Introduction to MongoDB

Reference: https://docs.mongodb.com/manual/

What is MongoDB

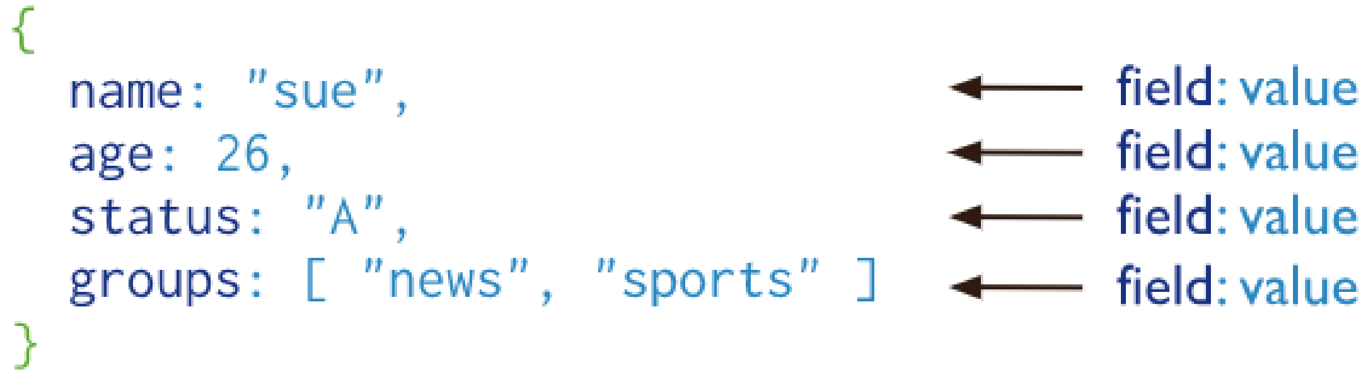
- MongoDB is an open-source **document-oriented** database that provides high performance, high availability, and automatic scaling

- MongoDB falls in the category of the NoSQL – Database which means it doesn’t follow fixed schema structure like in relational databases

- NoSQL
    - Originally referring to “non SQL” or “non relational”
    - Sometimes called “Not only SQL” to emphasize that they may support SQL-like query languages

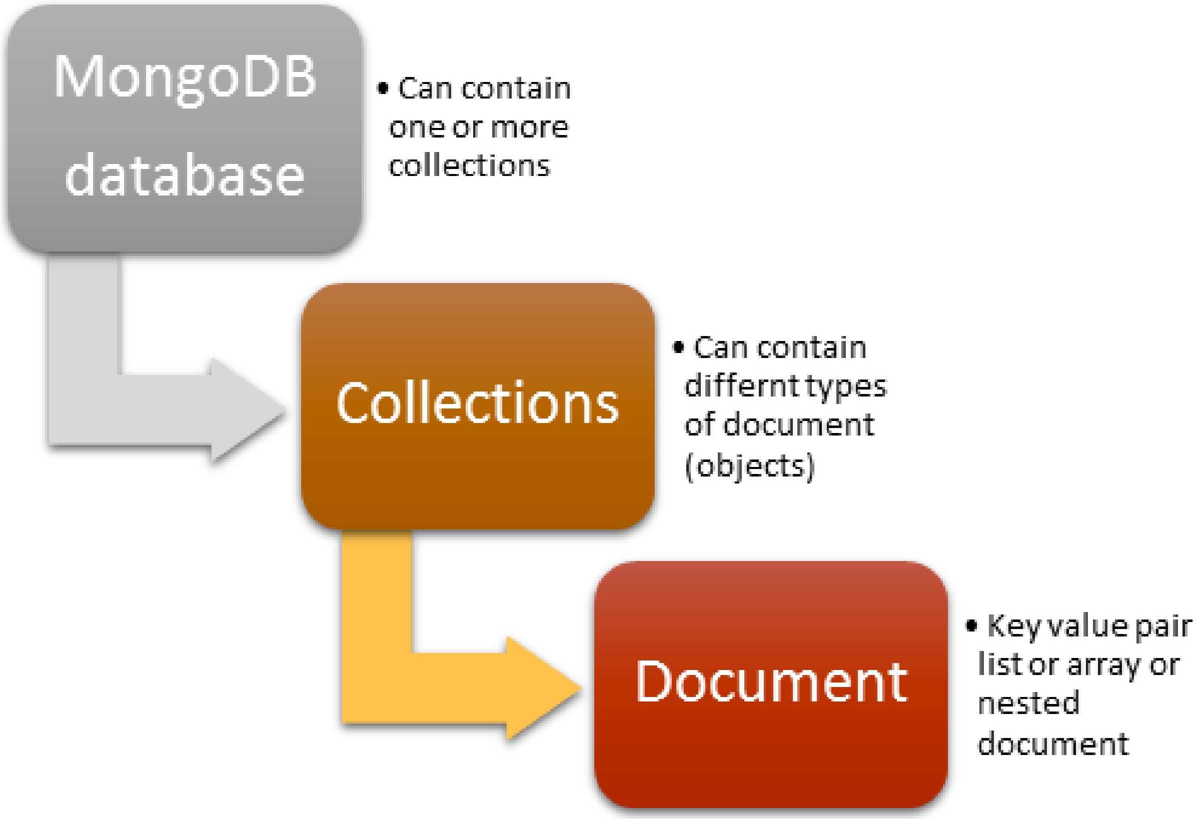
Document Database

- The MongoDB database consists of a set of databases
    - Each database contains multiple **collections**. Every collection can contain different types of object
    - Every object is also called **document**, which is a data structure composed of field and value pairs. The values of fields may include other documents, arrays, and arrays of documents.
    - A record in MongoDB is a document



```
{
   name: "sue",
   age: 26,
   status: "A",
   groups: [ "news", "sports" ]
}
```

