University of Science and Technology of Hanoi

ICT Department

Distributed Systems Report

Topic: Distributed Database

Group 6

Hanoi, February 2022

Contents

List of Figures

1 Introduction

1.1 Distributed Database

A distributed database (DDB) is a collection of databases that are physically spread across multiple sites in a computer network. This may be necessary if a database needs to be viewed by a large number of people all over the world. It must appear to users as a single database with no physical components in common.

In DDB, the files must be formatted, logically interconnected, and physically dispersed over numerous sites to form a distributed database system. A common interface for accessing the distributed data is required.

The applications of Distributed Database is huge, from the Corporate Management Information System to multimedia applications, Military control systems or Hotel chains. It is also used in the manufacturing control system and many other systems.

1.2 The Types of Distributed Database

There are two types of Distributed Database, which are: **Homogeneous** and **Heterogeneous Distributed Database**

1.2.1 Homogeneous Distributed Database

In a homogeneous database, all different sites store the database identically. The operating system, database management system, and the data structures used – all are the same at all sites. Hence, they're easy to manage.

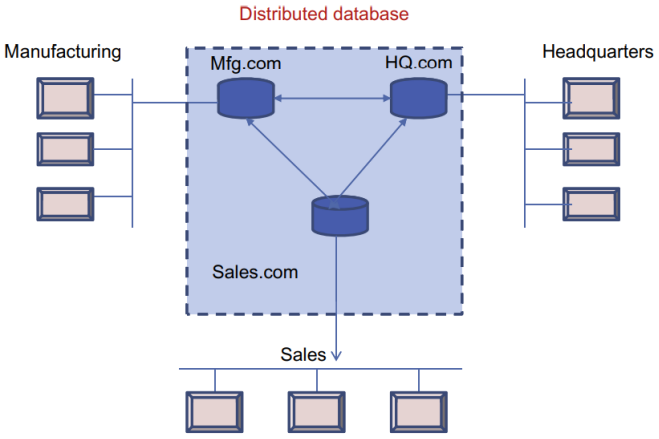


Figure 1: Homogeneous Distributed Database

1.2.2 Heterogeneous Distributed Database

In a heterogeneous distributed database, different sites can use different schema and software that can lead to problems in query processing and transactions. Also, a particular site might be completely unaware of the other sites. Different computers may use a different operating system, a different database application. They may even use different data models for the database. Hence, translations are required for different sites to communicate.

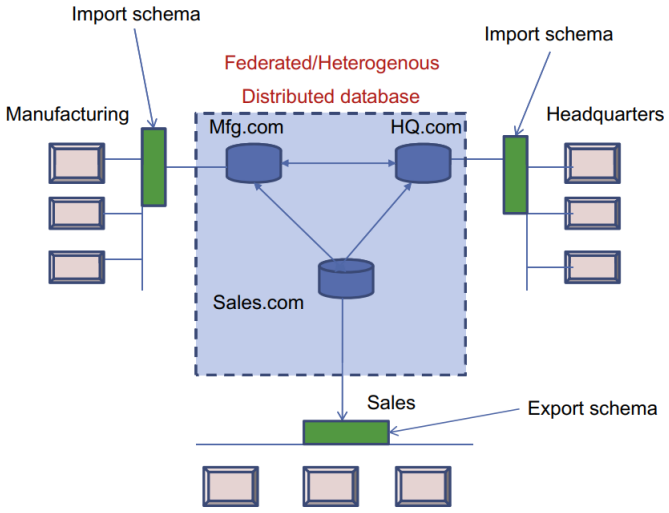


Figure 2: Heterogeneous Distributed Database

1.3 Distributed Data Storage

There are two ways in which data can be stored on different sites. These are: **replication** and **fragmentation**

1.3.1 Replication

In this approach, the entire relationship is stored redundantly at 2 or more sites. If the entire database is available at all sites, it is a fully redundant database. Hence, in replication, systems maintain copies of data.

This is advantageous as it increases the availability of data at different sites. Also, now query requests can be processed in parallel.

