CSE 512: Distributed Database Systems

Project Report: Design and Implementation of a Distributed Database System (Part 1)

Group Name: Data Dominators

1. Introduction

The objective of this project is to design and implement a distributed database system capable of handling real-time data updates and queries while ensuring data consistency and availability. The chosen topic for the database system is a Social Networking platform. The implementation utilizes PostgreSQL as the relational database management system.

2. Database Schema

2.1 Distributed Database Schema

The database schema consists of four main tables:

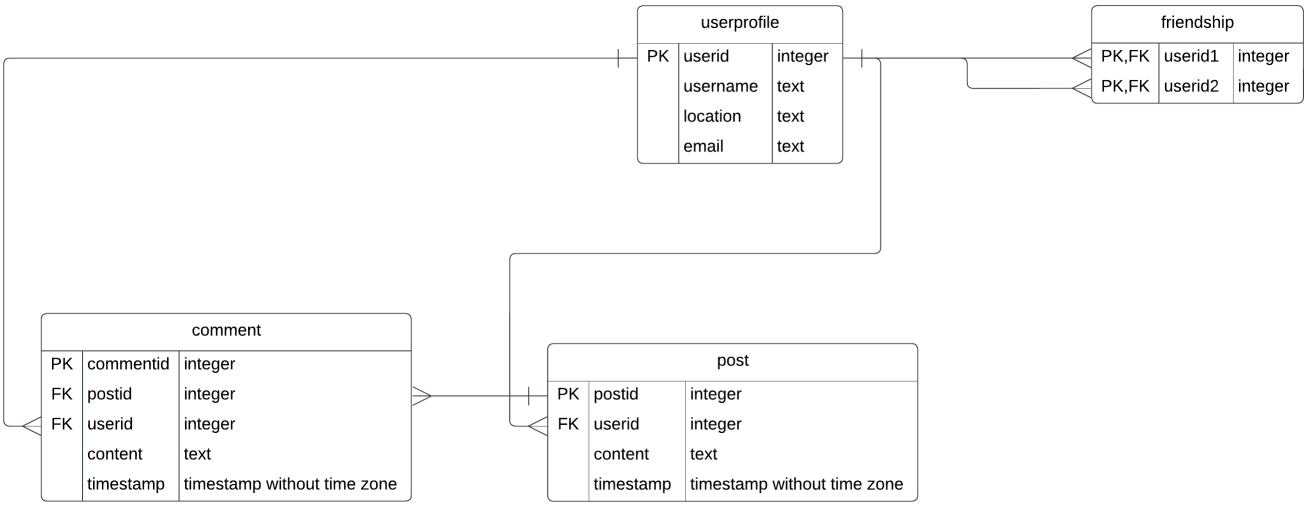
- **UserProfile:**
  - UserID (Primary Key)
  - UserName
  - Location
  - Email
- **Post:**
  - PostID (Primary Key)
  - UserID (Foreign Key referencing UserProfile.UserID)
- **Comment:**
  - CommentID (Primary Key)
  - PostID (Foreign Key referencing Post.PostID)
  - UserID (Foreign Key referencing UserProfile.UserID)
  - Content
  - Timestamp
- **Friendship:**
  - UserID1 (Foreign Key referencing UserProfile.UserID)

- UserID2 (Foreign Key referencing UserProfile.UserID)
- (Primary Key: UserID1, UserID2)

2.2 Database Tables

The script includes code for creating the necessary tables (UserProfile, Post, Comment, Friendship) with appropriate attributes, keys, and constraints.

2.3 Entity Relationship Diagram



3. Data Distribution

The data distribution strategy involves the creation of two child databases, child1 and child2. Users and their associated data are distributed between these child databases based on a hash function. Below is the screenshot of the code snippet showing how we are utilizing hash value to distribute the data.

```
hash_value = hashlib.sha256(str(user_id).encode()).hexdigest()
hash_integer = int(hash_value, 16)  # Convert hexadecimal hash to integer

if hash_integer % 2 == 0:
    with conn_child1.cursor() as cursor_child1:
        cursor_child1.execute("INSERT INTO Post (PostID, UserID) VALUES (%s, %s)", (post_id, user_id))
else:
    with conn_child2.cursor() as cursor_child2:
        cursor_child2.execute("INSERT INTO Post (PostID, UserID) VALUES (%s, %s)", (post_id, user_id))
```

