Jonathan Roland

WEB 335 Introduction to NoSQL

Instructor: Prof. Krasso

11/12/2020

Discussion 5.1

1. **What is the Bulk Insert operation and how does it work?**

When we want to add new documents to a MongoDB collection, we use inserts. Since version 3.0, MongoDB supports both insertOne() and insertMany() operations. Self-describing, insertOne() allows a single document to be inserted. Using insertMany() allows bulk insert operations. This is when multiple documents are grouped in an array and submitted at once to save time by pushing large sets of documents in a single operation. Example:

db.guitars.insertMany([

{“brand” : ”Ibanez”, ”model” : ”RG”},

{“brand” : ”Fender”, ”model” : ”Stratocaster”},

{“brand” : ”Paul Reed Smith”, ”model” : ”Custom 24”},

]);

This does come with some risks if duplicate records are encountered during the insert. The user can select parameters to either skip over the duplicate and continue or stop on the first such error.

If no \_id is provided for these documents, one will be generated automatically for them.

1. **How do you avoid duplicate records when using the update operation?**

MongoDB follows atomic principles to resolve multiple updates on the same document. It will update in the order received, with the last document ‘winning’ as it is the last write. A common mistake is to use matching functions to feed multiple updates to the same \_id. This will not write any updates and throw an error. To avoid this, test matching functions to ensure they only return a unique document. Matching updates on the \_id key is a good approach to this problem.

1. **What are query conditionals and how are they used (provide at least one example, in code, of a query conditional)?**

Query conditionals are comparison operators we can use in a find() query to return values that match those criteria. They are keywords that represent traditional less-than, greater-than and equal-to type comparisons:

* $lt: < (less than)
* $lte: <= (less than or equal to)
* $gt: > (greater than)
* $gte: > (greater than or equal to)
* $ne: != (not equal)

For example, if we wanted to return all documents in a guitars collection where the price was under or equal to $1500, we could form our query like so:

db.guitars.find({“price”:{“$lte”:1500}});

If we wanted to get all guitars priced in a range from $750 to $1500, we could use:

Db.guitars.find({“price”:{“$gte”:750,”$lte”:1500}});

If we wanted to exclude Ibanez guitars from the above search:

Db.guitars.find({“price”:{“$gte”:750,”$lte”:1500},”brand”:{“$nte”:”Ibanez”}});

Reference:

Bradshaw, S., Brazil, E., & Chodorow, K. (2019). *MongoDB: The Definitive Guide: Powerful and Scalable Data Storage* (3rd ed.). O’Reilly Media.