However, it has certain disadvantages as well. Data needs to be constantly updated. Any change made at one site needs to be recorded at every site that relation is stored or else it may lead to inconsistency. This is a lot of overhead. Also, concurrency control becomes way more complex as concurrent access now needs to be checked over a number of sites.

1.3.2 Fragmentation

In this approach, the relations are fragmented (i.e., they're divided into smaller parts) and each of the fragments is stored in different sites where they're required. It must be made sure that the fragments are such that they can be used to reconstruct the original relation (i.e, there isn't any loss of data).

Fragmentation is advantageous as it doesn't create copies of data, consistency is not a problem.

Fragmentation of relations can be done in two ways:

- **Horizontal fragmentation – Splitting by rows:** The relation is fragmented into groups of tuples so that each tuple is assigned to at least one fragment.
- **Vertical fragmentation – Splitting by columns:** The schema of the relation is divided

into smaller schemas. Each fragment must contain a common candidate key so as to ensure a lossless join.

In certain cases, an approach that is a hybrid of fragmentation and replication is used.

1.4 The Advantages and Disadvantages of Distributed Database

In this section, we will introduce about the benefits and the blockages of using Distributed Database

1.4.1 The Advantages of Distributed Database

The Distributed Database has brought a lot of benefits for database management system, such as:

- The database is easier to expand as it is already spread across multiple systems and it is not too complicated to add a system.
- The distributed database can have the data arranged according to different levels of transparency i.e data with different transparency levels can be stored at different locations.
- The database can be stored according to the departmental information in an organization. In that case, it is easier for organizational hierarchical access.
- There were a natural catastrophe such as fire or an earthquake all the data would not be destroyed it is stored at different locations.
- It is cheaper to create a network of systems containing a part of the database. This database can also be easily increased or decreased.
- Even if some of the data nodes go offline, the rest of the database can continue its normal functions.

1.4.2 The Disadvantages of Distributed Database

However, there are still a lot of difficulties while using distributed database, for example:

- The distributed database is quite complex and it is difficult to make sure that a user gets a uniform view of the database because it is spread across multiple locations.
- This database is more expensive as it is complex and hence, difficult to maintain.
- It is difficult to provide security in a distributed database as the database needs to be secured at all the locations it is stored. Moreover, the infrastructure connecting all the nodes in a distributed database also needs to be secured.
- It is difficult to maintain data integrity in the distributed database because of its nature. There can also be data redundancy in the database as it is stored at multiple locations.
- The distributed database is complicated and it is difficult to find people with the necessary experience who can manage and maintain it.

1.5 Distributed Database and Centralized Database

Opposite to distributed database, the concept of centralized database focuses on a database that is located, stored, and maintained in a single location.

These two kind databases have some differences in general, and are shown in the table below.

Centralized Database	Distributed Database
It is a database that is stored, located as well as maintained at a single location only.	It is a database which consists of multiple databases which are connected with each other and are spread across different physical locations.
The data access time in the case of multiple users is more in a centralized database.	The data access time in the case of multiple users is less in a distributed database.
The management, modification, and backup of this database are easier as the entire data is present at the same location.	The management, modification, and backup of this database are very difficult as it is spread across different physical locations.
This database provides a uniform and complete view to the user.	Since it is spread across different locations thus it is difficult to provide a uniform view to the user.
This database has more data consistency in comparison to distributed databases.	This database may have some data replications thus data consistency is less.
The users cannot access the database in case database failure occurs.	In distributed databases, if one database fails users have access to other databases.
Centralized database is less costly.	This database is very expensive.

1.6 Objective

In this project, our aim is to build a distributed database by using **homogeneous distributed database** and using **horizontal fragmentation** for storing distributed data.

2 Analysis and Design

2.1 Storing Distributed Data by Horizontal Fragmentation

The full table is the most common and complete form of the fragmentation. However, there are possible situations where the tables could be broken down and distributed to different locations. For example, a company's personnel files could be allocated to the location where the employees work. In such situations, horizontal fragment could be used to store information.

