WILEY

Document Based Database

Candidate : Bahaa Abualrub

Feb 2020

Contents

1 Introduction

This report discusses the design and implementation of a document based database with documents saved as JSON objects. While showing best practices with efficient and clean code written in JAVA.

The report starts with the technologies and devops tools used in building this project. Then it shows the details of the implementation and defends it against the Clean Code, Effective Java rules and SOLID principles.

Finally it points to some of the issues with this design and solutions that should be implemented in the future.

2 Implementation Details

2.1 Technologies

The database was built using :

- Intillij Idea Community Edition.
- Java Spring Boot.

2.2 Design Patterns

- **Singleton Pattern :**

  This creational design pattern is used when a class shouldn’t have more than one instance in the project.

  It is used in different locations in the code, one example is the indexManager class as shown in figure 1 below.

```java
public class IndexManager {
    private static IndexManager instance = new IndexManager();
    private HashMap<String,HashMap<String, BPlusTree<String>>> indexes;

    private IndexManager(){...}

    public static IndexManager getInstance() { return instance; }
```

Figure 1: Singleton Pattern In Index Manager Class

- **Observer Pattern :**

  This behavioral design pattern is used when we want to notify objects(observers) based on changes that happen to the observable object.

  The observer pattern is needed in our design since we need to update the database copies in the reading nodes when an administrative task happened (write,delete..). This was implemented in classes ReadNode(Observer) and LoadBallancer(Observable) as shown in figure 2.

```java
package com.db.node;

import ...

public class ReadNode implements java.util.Observer {
    private final String url;
    public ReadNode(String url) { this.url = url; }

    @Override
    public void update(Observable o, Object arg) {...}
}
```

```java
package com.db.node;

import java.util.Observable;

public class LoadBalancer extends Observable {

    public void scale(){...}

    public void update(){...}

    public void unScale(){...}
}
```

Figure 2: Observer and Observable Classes

2.3 Data Structures

- **HashMap**

  HashMap is a powerful data structure that hashes objects in a key-value structure using hash tables. It’s used a lot in the code since it’s the most efficient way to map objects to other objects (ex. Mapping each schema to it’s indexes).

- **Tree Set**

  Tree Set is java’s implementation of red-black tree which is a self-balancing binary tree. This data structure is great for fast insert/delete (see table 1 below) .

Table 1: TreeSet Operations Time Complexity

	add	delete	search
Time Compelxity	ln(N)	ln(N)	ln(N)

Tree Set is used in class DuplicateHandler to handle duplicate values of indexed attributes to make sure that read/write operations are as fast as possible.

- **B+ Tree**

  This is an extension of B Tree which is a tree structure that makes levels of indexing dynamically. B+ Tree is faster than B Tree and makes the deletion operation easier. In this tree, the following most hold:

  - Each node contains ***m*** keys.
  - The root has 2 to ***m*** children.
  - Any other node must have at least ***m/2*** children.

  Since this structure is great for multi-level indexing, it's used with the Tree Set to make indexes for attributes in our database. For each index there must be a B+ Tree that represents it, and makes accessing the data faster. (See Figure 3 below).

