**Natural high availability support**

The request from the computer room is sent to the main mongod database, and the data of the main database is synchronized to the two slave databases.

So for example, you have a server with only one mongoDB installed on it, but if the server suddenly breaks down, or if it loses power, or if the hardware fails, then your service will not be available once. As shown in the figure, there are three here, one mongoDB master, and two mongoDB slave. After the mongoDB master hangs, it still has two slave, and these two slave can detect the master. It hangs up, so it will choose one of the slaves to go up and replace the position of the main.

So it's called high availability, then, so that the services you depend on don't break.

**Data Consistency and Security Guarantee**

All client requests of users are requested to the master, and then the master synchronizes the logs to the two slave, but the master does not accept the requests of these two slave.

But what might this situation lead to? This may cause the two repositories not to receive the request written by your master database. For example, the master database receives a record and writes a record, but it sends the data to the two slave databases. When the two slave are for various other reasons, for example, the service of them is restarted, or the machine is restarted, they do not record this one record, and then the main will ignore it and respond directly. At this time, your slave database will be different from your master database data.

**Model one-to -one**

Here I downed the MovieLens 10M dataset, which consists of 10million movie ratings from “grouplens.org” web site as an example [5]. This source provides tree related datasets, and they are ratings, movies and tags.

This dataset divides a large amount of data into these three different files because of the "schema" limitation of the data and in order to let the downloader understand the data more clearly. When we read the data details, we will find that these three data documents have several fields that are the same and can be merged. MongoDB's One-to-One Relationships model can better organize this dataset.

In this way, when merging data, developers can not only quickly clean up the data and combine multiple documents into one document, but also avoid the repetition of movie information and reduce the difficulty of data cleaning. What’s more, after we finish inserting data, the subsequent search work becomes faster and more convenient.

**Model One-to-Many Relationships with Embedded Documents**

Using the same example as the previous model, a movie will be rated differently by different people at different times, and the information is scattered.

With this model, we can better aggregate the data. When we want to get the average rating of a movie, we don't need to search and recalculate in two datasets, we just need to search the movie between the MongoDB databases to get all the ratings for a single movie.

**Model One-to-Many Relationships with Document References**

This model “uses references between documents to describe one-to-many relationships between connected data. [4]”

When we sort out the reviews of movies, we find that there are many movies of the same genre with the same ratings. To aggregate movies of the same genre and with the same rating, we can use this model.