Hybrid Apps with MongoDB & Hadoop

Chris Harris, 10gen – BCS SPA Conference, 4 July 2012

MongoDB

A SQL database that stores JSON documents. Each row in a table is one document.

Where a conventional SQL database might have tables Book, Author, BookAuthor etc., the same data in MongoDB is a single JSON document. You can have multiple values for any field, add properties ad hoc etc.

Starting up a database instance:

```
./mongod --dbpath=/Users/xyz/…/data/node1
```

Starting up a shell:

```
./mongo
```

The shell is scriptable in Javascript. Commands;

```
> use spa                      // even if the spa database doesn’t exist yet!
> db.person.save({name : “Chris”})
```

This has now created the necessary database and table. No schema required!

```
> db.person.find()
```

Returns the Chris object

```
> db.person.save({name : “Fred”})
> t = db.person.find({name : “Chris”})
```

Returns just one record out of 2.

```
> t = db.person.findOne({name : “Fred”})
db.person.update({name : “Fred”}, {$set : {fr : 123123123}}, true, true) //
allow insertion; update all
```

Documents can be up to 60MB.

Common operations include indexing for rapid retrieval:

```
> db.person.find({name : “Tom”}).explain
```

This tells you that the query plan uses a scanner. Add a b-tree index:

```
> db.person.ensureIndex({name : 1})
```

You can sort and limit the number of results (e.g. to paginate the output):

```
> db.person.find().sort({name : 1}).limit(1).skip(1)
> db.person.find()
```

Returns a list of available commands.

Exercise 1

Insert 3000 student records:

```
for(i=0; i<1000; i++) {
  ['quiz', 'essay', 'exam'].forEach(function(name) {
    var score = Math.floor(Math.random() * 50) + 50;
    db.scores.insert({student: i, name: name, score: score});
  });
}
> show collections
scores
system.indexes
> db.scores.find().pretty()
```

Returns the first 30 or so records in pretty-printed format:

```
{
        "_id" : ObjectId("4ff446edba70aafb8359205d"),
        "student" : 0,
        "name" : "quiz",
        "score" : 60
}
{
        "_id" : ObjectId("4ff446edba70aafb8359205e"),
        "student" : 0,
        "name" : "essay",
        "score" : 74
}
```

Exercise 2

Now suppose you want to update everyone's grade to A if their score is over 90.

```
> db.scores.update({score : {$gte : 90}}, {$set : {grade : "A"}}, true, true)
> db.scores.find({grade : "A"}).limit(5).pretty()
```

There are no transactions in MongoDB. But updates and inserts are treated as atomic operations, while queries return a cursor that won't be affected by subsequent inserts, so you can rely on paginating the response.

Replication

MongoDB is protected by writing all updates to a master, which is replicated to slaves. The master is "elected" automatically as the "Primary". If the primary fails, a new primary is automatically elected. To become primary, a node must have the majority of votes – this eliminates the situation where two nodes in data centres that have been separated by a network failure both become masters. Administrators can override this. Votes can be biased by assigning each node a priority. E.g. nodes in a DR data centre can be prevented from ever electing themselves Primary (except if overridden by admin).

An “arbiter” node can be introduced, which is neither master nor slave but just has a vote for the primary.

Automatic recovery is supported.

Data is sharded for high performance (scalability) with high availability.

To start additional nodes on other port numbers:

```
./mongod --dbpath="x" --port=27018
./mongo
rs.initiate()
rs.status()
rs.add("xyz.local:27018") // up to 12
rs.status().pretty()
db.foo.save({x : 34})
```

When writing data to the database, you can specify how valuable the data is. By default, you get “fire and forget” – but you can specify w=2, meaning that the data has been secured to 2 instances before the call returns. You can also configure safeguards on connection pools, databases and collections.

Security Model

MongoDB applies security at the database level. Users either have read access, unlimited access or no access.

Hadoop

Chris showed the driver.

```
Mongo m = new Mongo()
DBCollection coll = m.getDB("words").getCollection("in");
InputStream is = etc.
```

Hadoop ran a map/reduce task that read all the words in the collection “in” and stored the occurrence count of each word in another collection called “out”.