NoSQL Database

Atypon Java / DevOps Training: Winter 2022

Khalid Nusserat

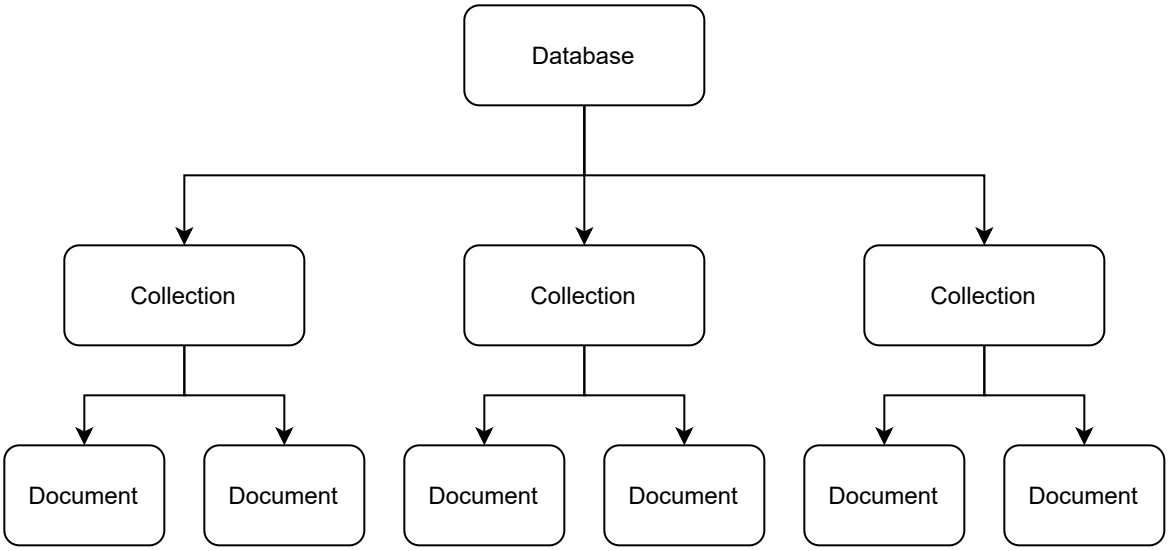
Introduction

The goal in this project is to design a non-relation document based database, similar to already existing solutions such as Mongodb. There are a number of constraints that must be taken into consideration during the design and the implementation:

- Reads are much more frequent that writes.
- The system is composed of a number of nodes, all of which store the exact same data, however one of these nodes acts as a primary node and is the only node allowed to perform writes, and it can also be responsible for additional management tasks. The primary node *cannot* perform reads, since these must be delegated to any of the secondary nodes for performance reasons.
- The other nodes are called secondary nodes, and are responsible for responding to read requests. They receive the latest updated version of the database by synchronising with the primary node, since the primary node at all times contain the latest correct version of the database, since it is the only node that is allowed to perform writes.
- Each document must have a schema that defines the document structure.
- Each document must have a unique ID that is efficiently indexed.
- The database must support creating either a single property index, or -*optionally*- a multi-property index.
- The database should be able to scale horizontally be replicating the secondary nodes and then balancing the requests between these replicas.
- Allows for the database schemas to be exported and imported.
- The system supports three types of users:
  - Default admin: Can access the entire database and manage the roles of all the users.
  - Admin: Can only access some the things that the default admin can access.
  - User: Can only do reads and writes on some collections.
- We are also required to create a demo application to demonstrate our final product.

Overview of the design

The documents in the database are stored following this hierarchy:



The database is composed of a number of *collections*, each of which contains a varying, non-limited number of *documents*, which are JSON files that contain the user stored data.

Each collection is only allowed to store documents that adhere to the same *scehma*, which defines the structure of the document.

In my design, I opted to abstract away as much as possible the implementation of the document from the rest of the system, to avoid making the implementation of document and the rest of the system tightly coupled. This was done by introducing the interface `Document`, which defines the functionalities that any document class must have, such as the ability to access a specific field and such.

The rest of the implementation depends on this interface rather than on a concrete implementation, therefore any future changes would not require us to change other parts of the implementation.