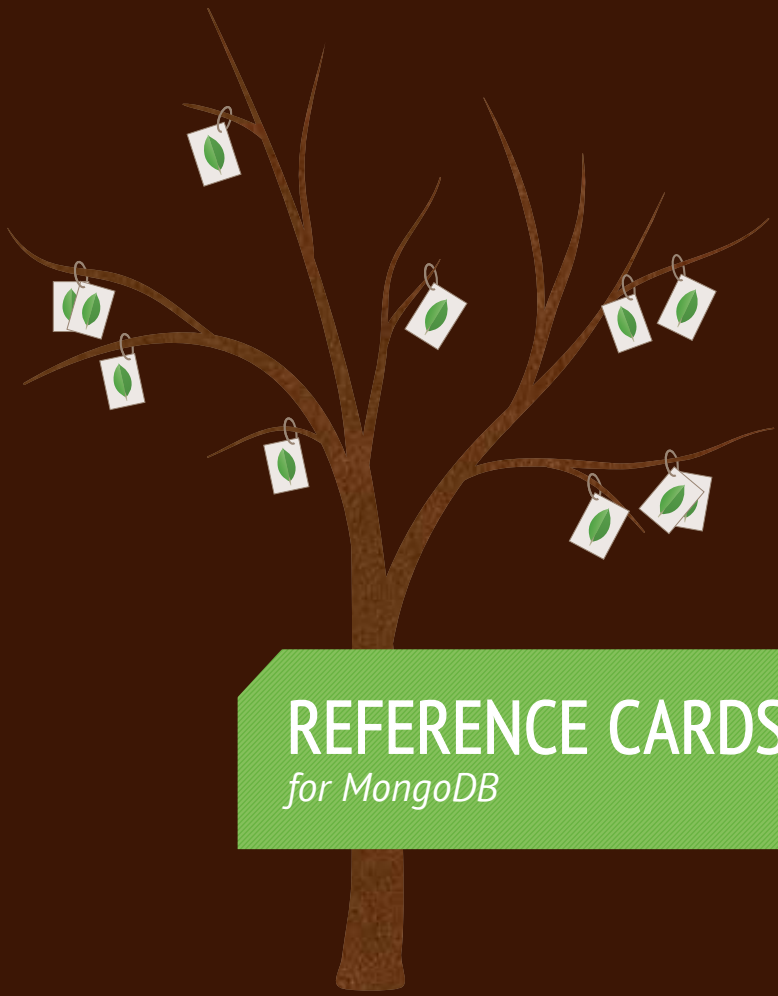

REFERENCE CARDS

for MongoDB

What is MongoDB?

MongoDB is an open-source, general purpose database.

Instead of storing data in rows and columns as one would with a relational database, MongoDB uses a document data model, and stores a binary form of JSON documents called BSON. Documents contain one or more fields, and each field contains a value of a specific data type, including arrays and binary data. Documents are stored in collections, and collections are stored in databases. It may be helpful to think of documents as roughly equivalent to rows in a relational database; fields as equivalent to columns; and collections as tables. There are no fixed schemas in MongoDB, so documents can vary in structure and can be adapted dynamically.

MongoDB provides full index support, including secondary, compound and geospatial indexes. MongoDB also features a rich query language; atomic update modifiers; text search; the Aggregation Framework for analytics similar to SQL GROUP BY operations; and MapReduce for complex, in-place data analysis.

Built-in replication with automated failover provides high availability. Auto-sharding enables horizontal scaling for large deployments. MongoDB also provides native, idiomatic drivers for all popular programming languages and frameworks to make development natural.

