Mapping SQL to MongoDB

This document expands on the properties collection created in the Java and Ruby training exercises.

1. Converting to MongoDB Terms

SQL Term	MongoDB Term
database	database
table	collection
index	index
row	document
column	field

2. Queries

Below are covered some of the main queries and how a query done against a database such as DB2 or Oracle is then done in MongoDB. This is just a selection of some of the possible queries.

a) Select

SQL Statement	MongoDB Statement
SELECT * FROM properties	db.properties.find()
SELECT DISTINCT name FROM properties	db.properties.distinct("name")
SELECT COUNT(*) FROM properties	db.properties.count()
SELECT * FROM properties ORDER BY name ASC	db.properties.find().sort({name: 1})
SELECT * FROM properties ORDER BY name DESC	db.properties.find().sort({name: -1})
SELECT name, title_no FROM properties	db.properties.find({}, {name: 1, title_no: 1, _id: 0})
SELECT * FROM properties WHERE name = ‘Simon’	db.properties.find({name: "Simon"})
SELECT * FROM properties WHERE name != ‘Simon’	db.properties.find({name: { $ne: "Simon"}})

SELECT COUNT(*) FROM properties WHERE name = ‘Simon’	db.properties.find({name: "Simon").count()
SELECT * FROM properties WHERE name = ‘Simon’ AND title_no = ‘ABC12345’	db.properties.find({name: "Simon", title_no: "ABC12345"})
SELECT * FROM properties WHERE name = ‘Simon’ OR title_no = ‘ABC12346’	db.properties.find({$or:[{name: "Simon"}, {title_no: "ABC12346"}]})
SELECT * FROM properties WHERE name LIKE ‘%Si%’	db.properties.find({name: /Si/})
SELECT * FROM properties WHERE name LIKE ‘Si%’	db.properties.find({name: /^Si/})
SELECT * FROM properties FETCH FIRST 3 rows only	db.properties.find().limit(3) if you only require 1 row you can use db.properties.findOne()

These following examples are not possible to be done against the properties database however these can be used if you had a number field such as age.

SQL Statement	**MongoDB Statement**
SELECT * FROM properties WHERE age = 30 ORDER BY name ASC	db.properties.find({age: 30}).sort({name: 1})
SELECT * FROM properties WHERE age > 30 AND age <= 40	db.properties.find({age: {gt: 30, $lte: 40}})
SELECT COUNT(AGE) FROM properties	db.properties.find({age: {$exists: true}}).count()

b) Insert

SQL Statement	**MongoDB Statement**

INSERT INTO properties (name, title_no) VALUES (‘Simon’,’ABC98765’)	db.properties.insert({name: "Simon", title_no: "ABC98765"})

c) Update

SQL Statement	MongoDB Statement
UPDATE properties SET name = ‘Dave’ WHERE title_no = ‘ABC12345’	db.properties.update({title_no: "ABC98765"}, { $set: {name: "Dave"}}, {multi: true})
UPDATE properties SET title_no = ‘ZYX12345’ WHERE name = ‘Simon’	db.properties.update({name: "Simon"}, { $set: {title_no: "ZYX12345"}}, {multi: true})

These following examples are not possible to be done against the properties database however these can be used if you had a number field such as age, and I have also used a status column.

SQL Statement	MongoDB Statement
UPDATE properties SET status = ‘C’ WHERE age > 25	db.properties.update({age: { $gt: 25 }}, { $set: {status: "C"}}, {multi: true})
UPDATE properties SET age = age + 2 WHERE status = ‘A’	db.properties.update({status: "A" }, { $inc: {age: 2}}, {multi: true})

d) Delete

SQL Statement	MongoDB Statement
DELETE FROM properties	db.properties.remove({})
DELETE FROM properties WHERE name = ‘Simon’	db.properties.remove({ name: "Simon"})