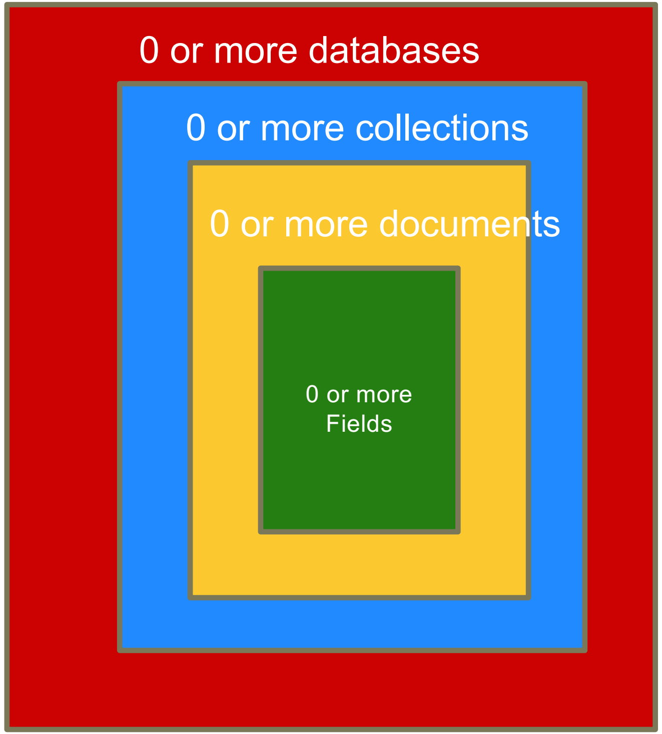
MongoDB

HTTPS://DOCS.MONGODB.ORG/MANUAL/

MongoDB is a document store



- A MongoDB instance may have zero or more ‘databases’
- A database may have zero or more ‘collections.
- A collection may have zero or more ‘documents.
- A document may have one or more ‘fields or attributes for an object.

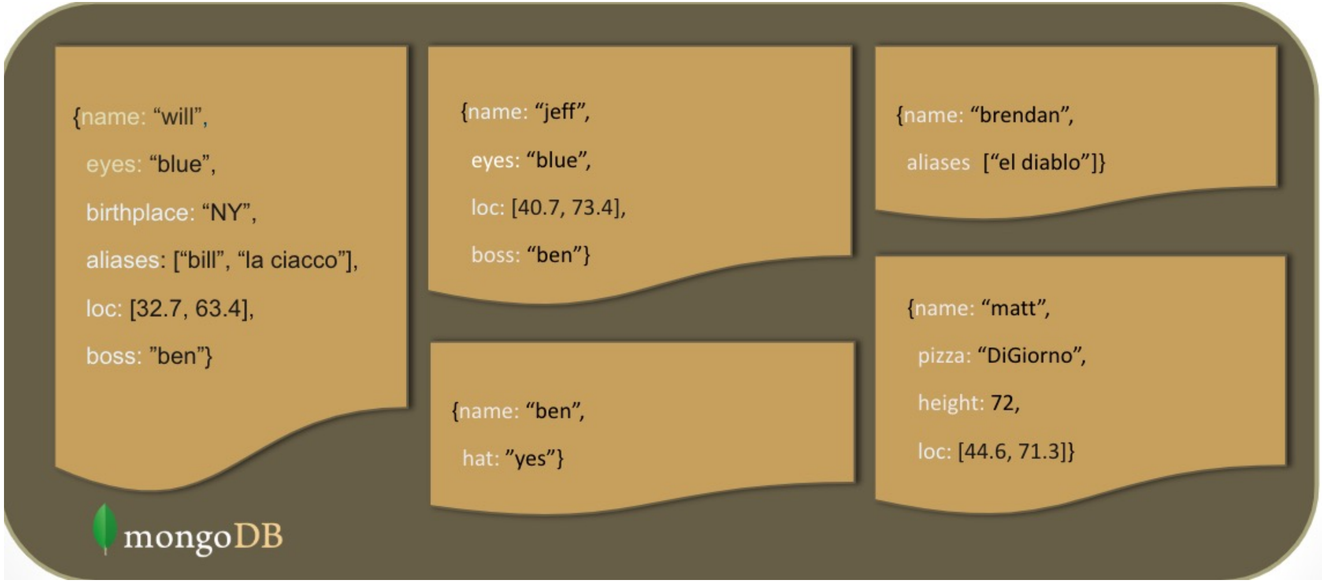
MongoDB has a flexible schema

MongoDB does not need a pre-defined data schema
Every document in a collection could have different data

No Data Definition Language:

Each object describes it own structure

Flexible schema



```
{name: "will",
  eyes: "blue",
  birthplace: "NY",
  aliases: ["bill", "la ciacco"],
  loc: [32.7, 63.4],
  boss: "ben"}
```

```
{name: "jeff",
  eyes: "blue",
  loc: [40.7, 73.4],
  boss: "ben"}
```

```
{name: "ben",
  hat: "yes"}
```

```
{name: "brendan",
  aliases  ["el diablo"]}
```

```
{name: "matt",
  pizza: "DiGiorno",
  height: 72,
  loc: [44.6, 71.3]}
```