4. Data Insertion, Update and Delete

The script provides mechanisms for efficient data insertion into the UserProfile, Post, Comment, and Friendship tables. The distribution of users and posts between child databases is handled based on the hash of the UserID. Below is the screenshot of the data insertion commands followed by results showing data after insertion.

```python
if __name__ == '__main__':
    # Create the 'Project' database
    #create_database(DATABASE_NAME)
    #create_database_child(child1)
    #create_database_child2(child2)

    with connect_postgres(dbname=DATABASE_NAME) as conn_project, \
        connect_postgres(dbname=child1) as conn_child1, \
        connect_postgres(dbname=child2) as conn_child2:
        conn_project.set_isolation_level(psycopg2.extensions.ISOLATION_LEVEL_AUTOCOMMIT)
        conn_child1.set_isolation_level(psycopg2.extensions.ISOLATION_LEVEL_AUTOCOMMIT)
        conn_child2.set_isolation_level(psycopg2.extensions.ISOLATION_LEVEL_AUTOCOMMIT)

        #create_table(conn_project)
        #create_table1(conn_child1)
        #create_table1(conn_child2)

        add_user(conn_project, 'Pujith ', 'Tempe', 'pujithsaip@gmail.com')
        add_user(conn_project, 'Rishi ', 'Tempe', 'rishikumar@gmail.com')
        add_user(conn_project, 'Ajay', 'Chandler', 'ajay@yahoo.com')
        add_post(conn_project,3)
        add_post(conn_project,2)
        add_friendship(conn_project, 1, 2)
        add_comment(conn_project, 1, 2, "Good")
```

```
Data in UserProfile table:
['userid', 'username', 'location', 'email']
(1, 'Pujith ', 'Tempe', 'pujithsaip@gmail.com')
(2, 'Rishi ', 'Tempe', 'rishikumar@gmail.com')
(3, 'Ajay', 'Chandler', 'ajay@yahoo.com')

Data in Friendship table:
['userid1', 'userid2']
(1, 2)
(2, 1)

Data in Post table:
['postid', 'userid']
(1, 3)
(2, 2)

Data in Comment table:
['commentid', 'postid', 'userid', 'content', 'timestamp']
(1, 1, 2, 'Good', datetime.datetime(2023, 11, 26, 18, 26, 20, 645516))
```

Below is the screenshot of the data update commands and followed by the result showing data after update.

```python
update_post(conn_project, 1, 1)
update_comment(conn_project, 1, "Very good")
update_user(conn_project, 1, "Pujith Sai", "New York", 'Pujithsai@gmail.com')
```

```
Data in UserProfile table:
['userid', 'username', 'location', 'email']
(2, 'Rishi ', 'Tempe', 'rishikumar@gmail.com')
(3, 'Ajay', 'Chandler', 'ajay@yahoo.com')
(1, 'Pujith Sai', 'New York', 'Pujithsai@gmail.com')

Data in Friendship table:
['userid1', 'userid2']
(1, 2)
(2, 1)

Data in Post table:
['postid', 'userid']
(2, 2)
(1, 1)

Data in Comment table:
['commentid', 'postid', 'userid', 'content', 'timestamp']
(1, 1, 2, 'Very good', datetime.datetime(2023, 11, 26, 18, 26, 20, 645516))
```

Below is the screenshot of the command deleting a user and followed by the screenshots showing data before and after deletion.

```
    delete_user(conn_project, 7, 'Pujith')
    print_data_in_child(conn_project)
```

Data before deletion:

```
Data in UserProfile table:
['userid', 'username', 'location', 'email']
(2, 'Rishi ', 'Tempe', 'rishikumar@gmail.com')
(3, 'Ajay', 'Chandler', 'ajay@yahoo.com')
(7, 'Pujith ', 'Tempe', 'pujithsaip@gmail.com')

Data in Friendship table:
No data found.

Data in Post table:
['postid', 'userid']
(2, 3)
(6, 7)

Data in Comment table:
No data found.
```

Data after deletion:

```
Data in UserProfile table:
['userid', 'username', 'location', 'email']
(2, 'Rishi ', 'Tempe', 'rishikumar@gmail.com')
(3, 'Ajay', 'Chandler', 'ajay@yahoo.com')

Data in Friendship table:
No data found.

Data in Post table:
['postid', 'userid']
(2, 3)

Data in Comment table:
No data found.
```

5. Data Retrieval

The script includes functions for retrieving data from both child databases. Functions such as get_user_by_username, get_post_by_id and get_comment_by_id demonstrate successful basic data retrieval queries. Below is the screenshot of the data retrieval commands followed by their results.

```
get_all_comments_on_post(conn_project, 1)
get_all_posts_of_user(conn_project, 2)
get_all_users(conn_project)
get_comment_by_id(conn_project, 1)
get_post_by_id(conn_project, 1)
get_user_by_username(conn_project, "Pujith")
```

```
get_all_posts_of_user result: [(2, 2)]
get_all_users result: [(2, 'Rishi ', 'Tempe', 'rishikumar@gmail.com'), (3, 'Ajay', 'Chandler',
'ajay@yahoo.com'), (1, 'Pujith Sai', 'New York', 'Pujithsai@gmail.com')]
get_comment_by_id result: (1, 1, 2, 'Very good', datetime.datetime(2023, 11, 26, 18, 26, 20, 645516))
get_post_by_id result: (1, 1)
get_user_by_username result: None
```

6. Conclusion

The implemented distributed database system successfully meets the Part 1 project requirements, including schema design, table creation, data distribution, data insertion, and data retrieval. The system is designed to handle a Social Networking platform with distributed data across multiple child databases, ensuring scalability and fault tolerance.