COSC 416
NoSQL Databases

MongoDB

Tim, Bryan, Derrick
University of British Columbia Okanagan

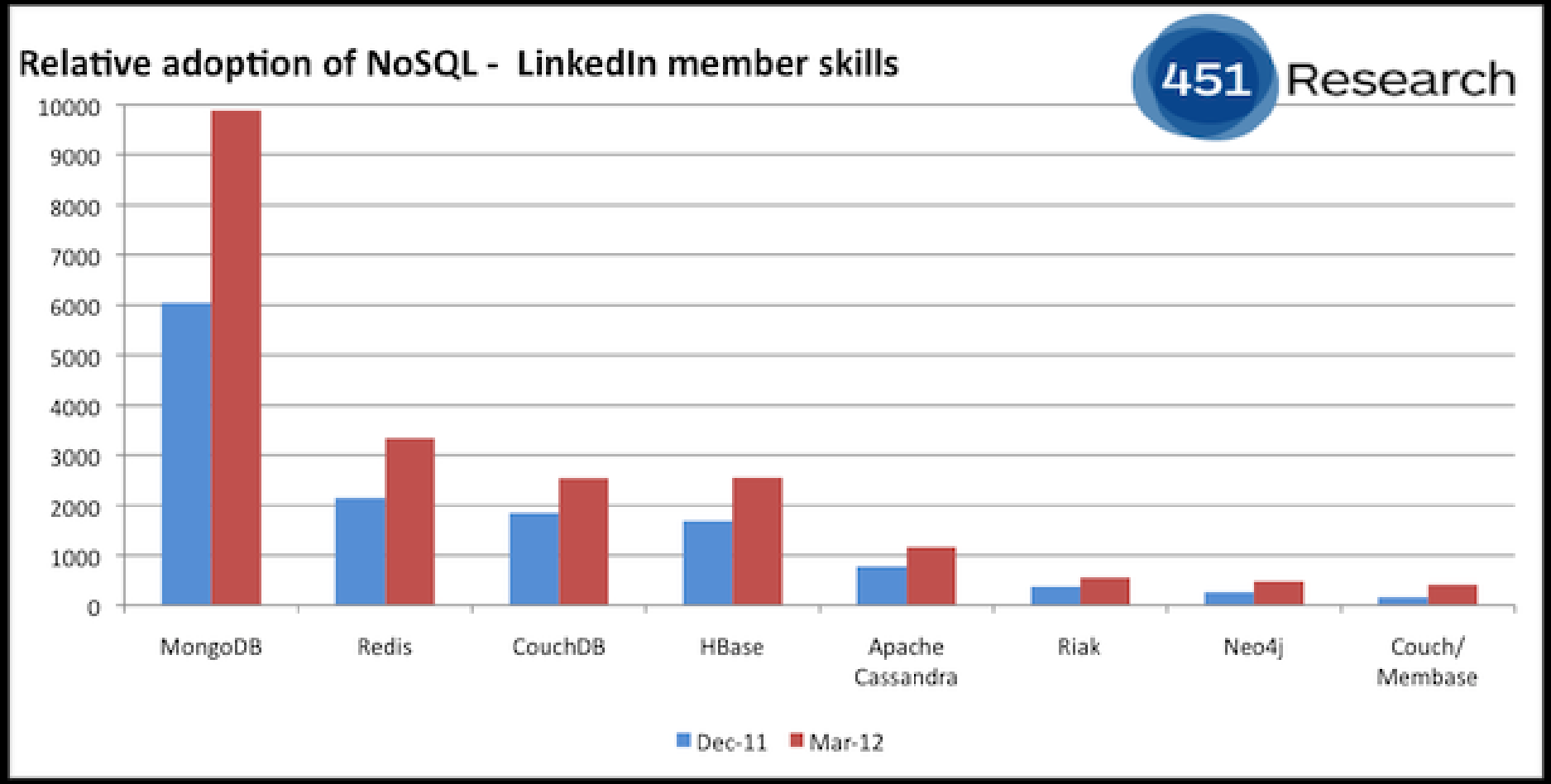
MongoDB

MongoDB is a open source document database that provides high performance, high availability, and easy scalability.

Document Database

- MongoDB stores structured data as JSON-like documents.
- Dynamic schemas (MongoDB calls the format BSON).
- Easy integration.
- Embedded documents and arrays reduce need for joins.
- MongoDB is the most popular NoSQL database management system.

Most Popular NoSQL Databases According to LinkedIn Skillsets

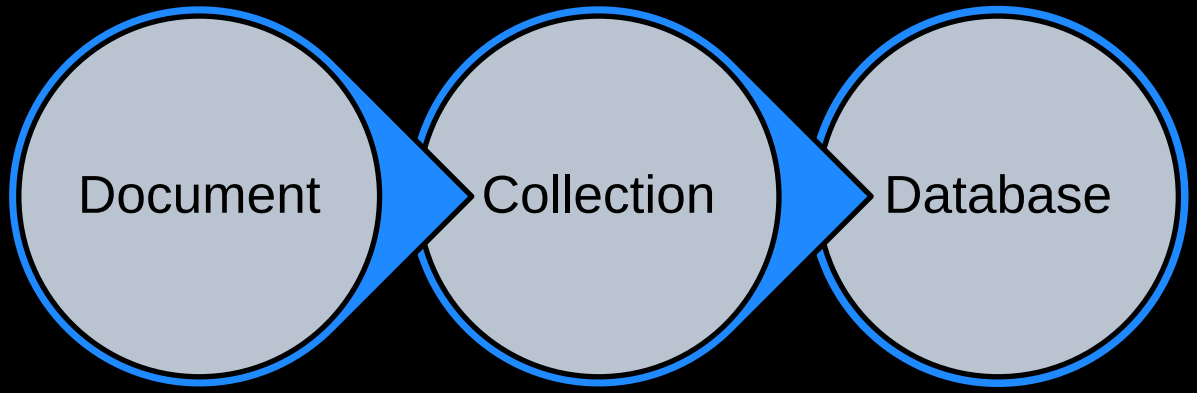


MongoDB Data Model

A MongoDB deployment hosts a number of databases.