RDB Concepts to NoSQL

RDBMS	Mongo DB	Notes
Database	Database	Both system have the USE command.
Table, View	Collection	Collection is not rigid in its structure
Row	Document	Document is not rigid in its structure – it can embed other object
Column	Field	Field can have different domains, can consist of a data value, an object , a reference to an object or a collection of objects
Index	Index	Same data structures
Join	Embedded document	JOIN added to MongoDB operations
Foreign key	Reference	Link 2 objects together
Partition	Shard	Portion of a collection stored on a specific server

MongoDB features

- Dynamic schema
  - No DDL
- Document-based database
- Supports secondary indexes
- Query language via an API
- Atomic writes and fully-consistent reads
  - If system configured that way
- Master-slave replication with automated failover (replica sets)
- Built-in horizontal scaling via automated range-based partitioning of data (sharding)
- Originally no joins, transactions, triggers or events
  - Outer joins added in MongoDB 3.2
  - Transactions added in MongoDB 4.0
  - MongoDB Atlas provides triggers and event functionality in MongoDB 3.6

Compass a GUI for MongoDB

We are going to use MongoDB’s Compass application to connect to a web-deployed Atlas database

Connection:

mongodb+srv://student:student@cluster0.jabrq.mongodb.net/test

Compass sets a context for querying the database

Formulating a query in Compass

MongoDB uses json notation to represent a query. It is surrounded by braces and contains a field that has a key and a value

{ name : “Joe” }

Find all documents with a field named *name* and has the value, Joe.

This query is performed across a collection of documents.

Json field types

Json fields support different value types:

Strings

Numbers

Boolean

Null

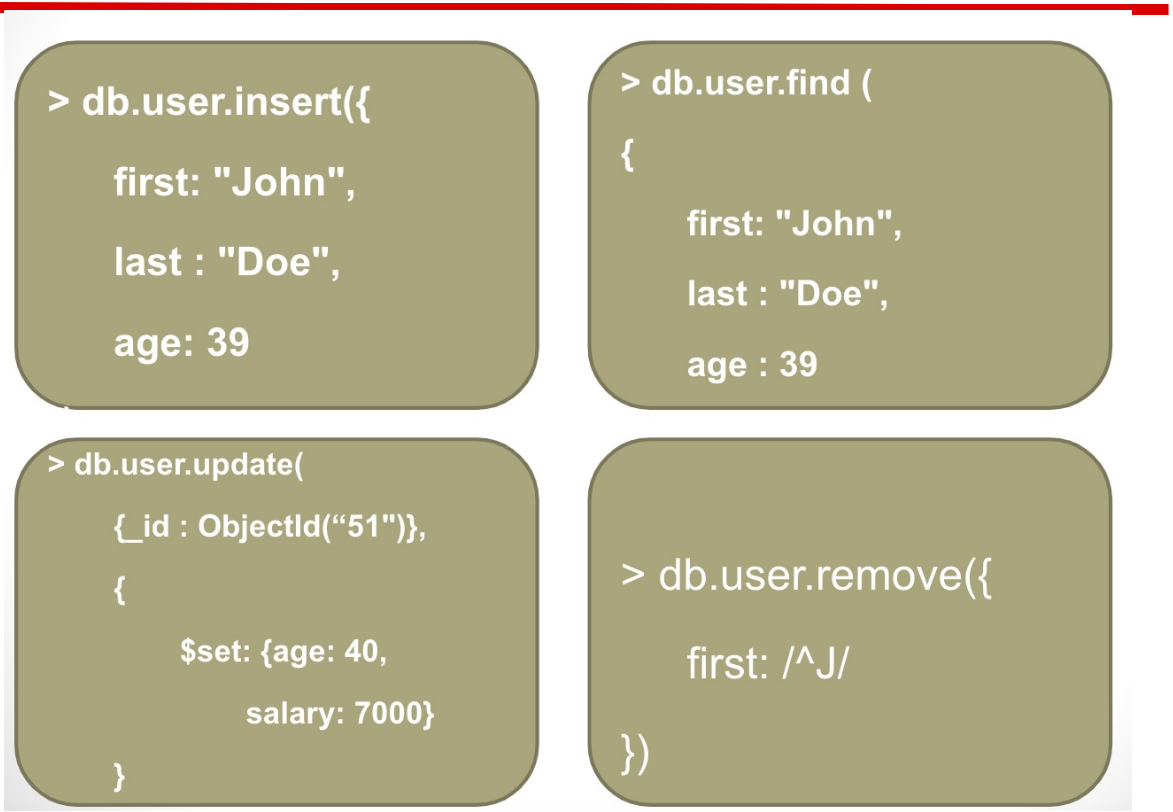
Arrays – a field can be a collection of values

Objects – embedded objects with its own structure

CRUD examples in MongoDB client

User is the name of the collection

_id is the primary key of the collection managed by Mongo



```
> db.user.insert({
    first: "John",
    last : "Doe",
    age: 39
```

```
> db.user.find (
{
    first: "John",
    last : "Doe",
    age : 39
```

```
> db.user.update(
    {_id : ObjectId("51")},
    {
        $set: {age: 40,
            salary: 7000}
    }
```

```
> db.user.remove({
    first: /^J/
})
```