Horizontal fragment is the selection of a subset of the rows of a table that when distributed become the table definition and content at the distributed location. This type of fragmentation is often controlled and/or guided by key attribute values

2.2 The Homogeneous Distributed Database

When a distributed data model refers to data that are fragmented on similar devices in its geographic distribution, it is called a homogeneous model. Homogeneous distributed designs are much easier to implement than others because everything is consistent and has no need for translation or reformatting.

Within this new homogeneous distributed model we have but two concerns. The first is to ensure that we have analyzed and fragmented the overall schema to build detail schemas for the local databases so they service the local work need. The data needs of the local users will take priority over the centralized users if we want to maintain the autonomy that is so highly prized.

The second consideration is that we have to fill the global schema with all of the information from the local database schemas so all the transaction activity can be directed appropriately. This ensures that the centralized users get what they need without victimizing the local sites for either control or performance.

2.3 The Database Schema

With the project of implementing distributed database, we use a simple database of student enrollment course.

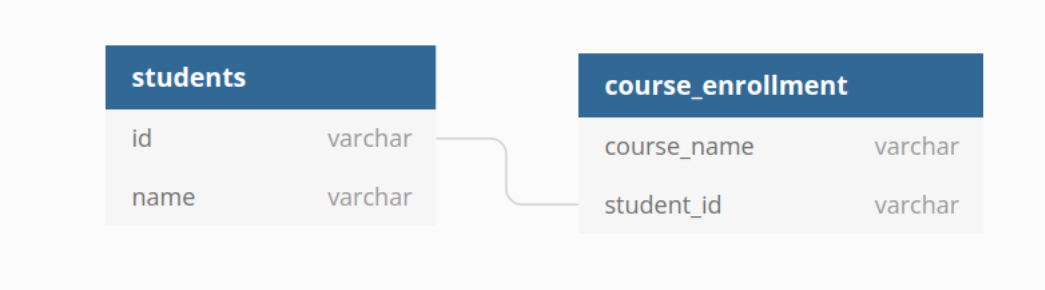


Figure 3: Database Schema

The following table describes the individual mission in each table

Table	Description
Course Enrollment	Contains the enrollment of students in each course
Students	Contains the name of students

2.4 The Database Architecture

In total, we have 5 databases. These small database instances are easier to manage as most of the data exists in the separate database worker servers. Coordinators hold smaller amounts of data, like some metadata and data that is not sensible to shard.

3 Implementation

In this section, we will introduce our tools and techniques, also present how we do our project specifically.

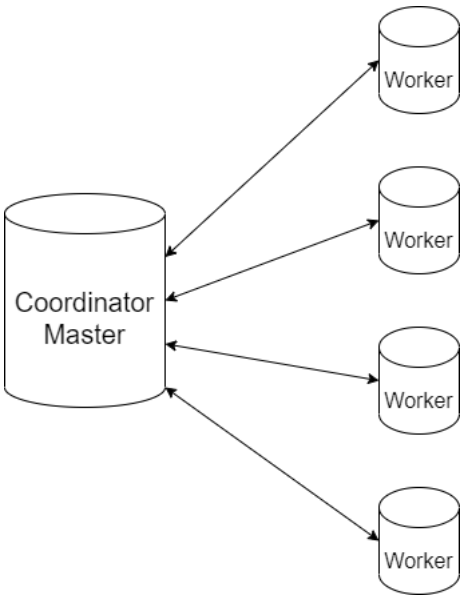


Figure 4: Database Architecture

3.1 The Tools and Techniques

3.1.1 PostgreSQL

PostgreSQL is a powerful, open source object-relational database system with a strong reputation for reliability and performance.

