# MongoDB Quick Reference Sheet

## SQL Database Equivalent terms

Database ~ Database
Table ~ Collection
Index ~ Index
Row ~ Document
Column ~ Field
Joining ~ Embedding & Linking

## CRUD Equivalents

**C**reate = Insert
**R**ead = Find
**U**pdate = Update
**D**elete = Remove

## Common Shell Commands

**show dbs**
Shows available databases

**use foo**
Use a database named 'foo'

**db.createCollection('scores')**
Creates a collection named 'scores'

**show collections**
Shows collections in the database

**db.scores.stats()**
Get stats for the collection 'scores'

**db.scores.drop()**
Drops the collection 'scores'

currentOp() - info on running operations

**db.runCommand({getLastError: 1})**
Run the getLastError command + check if the last operation succeded.

## Example Shell session using JS:

*The mongo shell provides a Javascript API.*

To Update an object in the collection:
$ **obj = db.a.findOne(_id: 'foo')**
$ **obj.x = 123**
$ **db.a.update( { _id : obj._id }, obj )** OR
$ **db.a.save(obj)** - save() is a Mongo Shell helper func. equivalent to update() above.

## 1. INSERT

**db.scores.insert({name:"A", score:80})**
Insert into collection named 'scores'

## 2. FIND

**db.scores.find({"name":"A"}, {"_id":0})**
Find 'Alpha', exclude 'id' field from result

**db.scores.findOne()**
Find one random document

**sort(), skip(), limit()**
db.scores.find( {…} ).sort( score:1 ).skip(5).limit(1)

**db.places.find( {loc: { $near: [90,90] } } )**
Find Geospatial coordinates near (90,90)

***Query operators***

**$gt, $lt, $gte, $lte, $in, $ne, $nin**
db.scores.find( { score : { "$gt" : 10 } } )
db.scores.find( {score : { $in : [10, 20] } } )

**$regex, $exists, $type**
db.scores.find( { name : { $regex : "ph" }, email : { $exists: true } } )
- Find records where name contains 'ph' and that have email.

**$or, $and, $not, $nor (Logical)**
db.scores.find( { $or : [{ score : { $lt: 50 } }, { score: {$gt: 90} }] } )

**$all, $size** - used with array fields

***Cursor related options***

*(where, cursor = db.collection.find() )*

**cursor.count()** - count of docs in collection

**cursor.explain()** - info on the query plan

**cursor.hint()** - override default index selection with specific index.

## 3. UPDATE

*update() does wholesale update of a doc; Use 'multi' to update all matching docs.*

**db.users.update({name : "B"}, {$set : {age : 30}}, {multi : true})**

**$set()** - set field on an existing doc
**$unset()** - remove field from existing doc

**$upsert** - Adds new key if not present

**$inc, $sum**
db.users.update( {name : "Bob" }, { $inc : {age : 2} }, {multi : true} )

**$push, $pop, $pull, $pushAll, $pullAll, $addToSet** - Array Operators

**findAndModify()** - *Atomic,* compared to query & update operation

## 4. REMOVE

**db.document.remove({x : "foo"})**
**db.document.remove()** - whole document

## 5. INDEX

Types of Indexes: Single field, Multikey, Compound, Geospatial, Text, Hashed

**db.collection.ensureIndex({"name", 1})**
**-** Creates a **Single field Index** with key as 'name' and ascending order

**db.collection.ensureIndex({"x":1, y:-1})**
- **Compound Index** with 'x' ordered ascending and 'y' descending
*(Note: Compound indexes have lot of rules, be sure of what you're doing.)*

**db.collection.ensureIndex( {"coord" : '2d'} )** - Geospatial index over the coord field

**db.collection.dropIndex( { x : 1, y : -1 } )**

**db.collection.getIndexes()** - Get indexes for a collection

**db.collection.totalIndexSize()**

**db.collection.reIndex()**

**compact()** - defragment and rebuild index

***Index creation options***

**{unique: true}** - reject documents with duplicate value for the indexed field.

**{name: 'foo'}** - Custom name for index (if not given, name is derived from the key).

**{dropDups: true}**

**{background: true}**

**{sparse: true}** - Create entries only for documents having the index key.

## 6. AGGREGATE

*(Aggregation is used for advanced queries, by pipelining stages.)*

Stages that can be piplined -
$group, $match, $project, $unwind, $sort, $skip, $limit, $geoNear, $out, $redact

Key ref. from previous stage has $ prefix -
$match : {"key", "value"}

Pipeline stages appear in an array -
db.zips.aggregate( [ {$group : {"_id" : "$state", "population":{"$sum":"$pop"} } } ] )

**mapReduce()**
- Aggregation is faster than mapReduce.
- mapReduce can do some extra things.
- Hadoop's connector is available too.

## ADDITIONAL NOTES

**Replication** - for fault tolerance
**Sharding** - allows DB to spread over distributed servers
**Logging** - to check slow operations etc.
**Profiling** - Observe/Spot errors
**Monitoring** - mongotop, mongostat
Max. document size - 16Mb

**Write Concern**
From 2.6, the write methods integrate 'write concern' in the method execution. Previous versions used getLastError() to return error info.

**Important note on Language drivers**
Various language libraries have functions similar to Mongo shell functions, but mostly use the language's naming conventions:
**For instance, the equivalent for findOne() in *PyMongo* is find_one()**

---

Draft v0.1.3, for MongoDB 3.6

Send improvements @nitinnain