Queries and What They Match

`{a: 10}`	Docs where **a** is **10**, or an array containing the value **10**.
`{a: 10, b: "hello"}`	Docs where **a** is **10** and **b** is **"hello".**
`{a: {$gt: 10}}`	Docs where **a** is greater than **10**. Also available: **$lt (<), $gte (>=)**, **$lte (<=)**, and **$ne (!=)**.
`{a: {$in: [10, "hello"]}}`	Docs where **a** is either **10** or **"hello".**
`{a: {$all: [10, "hello"]}}`	Docs where **a** is an array containing both **10** and **"hello".**
`{"a.b": 10}`	Docs where **a** is an embedded document with **b** equal to **10**.
`{a: {$elemMatch: {b: 1, c: 2}}}`	Docs where **a** is an array that contains an element with both **b** equal to **1** and **c** equal to **2**.
`{$or: [{a: 1}, {b: 2}]}`	Docs where **a** is **1** or **b** is **2**.
`{a: /^m/}`	Docs where **a** begins with the letter **m**.
`{a: {$mod: [10, 1]}}`	Docs where **a** mod **10** is **1**.
`{a: {$type: 2}}`	Docs where **a** is a string (see bsonspec.org for more).

The following queries cannot use indexes as of MongoDB v2.0. These query forms should normally be accompanied by at least one other query term which does use an index:

`{a: {$nin: [10, "hello"]}}`	Docs where **a** is anything but **10** or **"hello"**.
`{a: {$size: 3}}`	Docs where **a** is an array with exactly 3 elements.
`{a: {$exists: true}}`	Docs containing an **a** field.
`{a: /foo.*bar/}`	Docs where **a** matches the regular expression **foo.*bar**.
`{a: {$not: {$type: 2}}}`	Docs where **a** is not a string. **$not** negates any of the other query operators.

Update Modifiers

`{$inc: {a: 2}}`	Increment **a** by **2.**
`{$set: {a: 5}}`	Set **a** to the value **5**.
`{$unset: {a: 1}}`	Delete the **a** key.
`{$push: {a: 1}}`	Append the value **1** to the array **a**.
`{$push: {a: {$each: [1, 2]}}}`	Append both **1** and **2** to the array **a**.
`{$addToSet: {a: 1}}`	Append the value **1** to the array **a** (if the value doesn't already exist).
`{$addToSet: {a: {$each: [1, 2]}}}`	Append both **1** and **2** to the array **a** (if they don't already exist).
`{$pop: {a: 1}}`	Remove the last element from the array **a**.
`{$pop: {a: -1}}`	Remove the first element from the array **a**.
`{$pull: {a: 5}}`	Remove all occurrences of **5** from the array **a**.
`{$pullAll: {a: [5, 6]}}`	Remove all occurrences of **5** or **6** from the array **a**.

Mapping SQL to MongoDB

Converting to MongoDB Terms

MYSQL EXECUTABLE	ORACLE EXECUTABLE	MONGODB EXECUTABLE
mysqld	oracle	mongod
mysql	sqlplus	mongo

SQL TERM	MONGODB TERM
database	database
table	collection
index	index
row	document
column	field
joining	embedding & linking

Queries and other operations in MongoDB are represented as documents passed to **`find`** and other methods. Below are examples of SQL statements and the analogous statements in MongoDB JavaScript shell syntax.

SQL	MONGODB
CREATE TABLE users (name VARCHAR(128), age NUMBER)	db.createCollection("users")
INSERT INTO users VALUES ('Bob', 32)	db.users.insert({name: "Bob", age: 32})
SELECT * FROM users	db.users.find()
SELECT name, age FROM users	db.users.find({}, {name: 1, age: 1, _id:0})
SELECT name, age FROM users WHERE age = 33	db.users.find({age: 33}, {name: 1, age: 1, _id:0})
SELECT * FROM users WHERE age > 33	db.users.find({age: {$gt: 33}})
SELECT * FROM users WHERE age <= 33	db.users.find({age: {$lte: 33}})