A *database* holds a set of collections. A *collection* holds a set of documents. A *document* is a set of key-value pairs.

Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.



Main Features

Flexibility

Dynamic schema makes it easy to evolve your data.

Power

Secondary indexes, dynamic queries, sorting, updates, upserts, and easy aggregation.

Speed/Scaling

Faster Queries. No need for joins. Sharding.

Ease Of Use

Easy to install, configure, maintain, and use.

Core MongoDB Operations (CRUD)

Create

-insert(): insert a document into a collection

Read

-find(): primary method to select documents from a collection

-findOne(): returns only a single object

Update

-update(): corresponds to the UPDATE operation in SQL

-save(): performs an update that replaces the existing document

Delete

-remove(): delete documents from a collection

MongoDB shell is JavaScript-Based.

You can do things like:

```
a = 5;
a * 10;
for(i=0; i<10; i++) { print('hello'); };
```

We store data in documents (JavaScript objects).

A simple document:

```
var person = {name: 'Ed', languages: ['c', 'ruby', 'js']};
```

Saving to MongoDB

Then you save your 'documents' to MongoDB (database)

db.scores.save(person)

In this case, the document 'person' is saved to a collection called 'scores'

You can add documents to a collection using a loop

```
for(i=0; i<10; i++) { db.scores.save({a: i, exam: 5}) };
```

Basic Queries

Find all documents where a == 2;

```
db.scores.find({a: 2});
```

Query Operators

```
$lt  - '<'
$lte - '<='
$gte - '>='
$gt - '>'
$ne  - '!='
$in - 'is in array',
$nin - '! in array'
```

Query Operator Examples

```
db.inventory.find({qty:{$gt:20}})
```

- Query selects all documents in the inventory collection where the qty field value is greater than 20.

```
db.inventory.find({qty:{$in:[5, 15]}})
```

- Query selects all documents in the inventory collection where the qty field is either 5 or 15.

Updating

```
db.users.save({name: 'Johnny', languages: ['ruby', 'c']});
db.users.save({name: 'Sue', languages: ['scala', 'lisp']});


db.users.update({name: 'Johnny'}, {name: 'Cash', languages:
['english']});
```

- This query will update the name of 'Johnny' into 'Cash' and change the languages to 'english'.

Updating

MongoDB also supports partial updates to documents. For example, you can set a value:

```
db.users.update({name: 'Cash'}, {'$set': {'age': 50} });
```

You can also push and pull items from arrays:

```
db.users.update({name: 'Sue'}, {'$pull': {'languages': 'scala'} }); // pull removes that value.

db.users.update({name: 'Sue'}, {'$push': {'languages': 'ruby'} });  // push adds that value to the document.
```

Deleting

To delete only matching documents, add a query selector to the remove method:

```
db.users.remove({name: 'Sue'});
```

To delete everything from a collection:

```
db.scores.remove();
```

Query Operators

$all – selects the documents where the field holds an array and contains all elements.

```
db.person.find({tags: {$all: [“applications”,”school”,”book”] }})
```

This query selects all documents in the person collection where the tag field contains an array with all the elements 'applications', 'school', 'book'.

Indexes

Every document must have a unique _id field. MongoDB creates this index by default on all collections.

_id field is a primary key for your collection. The default value of _id is ObjectID on every insert().

You can’t delete or change the index _id.

MongoDB Tutorial

MONGO DOCS
MongoDB Documentation »

TRY IT OUT
Try The Online Shell »

DOWNLOADS
Download MongoDB »

DRIVERS
Get The Latest Drivers »

A Tiny MongoDB **Browser Shell**

Just enough to scratch the surface (mini tutorial included)

X CLOSE

```
MongoDB browser shell version: 0.1.0
connecting to random database
type "help" for help
type "tutorial" to start the tutorial
> tutorial
This is a self-guided tutorial on MongoDB and the MongoDB shell.
The tutorial is simple, more or less a few basic commands to try.
To go directly to any part tutorial, enter one of the commands t0, t1, t2...t10
Otherwise, use 'next' and 'back'. Start by typing 'next' and pressing enter.
> next
1. JavaScript Shell
The first thing to notice is that the MongoDB shell is JavaScript-based.
So you can do things like:
  a = 5;
  a * 10;
  for(i=0; i<10; i++) { print('hello'); };
Try a few JS commands; when you're ready to move on, enter 'next'
>
```

MongoDB Design Conclusions

- Data model document based (Non-relational approach).
- Large data sets can be partitioned across many machines. (Large scale robust systems).
- Writing code: easier, faster, and more agile.
- Provides powerful features similar in a traditional relational DBMS such as secondary indexes, unique key constraints, atomic operations, multi-document updates.

Resources

Operators

http://docs.mongodb.org/manual/reference/operators/

MongoDB Driver and Client Libraries

http://docs.mongodb.org/ecosystem/drivers/

Tutorial

http://www.mongodb.org/

MongoDB Manual

http://docs.mongodb.org/manual/

JavaScript Syntax

http://en.wikipedia.org/wiki/JavaScript_syntax