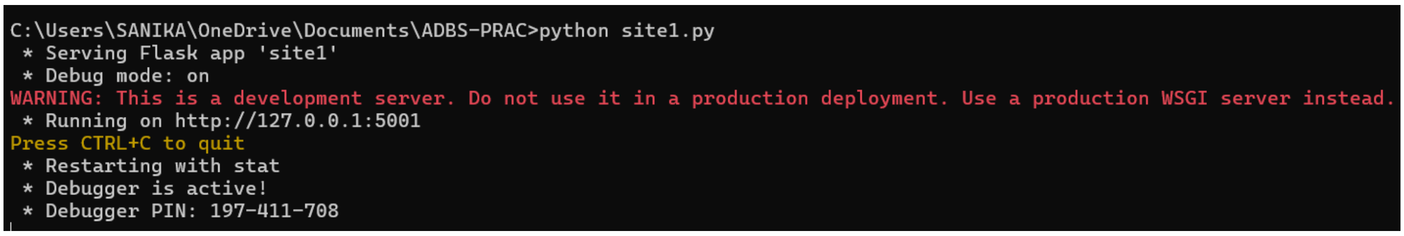
Output:



```
C:\Users\SANIKA\OneDrive\Documents\ADBS-PRAC>python site1_db.py
Database site1.db created successfully!

C:\Users\SANIKA\OneDrive\Documents\ADBS-PRAC>python site2_db.py
Database site2.db created successfully!
```

site1.py:

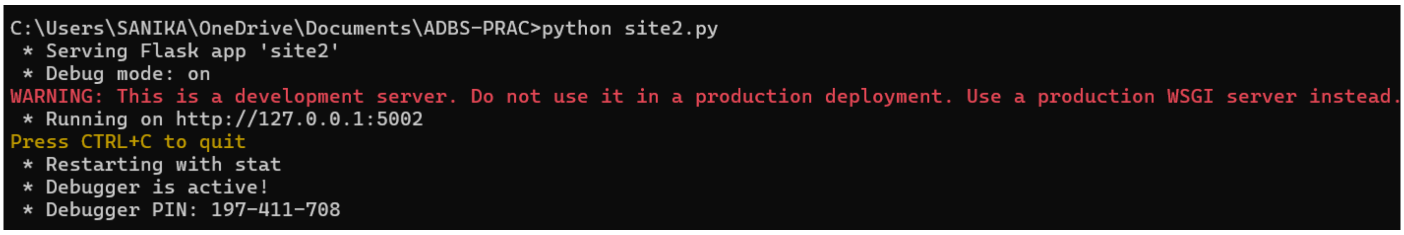
```
from flask import Flask, request, jsonify
import sqlite3

app = Flask(__name__)

# Step 1: Extract accno from table1
@app.route('/semi_join_step1', methods=['GET'])
def semi_join_step1():
    conn = sqlite3.connect('site1.db')
    cursor = conn.cursor()

    # Extract accno column from table1
    cursor.execute("SELECT accno FROM table1")
    accnos = [row[0] for row in cursor.fetchall()]

    conn.close()
    return jsonify({"temp1": accnos})  # Send temp1 to Site 2

# Step 3: Receive filtered data (temp2) and perform the final join
@app.route('/semi_join_step3', methods=['POST'])
def semi_join_step3():
    temp2 = request.json['temp2']

    conn = sqlite3.connect('site1.db')
    cursor = conn.cursor()

    # Perform final join
    query = '''
        SELECT t1.accno, t1.cname FROM table1 t1
        WHERE t1.accno IN ({})
    '''.format(','.join('?' * len(temp2)))
    cursor.execute(query, [t[0] for t in temp2])
    result = cursor.fetchall()

    conn.close()
    return jsonify({"semi_join_result": result})

if __name__ == '__main__':
    app.run(port=5001, debug=True)
```



```
C:\Users\SANIKA\OneDrive\Documents\ADBS-PRAC>python site1.py
 * Serving Flask app 'site1'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5001
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 197-411-708
```

site2.py:

```
from flask import Flask, request, jsonify
import sqlite3

app = Flask(__name__)

# Step 2: Receive temp1 and filter using table2
@app.route('/semi_join_step2', methods=['POST'])
def semi_join_step2():
    temp1 = request.json['temp1']

    conn = sqlite3.connect('site2.db')
    cursor = conn.cursor()

    # Perform join with table2 to filter relevant records
    query = "SELECT accno, bname, bal FROM table2 WHERE accno IN ({})".format(
        ','.join('?' * len(temp1))
    )
    cursor.execute(query, temp1)
    temp2 = cursor.fetchall()

    conn.close()
    return jsonify({"temp2": temp2})  # Send filtered data back to Site 1

if __name__ == '__main__':
    app.run(port=5002, debug=True)
```

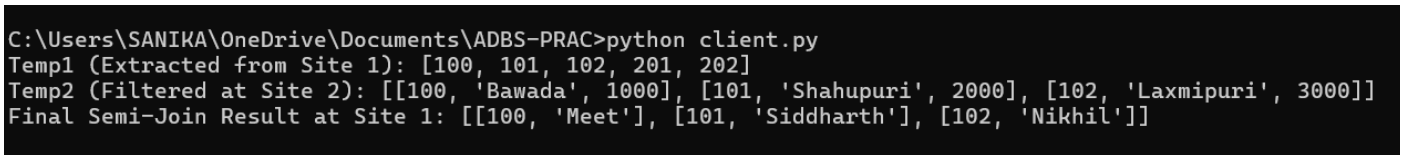


```
C:\Users\SANIKA\OneDrive\Documents\ADBS-PRAC>python site2.py
 * Serving Flask app 'site2'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5002
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 197-411-708
```

Client.py:

```
import requests

# Step 1: Request temp1 from Site 1
response = requests.get("http://127.0.0.1:5001/semi_join_step1")
temp1 = response.json()['temp1']
print("Temp1 (Extracted from Site 1):", temp1)

# Step 2: Send temp1 to Site 2 for filtering
response = requests.post("http://127.0.0.1:5002/semi_join_step2", json={"temp1": temp1})
temp2 = response.json()['temp2']
print("Temp2 (Filtered at Site 2):", temp2)

# Step 3: Send temp2 back to Site 1 for final join
response = requests.post("http://127.0.0.1:5001/semi_join_step3", json={"temp2": temp2})
final_result = response.json()['semi_join_result']
print("Final Semi-Join Result at Site 1:", final_result)
```

Output:



```
C:\Users\SANIKA\OneDrive\Documents\ADBS-PRAC>python client.py
Temp1 (Extracted from Site 1): [100, 101, 102, 201, 202]
Temp2 (Filtered at Site 2): [[100, 'Bawada', 1000], [101, 'Shahupuri', 2000], [102, 'Laxmipuri', 3000]]
Final Semi-Join Result at Site 1: [[100, 'Meet'], [101, 'Siddharth'], [102, 'Nikhil']]
```

Conclusion: Semi join operation in distributed DBMS reduces the cost of performing the join operation.