SQL	MONGODB
SELECT * FROM users WHERE age > 33 AND age < 40	db.users.find({age: {$gt: 33, $lt: 40}})
SELECT * FROM users WHERE age = 32 AND name = 'Bob'	db.users.find({age: 32, name: "Bob"})
SELECT * FROM users WHERE age = 33 OR name = 'Bob'	db.users.find({$or:[{age:33}, {name: "Bob"}]})
SELECT * FROM users WHERE age = 33 ORDER BY name ASC	db.users.find({age: 33}).sort({name: 1})
SELECT * FROM users ORDER BY name DESC	db.users.find().sort({name: -1})
SELECT * FROM users WHERE name LIKE '%Joe%'	db.users.find({name: /Joe/})
SELECT * FROM users WHERE name LIKE 'Joe%'	db.users.find({name: /^Joe/})
SELECT * FROM users LIMIT 10 SKIP 20	db.users.find().skip(20).limit(10)
SELECT * FROM users LIMIT 1	db.users.findOne()
SELECT DISTINCT name FROM users	db.users.distinct("name")
SELECT COUNT(*) FROM users	db.users.count()
SELECT COUNT(*) FROM users WHERE AGE > 30	db.users.find({age: {$gt: 30}}).count()
SELECT COUNT(AGE) FROM users	db.users.find({age: {$exists: true}}). count()
UPDATE users SET age = 33 WHERE name = 'Bob'	db.users.update({name: "Bob"}, {$set: {age: 33}}, {multi: true})
UPDATE users SET age = age + 2 WHERE name = 'Bob'	db.users.update({name: "Bob"}, {$inc: {age: 2}}, {multi: true})
DELETE FROM users WHERE name = 'Bob'	db.users.remove({name: "Bob"})
CREATE INDEX ON users (name ASC)	db.users.ensureIndex({name: 1})
CREATE INDEX ON users (name ASC, age DESC)	db.users.ensureIndex({name: 1, age: -1})
EXPLAIN SELECT * FROM users WHERE age = 32	db.users.find({age: 32}).explain()

Replication

About Replication

Replication is the process of duplicating data from one system to another. MongoDB provides replication through a configuration called a **replica set**. Each replica set has a **primary** and one or more **secondaries**.

Replica sets provide automatic failover and recovery. If the primary becomes unavailable, the set will attempt to elect one of the secondaries as the new primary. A **majority** of the servers must be able to reach the primary in order to both elect it and to keep it a primary.

What is a Majority?

If your set consists of...

1 server, 1 server is a majority.

2 servers, 2 servers are a majority.

3 servers, 2 servers are a majority.

4 servers, 3 servers are a majority.

...

Setup

To initialize a three-node replica set with one arbiter, start three **`mongod`** instances, each using the **`--replSet`** flag followed by a name for the replica set. For example:

```
mongod --replSet cluster-foo
```

Next, connect to one of the **`mongod`** instances and run the following:

```
rs.initiate()
rs.add("host2:27017")
rs.add("host3:27017", true)
```

`rs.add()` can also accept a document parameter, such as **`rs.add({"_id": "4", "host": "host4:27017"})`**. The document can contain the following options:

`priority: n`	Members will be elected primary in order of priority, if possible. **`n=0`** means this member will never be a primary.
`slaveDelay: n`	This member will always be a secondary and will lag **`n`** seconds behind the master.

`arbiterOnly: true`	This member will be an arbiter.
`hidden: true`	Do not show this member in **`isMaster`** output. Use this option to hide this member from clients.
`tags: [...]`	Member location description. *See docs.mongodb.org/manual/data-center-awareness.*

Shell Helpers

`rs.initiate()`	Create a new replica set with one member.
`rs.add("host:port")`	Add a member.
`rs.remove("host:port")`	Remove a member.

Administration

`rs.status()`	See the status of each member.
`rs.conf()`	See the current set configuration.
`rs.reconfig(newConfig)`	Change the set configuration.
`rs.isMaster()`	See which member is primary.
`rs.stepDown(n)`	Force the primary to become a secondary for **n** seconds, during which time an election can take place.
`rs.freeze(n)`	Prevent a secondary from becoming the primary for **n** seconds (**`n=0`** means unfreeze).

