CSE 512: Distributed Database Systems

Project Report: Fragmentation and Replication Techniques (Part 2)

Group Name: Data Dominators

1. Introduction

The objective of Part 2 is to enhance the performance of the distributed database system by implementing fragmentation and replication techniques. The focus is on optimizing data storage and retrieval within the chosen Social Networking platform. The selected techniques include Horizontal Fragmentation, Vertical Fragmentation, and Replication Strategies. PostgreSQL continues to serve as the relational database management system.

2. Horizontal and Vertical Fragmentation

2.1 Horizontal Fragmentation

Implement horizontal fragmentation by splitting tables based on specific criteria. In this case, consider fragmenting the User Profile table based on geographical location Tempe. We wanted to separate the people who are living in the Tempe region and using social media. For this we have created a function that will do horizontal fragmentation and will update the table whenever it is performed.

Code Executing the Function:

```
horizontal_fragmentation_by_region(conn_project, 'Tempe')
print_data_in_child_horizontal(conn_project)
```

Results:

Data is being segregated based on region.

```
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.

Data in userprofile_tempe table:
['userid', 'username', 'location', 'email']
(2, 'Rishi ', 'Tempe', 'rishikumar@gmail.com')
```

2.2 Vertical Fragmentation

Divide tables into smaller subsets based on columns to optimize data retrieval. Identify columns frequently accessed together and create new tables accordingly. Here we have observed that email is one of the main things in social media that is used for login and to share information. So, we felt that it needs to be separated from the user profile Table and needs to be in a separate table so that it can be accessed quickly. We did that using vertical fragmentation. We also created separated crud functions based on the vertical fragmentation so that it won't be a problem while we operate the crud operations after vertical fragmentation.

Code to Execute:

```
vertical_fragmentation_email_only(conn_project)
print_data_in_child_vertical(conn_project)
```

Results of UserProfile Table undergoing vertical fragmentation:

```
Data in UserProfile table:
['userid', 'username', 'location']
(2, 'Rishi ', 'Tempe')
(3, 'Ajay', 'Chandler')

Data in Friendship table:
['userid1', 'userid2']
(2, 3)
(3, 2)

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

Data in Comment table:
['commentid', 'postid', 'userid', 'content', 'timestamp']
(3, 2, 3, 'Poor', datetime.datetime(2023, 11, 27, 0, 4, 42, 741289))

Data in userprofile_tempe table:
['userid', 'username', 'location', 'email']
(2, 'Rishi ', 'Tempe', 'rishikumar@gmail.com')

Data in UserProfile_EmailOnly table:
['userid', 'email']
(2, 'rishikumar@gmail.com')
(3, 'ajay@yahoo.com')
```

We have also developed separate CRUD operations so that it vertical fragmentation wont hurt them

Code executing the CRUD:

```
update_user_crud(conn_project, 3, "Ajay", 'NewYork', 'ajayreddy@gmail.com')
print_data_in_child_vertical(conn_project)
```

Results:

```
Data in UserProfile table:
['userid', 'username', 'location']
(2, 'Rishi ', 'Tempe')
(3, 'Ajay', 'NewYork')

Data in Friendship table:
['userid1', 'userid2']
(2, 3)
(3, 2)

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

Data in Comment table:
['commentid', 'postid', 'userid', 'content', 'timestamp']
(3, 2, 3, 'Poor', datetime.datetime(2023, 11, 27, 0, 4, 42, 741289))

Data in userprofile_tempe table:
['userid', 'username', 'location', 'email']
(2, 'Rishi ', 'Tempe', 'rishikumar@gmail.com')

Data in UserProfile_EmailOnly table:
['userid', 'email']
(2, 'rishikumar@gmail.com')
(3, 'ajayreddy@gmail.com')
```

3. Replication Setup

3.1 Master and slave:

We have set up the main database and then set up two child databases to store the replicated data. Here we are replicating the tables based on hashing. We are creating the hash based on the user id and then we are storing the data based on the hash key in child1 database or child 2 databases. All crud operations are coded in such a way that once they operate on the main database they changed/new entry into the tables will be replicated directly into the child databases based on the hash key.

The tables we have replicated are the user profile table and post table.

Child Databases:

```
Child Database

Data in UserProfile table:
['id', 'userid', 'username', 'location', 'email']
(1, 3, 'Ajay', 'Chandler', 'ajay@yahoo.com')

Data in Post table:
['id', 'postid', 'userid']
(1, 1, 3)
Child Database

Data in UserProfile table:
['id', 'userid', 'username', 'location', 'email']
(1, 1, 'Pujith ', 'Tempe', 'pujithsaip@gmail.com')
(2, 2, 'Rishi ', 'Tempe', 'rishikumar@gmail.com')

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

Executing the crud and the replication will be done along with the main database update:

```
update_user_crud(conn_project, 3, "Ajay", "New York", "ajay@yahoo.com")
print_data_in_child_vertical(conn_project)
print_data_in_child1(conn_child1)
print_data_in_child1(conn_child2)
```

Results of updated data in master and slave nodes.

Master Database updating:

```
Data in UserProfile table:
['userid', 'username', 'location']
(1, 'Pujith ', 'Tempe')
(2, 'Rishi ', 'Tempe')
(3, 'Ajay', 'New York')

Data in Friendship table:
['userid1', 'userid2']
(2, 3)
(3, 2)

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

Data in Comment table:
['commentid', 'postid', 'userid', 'content', 'timestamp']
(1, 2, 3, 'Poor', datetime.datetime(2023, 11, 27, 0, 19, 40, 32293))

Data in userprofile_tempe table:
['userid', 'username', 'location', 'email']
(1, 'Pujith ', 'Tempe', 'pujithsaip@gmail.com')
(2, 'Rishi ', 'Tempe', 'rishikumar@gmail.com')

Data in UserProfile_EmailOnly table:
['userid', 'email']
(1, 'pujithsaip@gmail.com')
(2, 'rishikumar@gmail.com')
(3, 'ajay@yahoo.com')
```

Slave nodes updation:

```
Child Database

Data in UserProfile table:
['id', 'userid', 'username', 'location', 'email']
(1, 3, 'Ajay', 'New York', 'ajay@yahoo.com')

Data in Post table:
['id', 'postid', 'userid']
(1, 1, 3)
Child Database

Data in UserProfile table:
['id', 'userid', 'username', 'location', 'email']
(1, 1, 'Pujith ', 'Tempe', 'pujithsaip@gmail.com')
(2, 2, 'Rishi ', 'Tempe', 'rishikumar@gmail.com')

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

Conclusion:

The implemented distributed database system successfully meets the Part 2 project requirements. We have successfully implemented the Vertical fragmentation, Horizontal Fragmentation and Replication of main database into two other Databases. l. The system is designed to handle a Social Networking platform with distributed data across multiple child databases, ensuring scalability and fault tolerance. We even made separate crud operation for vertical fragmentation to work smoothly and help retrieve email information quickly.