DATA70141 Understanding Databases 2022-23

Individual Report

NoSQL Database for Amazon Online Shopping

European Expansion

Yanfei Shan: 11118569

1. Personal Role

Firstly, I was involved in the decision-making for database schema design. I suggested dividing the *shopping basket* into *fresh product* and *other product*. Fresh product and other product are in the different warehouse (one in the Morrison warehouse, one in the Amazon warehouse), and they are delivered in different ways, so it is not convenient to put them in the same shopping basket for later shipping fee computing and partner allocation. Meanwhile, we divided the *current order* collection into *fresh* and *other*.

Then I participated in data insertion. I was responsible for inserting *Morrisons' Daily Inventory* and *Partner Status*. We assumed that products' stock varies from warehouse to warehouse, indicating that customers may have to choose another warehouse for their ideal items. To implement this assumption, I manually updated some items with 0 in stock and used two for loops to make *the final number of inserts = the number of total products * the number of warehouses*.

In addition, I proposed the computing method of shipping cost, and did my part in report writing and schema drawing.

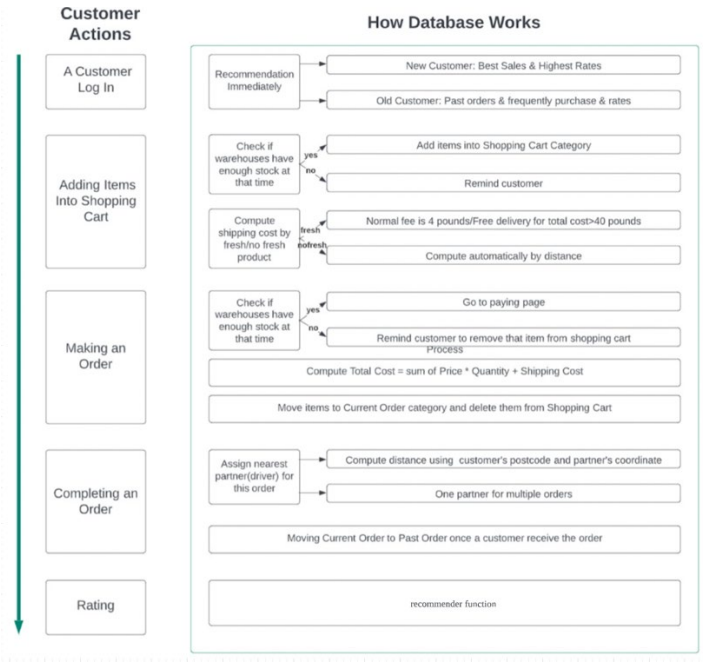

Figure 1: Flow chart of customer actions and corresponding database functions

2. Reflection from Amazon Database Project

What I have learned from database design is the use of *reference* and *embedded document*:

- For limited data, which is not easily changed and are often retrieved, we'd better to use *embedded document*. For example, the *address* for *Customer*, the *basket item* of *Shopping Basket*.

- By using two references, we can determine a particular document. For example, after knowing the *product id* and the *warehouse id*, we can know the stock of a product in a certain warehouse.

Another thing I benefit from this project is commanding how to connect to MongoDB from python and applying operations based on python.

Designing database is the most challenging part for me, because all subsequent functions and queries are based on this part. There are many ways in which database schema can be implemented, but it is more difficult to make optimization.

3. Multi Leader Replication

Suppose the company sets up a new data centre in Europe. In that case, I'll suggest having a *multi-leader replication*, which means setting the new data centre as a new leader apart from the original leader in the UK.

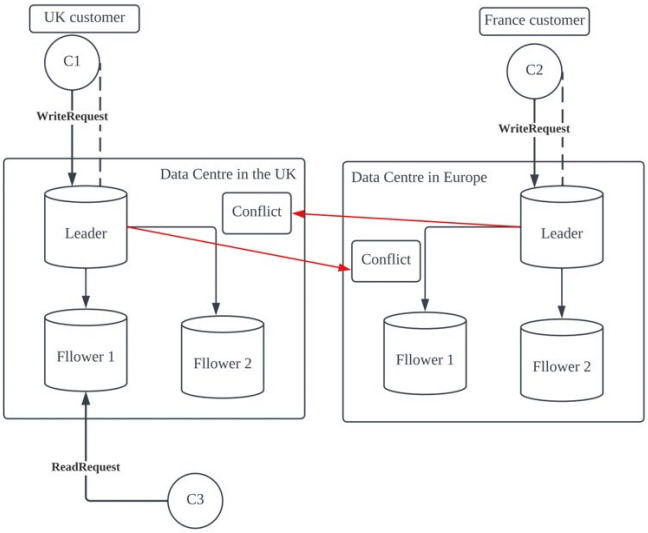

Figure 2: Demonstration of multiple-leader-replication for new data centre

This replication will bring two benefits: First it will save some latency and reduce the response time of the servers. If we take single-leader replication, a user in France should send requests to the data centre in the UK. He may face network delays. By taking the European centre as a leader, the user in France can send the request to the European data centre nearby, which can solve the network latency problem.

Secondly, it keeps the data centre independence. If the system in the UK does not work for some reason, then the data centre in Europe can continue to accept requests from all users, which means the dataset will have consistently good performance and remain correct and complete.

4. Key Range Based Partition and Sharding

With the development of Amazon business in Europe, more products and customers from more countries are added to our dataset. In order to handle the increase load and

to increase overall storage capacity, we need to implement partition and sharding on our large database.