3.1.2 Python

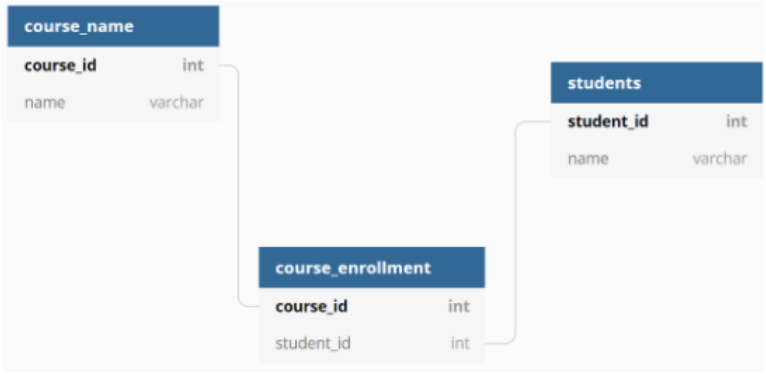
Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small- and large-scale projects.

In our project, we use Python's extension Pyscopg. Psycopg is the most popular PostgreSQL database adapter for the Python programming language. Its main features are the complete implementation of the Python DB API 2.0 specification and the thread safety (several threads can share the same connection). It was designed for heavily multi-threaded applications that create and destroy lots of cursors and make a large number of concurrent "INSERT"s or "UPDATE"s.

3.2 The Implementation

3.2.1 First Implementation

At first, we try to use PostgreSQL and Citus - an open source extension that transforms Postgres into a distributed database. The result is quite good, however it does not show what we have learned in the course, therefore we have tried another method, which is shown in the next section.



(a) The database schema

```
postgres=# SELECT * FROM course_enrollment;
 course_id | student_id
-----------+------------
         1 | BI10-154
         1 | BI10-098
         2 | BI10-098
         1 | BI10-195
         1 | BI10-146
         2 | BI10-146
         1 | BI10-128
         2 | BI10-128
(8 rows)
```

(b) Example of using Citus and PostgreSQL

Figure 5: First Implementation

3.2.2 Second Implementaion

In the second implementation, we have tried to create our own distributed database, with the help of PostgreSQL and Python. Due to some limit of time, we cannot perform the database as the original one since we just have done the join two distributed tables, not three in the previous implementation. The project structure is described as below:

- commands.py: Creates sql command templates.

- init.py: Creates connection to clusters. It also keep the connection with the database, also create metadata tables on coordinator if there doesn't exist.
- node-config.py: Stores clusters connection info. If we want to add more workers, we just simply add the information of the workers (e.g. dbname, user, password, host and post)
- querry-planner: Distributes query to workers nodes, receives the command and gives the results.
- utils.py: utility functions (e.g. runs the command, creates distributed tables, etc.)

Here is the setup of the project:

- Create some postgres cluster
- Include cluster connection info in node-config.py
- Run main.py to test the project

4 Results

The results of what we have done are shown below, with figures and descriptions.

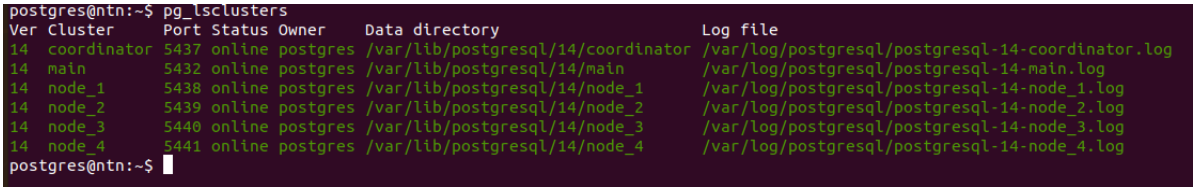


Figure 6: The workers and coordinator

```
ntn@ntn:~/Desktop/ds2022/Distributed Database$ python3 main.py
Connecting to coordinator node.
Connection established
Connecting to worker: node_1
Connection established
Connecting to worker: node_2
Connection established
Connecting to worker: node_3
Connection established
Connecting to worker: node_4
Connection established
Checking shards_table existance.
shards_table existed.
Checking distribution_table existance.
distribution_table existed.
This table has already been distributed before
This table has already been distributed before
Choose an options below:
    1. View all student
    2. Create a new student
    3. Show course enrollment
    4. Enroll a student
    5. Delete a student by id
    6. Exit
```