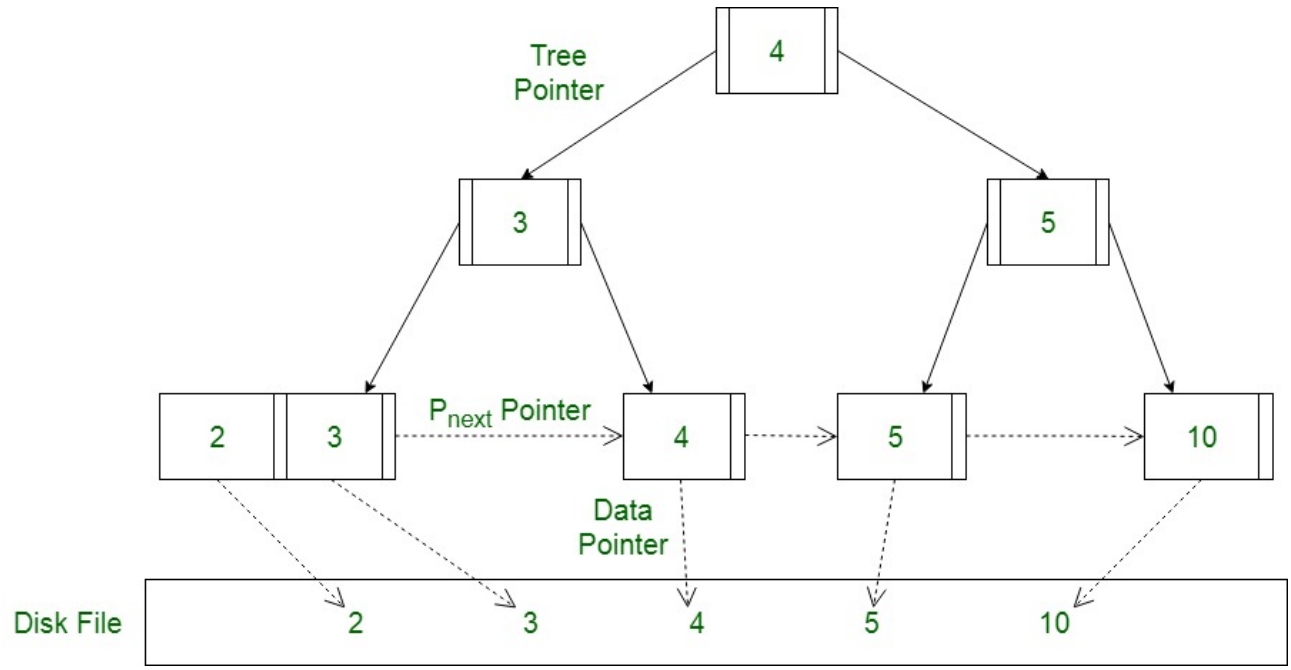


Figure 3: B+ Tree Structure

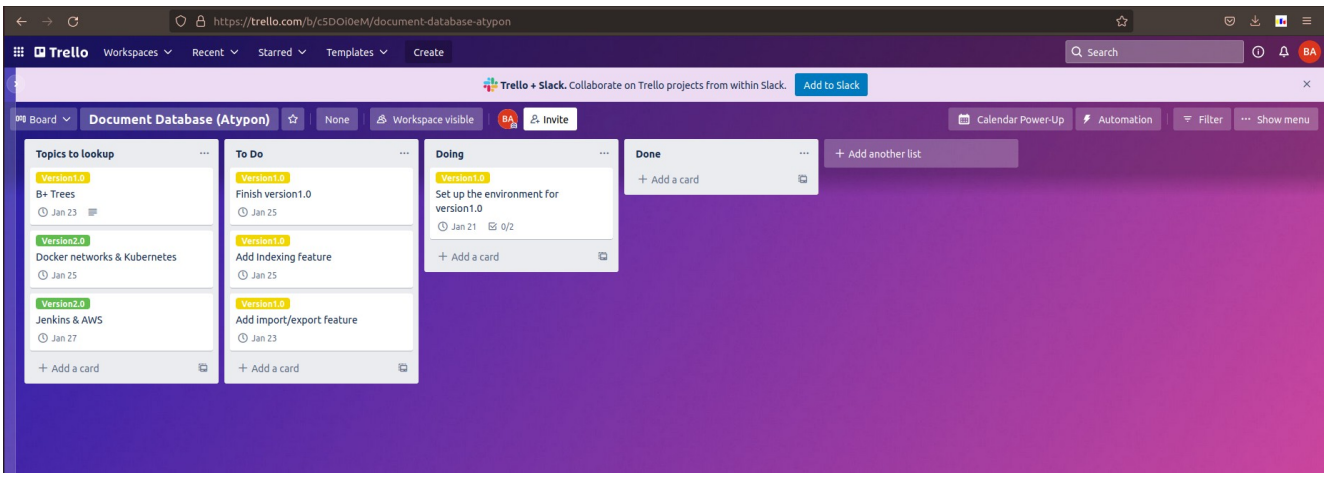
2.4 Database Protocol

The communication between nodes inside the cluster and the communication between database and client are done with RESTful apis. The apis are documented using **Postman Collection** which is provided with the other files.

4 DevOps

4.1 Trello

Trello is a visual work management tool that helps organizing the tasks and tracking progress. I used it to manage this project and be able to evaluate the status of it(see Figure 4 below).



4.2 Git / GitHub

Git is the most famous version control system that is used to handle contributions in project. With GitHub as the hosting platform I made a repository for this project, you can find it here : https://github.com/Bahaa55/Document-Database-Atypon

4.3 Maven

Maven is a build automation tool that handles the dependencies of the project. I used it in Intillij Idea community edition to make Java web application.

4 Clean Code

4.1 Comments

- **C1: *Inappropriate Information***
- **C2: *Obsolete Comment***
- **C3: *Redundant Comment***
- **C4: *Poorly Written Comment***
- **C5: *Commented-Out Code***

The code doesn't contain any of the five bad comments examples listed above.

4.2 Environment

- **E1: *Build Requires More Than One Step***

  This is solved by using Maven, which insures that all dependencies are handled and the run only requires one maven command (mvn clean spring-boot:run).

- **E2: *Tests Require More Than One step***

  There’s no testing. Only the api’s documentation that is provided as **Postman Collection.**

4.3 Functions

- **F1:** ***Too Many Arguments***

  This is avoided since the longest function requires three parameters.

- **F2:** ***Output Arguments***

  This is avoided, all functions are clear that they require input arguments.

- **F3:** ***Flag Arguments***

  This is avoided since no function has a flag argument.

- **F4:** ***Dead Function***

  This is avoided, all methods are important for the project to work successfully.

4.4 General

- **G1:** ***Multiple Languages in One Source File***

  The project is written only in Java.

- **G2: *Obvious Behavior Is Unimplemented***

- **G4: *Overridden Safeties***

  There’s no override to safeties in the code.

- **G5: *Duplication***

  I made sure that the code contains as less duplication as possible.

- **G9: *Dead Code***

  There’s no dead code anywhere in the project.

- **G10: *Vertical Separation***