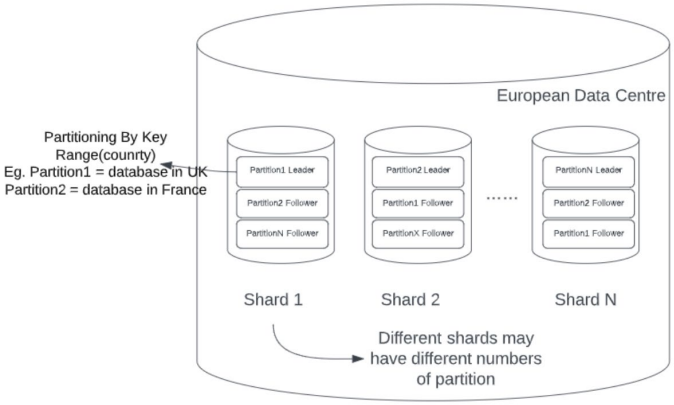

Figure 3: Demonstration of partition and allocation for new European data centre

For the newly built European data centre, the database is partitioned by *Country*. Eg. *Customer*, *Warehouse*, and other collections from the same country will be placed in the same partition.

Several reasons are given: 1. the number of countries is limited. If Amazon runs business via 10 European countries, the data centre will be separated into 10 partitions. 2. Cross-shard queries are super-expensive. Customers tend to shop only in their country, and partitioning by country allows the query to be retrieved in only one shard. If we set Character(A-Z) as the key range, there will be lots of cross-shard operations when implementing basic purchase operations.

Since we are going to have a large database, we should do replication for every partition as a back up to minimize the maintenance time and improve the stability of our database.

Let's say we back up Partition1 on shard2 as Partition1 Follower and only send write requests to Partition1 Lead on shard1. But in the event of a problem with the partition1 lead, we will use Partition1 Follower to fix it. Number of partitions on each shard is based on the shard's own capacity. We try to back up as much data as possible on two or more shards to improve database availability.

5. Re-design of the database

When company expanding its operations to some EU countries, we should slightly change the database design to accommodate the new business model. Since the business is not just local to the UK anymore, we need to consider a scenario: If a customer in France is doing a search on our extension platform, he doesn't expect to get the products out of France. This means we need to add the *Country* attribute to several collections: *Customer*; *Product*; *Warehouse*.

Country in the *Customer* collection indicates the location of the customer. *Country* in the *Product* collection indicates the location of the product. It is different from the *Country of Origin* attribute. For example, a product is produced in the UK, but was transferred to France for sale. So, the *Country* attribute of this product is France, but

the *Country of Origin* is UK. *Country* in the Warehouse collection indicates the location of the warehouse.

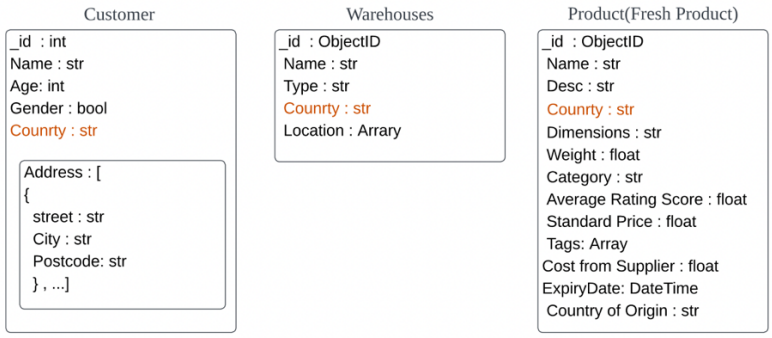

Figure 4: Changed collections' structure

When a customer searches on the platform, we first obtain the customer's *country*, then use this country to filter the *Product* collection, and then search for the *keywords* entered by the customer in the filter results, so that we can obtain the product in the same country.

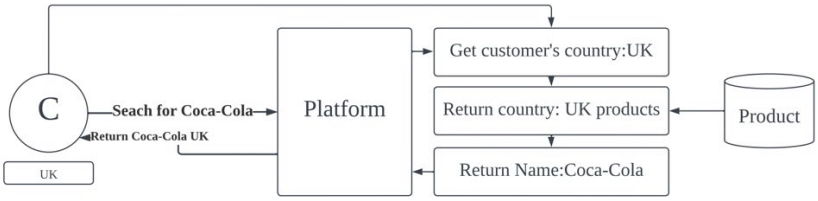

Figure 5: Flow chart of customer search under new platform

Currency issues also need to be considered. The Pound sterling(£) is used in the UK and the Euro(€) in the European region. This means that when we insert data, we must initially enter the product as two different products, such as Coca-Cola, which are both from UK, but they are actually two different products with different *_ids* and *prices*.

Figure 6: Same product in different countries

We don't need to add the *Country* attribute to the *rating related collection* because for the same products from different countries, they use different *_ids*, then we will get different rates.

Of course, with the business expansion, I would like to add a little detail in the customer: IfMembership, the customer who is the membership, can enjoy the free shipping cost. And we will develop a new recommendation strategy for them to improve customer loyalty.

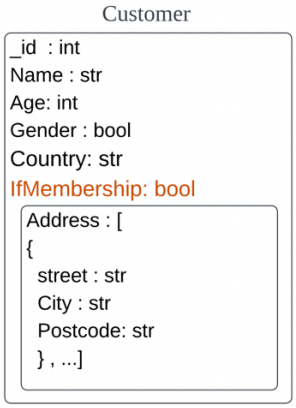

Figure 7: Changed customer collection