William Austin

06/28/2023

WEB 335

Discussion 6.1 - Indexes

Indexes in MongoDB are similar to a book’s index. (Chodorow, 2017). If you’re looking for particular information in a book with an idea, you first search the index, usually a much smaller amount of info, for the topic, or keyword you’re looking for. That book index would give you the pages in the book that are relevant to that topic or keyword. This is the same idea in MongoDB indexes. If you query MongoDB, it is helpful for it to know where to look.

The main benefit of indexes is speed. An unindexed collection can lead to long query times, and even critical failures if the query takes too long. If we limit where MongoDB has to search we can speed up the results and make everything more efficient. Let’s say you have a Database of all Instagram users, and one user wants to find another user by searching their “username”. If Instagram had an unindexed collection of users, it would have to do a full collection scan of millions of users to find the one user. If we index our Instagram user collection by user name, MongoDB could perform this same search almost instantaneously. Our index has told MongoDB exactly where to look.

We create indexes with built in `createIndex()` method. The developer then tells MongoDB the parameters of the index and other possible indexing options. (Chodorow, 2017). There are multiple ways of indexing, including compound indexes (two or more keys), or sparse indexes.

Sparse indexes are those where you have a field in your collection documents that may or may not exist. The best way to explain this is through an example. Let’s say you have a database of students, 9th - 12th grade. One of the fields of the student documents is ‘GPA’. Now, the entering freshman don’t have a GPA yet, it would be a null value or missing from those documents in the collection. So if we created an index by GPA we would be creating a sparse or partial index. (MongoDB). This means that not every document in the collection is in the index. It is still a useful way to index students however, as we need to see GPAs to know who to enroll in which honors classes, or AP classes or what have you.

Generally speaking, we want to index based on things that will make the number of documents searched lower. Let’s say you have your collection of Instagram users again, and we index by gender and name (a terrible idea). When I search for a particular user’s name, MongoDB would do the first key first, it would first find all the women on instagram (about half of the total users) and then it would search for the user name. Gender is said to have ‘low cardinality’. That field doesn’t narrow our search down much. Username is said to have ‘high cardinality’. That field is unique to every user. Something like age or location would be somewhere in between. (Chodorow, 2017).

Capped Collections are a feature of some collections on MongoDB where only a set amount of documents are allowed. So lets say you capped your Yelp page to only keep 100 of the most recent reviews. As new reviews were added, Mongo would delete the old ones, starting with the oldest. It is a good way to limit the amount of data in a given collection. Especially, as old documents become obsolete or immaterial.

Sources

CHODOROW, K. (2017). *MONGODB: The definitive guide*. O’REILLY MEDIA, INC, USA.

MongoDB. (n.d.). *Sparse Indexes*. MongoDB Manual. https://www.mongodb.com/docs/manual/core/index-sparse/