Tips

- Run at least one member with journaling enabled to ensure that you always have a clean copy of your data available.
- Start replica set members with the **`--rest option`** to enable the web interface.

Indexing

Index Creation

db.coll.ensureIndex(key_pattern, options)

Create an index on collection **coll** with the given key pattern and options.

Indexing Key Patterns with Sample Queries

{username: 1} Ex: db.users.find({username: 'smith'});	Simple index on **username**.
{last_name: 1, last_login: -1} Ex: db.users.find({last_name: 'jones'}).sort({last_login: -1})	Compound index with **last_name** ascending and **last_login** descending. Note that key order on compound indexes matters.
{coord: '2d'} Ex: db.places.find({coord: {$near: [50, 50]}})	Geospatial index, where **coord** is a coordinate **(x,y)** where -180 < **x**, **y** < 180. Note that **$near** queries return the closest points to the given coordinate.
{loc: '2dsphere'} db.places.find({coord: {$near: {$geometry: {type: "Point", coordinates: [125, 90]}}}})	Geospatial index where the **loc** field stores GeoJSON data. Note that you should always store coordinates in the following order: longitude, latitude. Valid longitude values are between -180 and 180. Valid latitude values are between -90 and 90.

Index Creation Options

{unique: true}	Create a unique index. To check insertion failures, you must use your driver's safe mode.
{dropDups: true}	Use with the **unique** option. Drop documents with duplicate values for the given key pattern on index creation.
{background: true}	Create this index in the background; useful when you need to minimize index creation performance impact.
{sparse: true}	Create entries in the index only for documents having the index key.
{name: 'foo'}	Specify a custom name for this index. If not specified, the name will be derived from the key pattern.

Examples

`db.users.ensureIndex({username: 1}, {unique: true})`	Create a unique index on **`username`**.
`db.products.ensureIndex({category: 1, price: -1}, {background: true})`	Create a compound index on **`category`** and **`price`** and build it in the background.
`db.places.ensureIndex({loc: '2dsphere'})`	Create a **`2dsphere`** geospatial index on **`loc`**.

Administration

`db.users.getIndexes()`	Get a list of all indexes on the **`users`** collection.
`db.users.totalIndexSize()`	Get the number of bytes allocated by indexes for the **`users`** collection.
`db.users.reIndex()`	Rebuild all indexes on this collection.
`db.users.dropIndex({x: 1, y: -1})`	Drop the index with key pattern **`{x: 1, y: -1}`**. Use **`db.users.dropIndexes()`** to drop all indexes on the **`users`** collection.

Tips

You can use a compound index on **`{username: 1, date: 1}`** for the following queries:

```
db.users.find({username: "Jones"});
db.users.find({username: /^Jones/});
db.users.find({username: "Jones", date: new Date()});
db.users.find({username: "Jones"}).sort({date: -1});
db.users.find({}).sort({username: 1, date: 1}).limit(100);
```

Note that with this index, a separate single-key index on **`{username: 1}`** is unnecessary.

Resources

Learn

In-browser Tutorial - try.mongodb.org
Downloads - mongodb.org/downloads
MongoDB Manual - docs.mongodb.org/manual
Online Education - education.mongodb.com
Presentations - mongodb.com/presentations
In-person Training - mongodb.com/training

Support

Stack Overflow - stackoverflow.com/questions/tagged/mongodb
Google Group - groups.google.com/group/mongodb-user
IRC - irc://irc.freenode.net/#mongodb
Bug Tracking - jira.mongodb.org
MongoDB Management Service - mms.mongodb.com
Commercial Support - mongodb.com/support

Community

MongoDB User Groups (MUGs) - mongodb.com/user-groups
MongoDB Events - mongodb.com/events

Social

Twitter - @MongoDB, @MongoDB_Inc
Facebook - facebook.com/mongodb
LinkedIn - linkedin.com/groups/MongoDB-2340731

Contact

Contact MongoDB - mongodb.com/contact

Published August 2013.
For more information or to download MongoDB, visit mongodb.org.