Figure 7: Secure the connection to workers/coordinator

```
Choose an options below:
    1. View all student
    2. Create a new student
    3. Show course enrollment
    4. Enroll a student
    5. Delete a student by id
    6. Exit

1
('BI10-195', 'Nguyen Quang Vinh')
('BI10-098', 'Mai Xuan Hieu')
('BI10-154', 'Nguyen Tuong Quynh')
('BI10-146', 'Nguyen Anh Quan')
('BI10-128', 'Nguyen Tran Nguyen')
('BI10-158', 'Nguyen Khang Thai')
```

Figure 8: Function 1 - View all student

```
Choose an options below:
    1. View all student
    2. Create a new student
    3. Show course enrollment
    4. Enroll a student
    5. Delete a student by id
    6. Exit

2
Enter student id: BI10-149
Enter student name: Tran Hong Quan
Inserted into node node_4, shard students_7
```

Figure 9: Function 2 - Create a new student

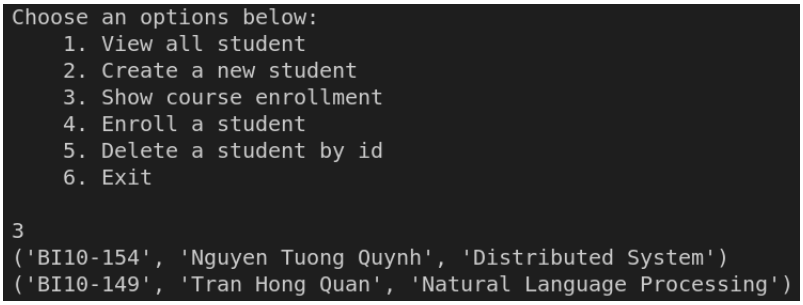


Figure 10: Function 3 - Show course enrollment

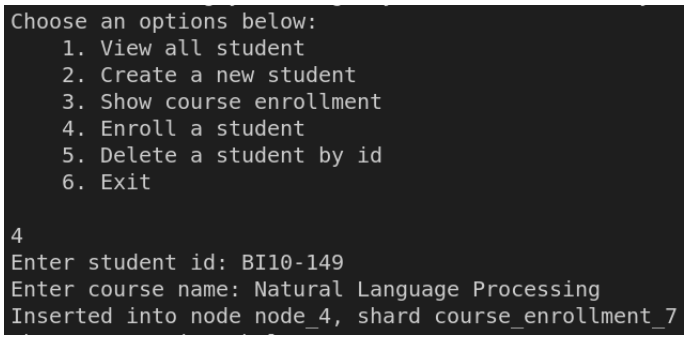


Figure 11: Function 4 - Enroll a student

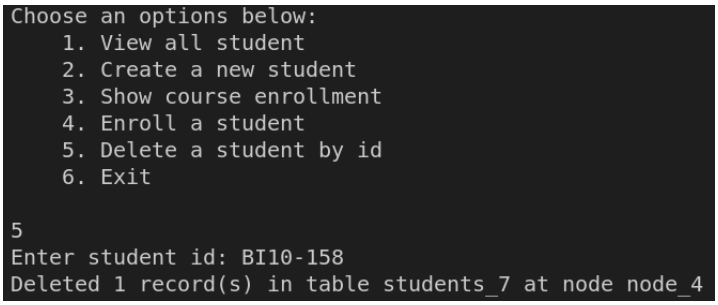


Figure 12: Function 5 - Delete a student by ID

5 Conclusion and Future Works

5.1 Conclusion

In conclusion, this report has shown specific details and information in regard to the implementation of managing student course enrollment by using distributed database, with the help of PostgreSQL and Python. The distributed database is successfully at doing and providing several needed tasks for viewing and querying student course enrollment in each individual worker.

5.2 Future Works

Many adaptations have been left for the future due to lack of time, such as the setting up more join of distributed tables (currently there are two tables joining). More additional features can be developed in the future, like the creation of replication, which will be beneficial in case some workers are broke down.