Comparison: insert user and emails into DB

SQL STATEMENTs

```
START TRANSACTION;
INSERT INTO contacts VALUES
   (NULL, ‘joeblow’);
INSERT INTO contact_emails
  VALUES
  ( NULL, ”joe@blow.com”,
    LAST_INSERT_ID() ),
  ( NULL, “joseph@blow.com”,
    LAST_INSERT_ID() );
COMMIT;
```

MongoDB

```
contacts.save( {
   userName: “joeblow”,
   emailAddresses: [
     “joe@blow.com”,
     “joseph@blow.com” ] } );
```

Queries in Compass

You provide a filter that documents need to satisfy

{} – no filter

General structure:

{ <field1>: { <operator1>: <value1> }, ... }

Example:
{ status: { $in: ["A", "D"] } }

Compound queries with AND

A compound query can specify conditions for more than one field

When more than one restriction is provided for the query, they form a conjunction with the logical AND

Example:

{ status: "A", qty: { $lt: 30 } }

Find all instances where status = A and qty is less than 30

Compound queries with OR

Use the OR operator to specify either criteria can be TRUE to satisfy the filter

Example:

```
{ $or: [ { status: "A" }, { qty: { $lt: 30 } } ] }
```

Find all instances where status = A or qty is less than 30

note the array bracket notation

Combining AND and OR in a query

Like in SQL, we can create a complex predicate using both AND and OR

EXAMPLE:

```
{ status: "A", $or: [ { qty: { $lt: 30 } }, { item: /^p/ } ] }
```

Find all documents with status = A AND where the quantity field is less than 30 OR the item begins with the letter p

Match on an embedded object

If you want to match values in an embedded object, we still use the format {field: value}. Value is an object with the values you plan to match.

Example:

{ size: { h: 14, w: 21, uom: "cm" } }

{{field: value}}

Filter does an exact match on the specified embedded object, match the names of the fields, the values of the fields, even the order of the fields need to be correct.

Querying a field in an embedded object

Use dot notation to specify a field in an embedded object.
Outside_field.inside_field

The parser requires quotes around the field specification
“outside_field.inside_field”

Example:
{ "size.uom": "in" }

Matching a field that is an array

You can specify an exact match for a field that is an array by providing an array for the value.

```
{ tags: ["red", "blank"] }
{{field: value}}
```

Finds an object where the tag field is an array of value [“red”, “blank”]

Matching a field that contains a subset

You can use the $all function to match both values, without the match paying attention to order of the array or other values in the array.

```
{ tags: { $all: ["red", "blank"] } }
```

Finds all objects that are tagged as “red” and “blank”

Query an array for an element

It is simple to find all objects with an array element equal to a specific value

```
{ tags: "red" }
```

Finds all objects that are tagged as “red”

Filter an array with a criteria

It is simple to find all objects with an array element satisfying any of the predefined logical operations in MongoDB

```
{ dim_cm: { $gt: 25 } }
```

Finds all where the dim_cm > 25

Providing multiple criteria for an array

When specifying compound conditions on array elements, you can specify the query such that either a single array element meets these conditions, or any combination of array elements meets the conditions.

{ dim_cm: { $gt: 15, $lt: 20 } }

one element can satisfy the greater than 15 condition and another element can satisfy the less than 20 condition, or a single element can satisfy both

Provide multiple criteria for an array element

Use $elemMatch operator to specify multiple criteria on the elements of an array such that at least one array element satisfies all the specified criteria.

```
{ dim_cm: { $elemMatch: { $gt: 22, $lt: 30 } } }
```

one element is greater than the 22 condition and less than 30

Query for an array element by position

You can specify a filter for a specific array element

```
{ "dim_cm.1": { $gt: 25 } }
```

Retrieves all documents where the second element in the array dim_cm is greater than 25.

Array indexing starts at 0.

Query by array size

You can specify a filter that retrieves documents where the array size meets a criterion, using the $size operator

```
{ "tags": { $size: 3 } }
```

Retrieves all documents where the tags array has 3 elements

Query operators

Name	Description
$eq	Matches value that are equal to a specified value
$gt, $gte	Matches values that are greater than (or equal to a specified value
$lt, $lte	Matches values less than or (equal to) a specified value
$ne	Matches values that are not equal to a specified value
$in	Matches any of the values specified in an array
$nin	Matches none of the values specified in an array
$or	Joins query clauses with a logical OR returns all
$and	Join query clauses with a loginal AND
$not	Inverts the effect of a query expression
$nor	Join query clauses with a logical NOR
$exists	Matches documents that have a specified field

Summary

- MongoDB
  - Document oriented data store, flexible schema, provides a query language via an API, consistent reads on primary replica set
  - Has added transactions, joins and references to objects to support relational operations

References

https://docs.mongodb.com/manual/tutorial/query-documents/

https://docs.mongodb.com/manual/tutorial/query-embedded-documents/

https://docs.mongodb.com/compass/current/