Assignment - 3
Big Data Analytics

Aakarsh Jain
2021507

September 2024

1 Data Modeling & Schema Design:

First, we assume the following attributes in each of the tables in PostgreSQL DB -

- Departments - ID, Name
- Students - Name, Roll no, Department ID etc (personal immutable info of every students, roll no. acts as primary key)
- Instructors - ID, Name, Department ID (personal immutable info of every instructor)
- Courses - Course code, Name, Department ID, Instructor, Semester Offere, Core/Elective
- Enrollments - Course code, Student roll no., Semester Offerered

Now, since we are migrating to MongoDB, it would be preferable to modify the above schema based on the queries we have to serve. We begin with 4 collections -

- Students - Name, Roll no., dept_id
- Instructor - ID, Name, Collection of Courses instructed, dept_id
- Department - ID, Name
- Courses - Course code, name, dept_id, enrollments

The `Enrollment` table can now be sharded and stored inside `Course` document. This approach also allows us to store the results for some queries inside the document itself, and update whenever we have a new entry. Furthermore, this schema also reduces the need to perform joins wherever possible. We still store the course-professor mapping in the `Instructor` document, along with the semester in which the course was offered, as this is more optimized for the given queries

2 Data Migration

The script `migrate.py` helps to migrate all the data from PostgreSQL to MongoDB database, with the above shared schema in mind.

We start by creating the `student` and `department` collection since all the relevant information is already available in the respective tables. We can then create the `instructor` collection. Now, once we start constructing the `course` collection, we take care to store the information about instructor taking the course back in the `instructor` document for query optimization.

Now for every enrollment, we store that in the corresponding `course` document for query optimisation. This violates some of our normalisation conditions and buckets the data for faster processing.

3 Query Implementation using Apache Spark

The script `queries.py` implements a separate method to perform each of the query listed in the problem statement. The MongoDB connector for Apache Spark is used to connecting Apache spark with the database. To execute any query, modify the `main` function in the script to call the correct helper function for each query with the required parameters.

4 Performance Analysis and Optimisation

The following performance comparisons are made for a database having $\approx$ 10k rows, with roughly equal distribution among all departments and courses.

- We start by indexing `Student`, `Courses` and `Professor` collections.

  For queries such as getting information of every student rolled in a course, this leads to a massive speedup of $\approx 500ms$ since students are accessed using their roll no. repeatedly.

  In other queries where we access courses and instructors using their IDs, we see similar improvements.

- The query for finding average number students enrolling in an instructors course can be further improved by caching the documents of each course. This led to $\approx 1.2s$ improvement, as each professor had 30 courses on average in the generated database.

- Since we have already partitioned data, all enrollment in the respective course document, the benefit due to techniques such as `Predicate pushdown` (using filtering commands earlier in the query) and `Partitioning & Bucketing` is limited.