RDBMS and MongoDB


MongoDB database
• Can contain one or more collections
Collections
• Can contain differnt types of document (objects)
Document
• Key value pair list or array or nested document

Relational Database	MongoDB
Database	Database
Table	Collection
Row	Document
Column	Field

https://www.codeproject.com/Articles/1037052/Introduction-to-MongoDB

Advantages of using documents

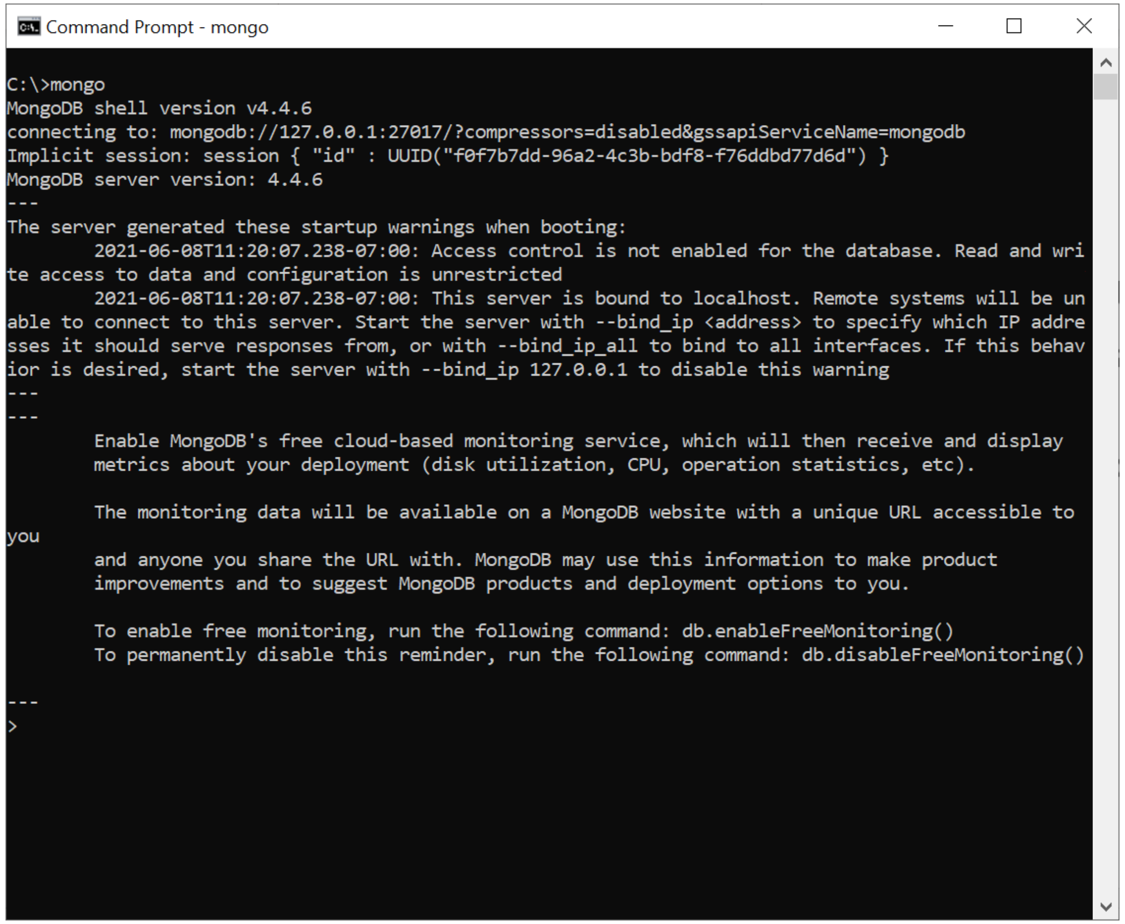
- Documents (i.e., objects) correspond to native data types in many programming languages
- Embedded documents and arrays reduce need for expensive joins
- Dynamic schema supports fluent polymorphism

SQL vs MongoDB

	SQL	MongoDB
Focus	Data Storage	Data usage
Design	• Everything predesigned • Rigid schema • One value / column	• Easy to evolve as needed • Each record can have different fields* • Can contain an array of values
Terminology	• DB • Table • Row (Record) • Column (Attribute)	• DB • Collection • Document • Field

mongo Shell

- The mongo shell is an interactive JavaScript interface to MongoDB
- You can use the mongo shell to query and update data as well as perform administrative operations
- You can check the MongoDB installation lab on how to install MongoDB and start the mongo shell



```
C:\>mongo
MongoDB shell version v4.4.6
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("f0f7b7dd-96a2-4c3b-bdf8-f76ddbd77d6d") }
MongoDB server version: 4.4.6
---
The server generated these startup warnings when booting:
        2021-06-08T11:20:07.238-07:00: Access control is not enabled for the database. Read and wri
te access to data and configuration is unrestricted
        2021-06-08T11:20:07.238-07:00: This server is bound to localhost. Remote systems will be un
able to connect to this server. Start the server with --bind_ip <address> to specify which IP addre
sses it should serve responses from, or with --bind_ip_all to bind to all interfaces. If this behav
ior is desired, start the server with --bind_ip 127.0.0.1 to disable this warning
---
---
        Enable MongoDB's free cloud-based monitoring service, which will then receive and display
        metrics about your deployment (disk utilization, CPU, operation statistics, etc).

        The monitoring data will be available on a MongoDB website with a unique URL accessible to
you
        and anyone you share the URL with. MongoDB may use this information to make product
        improvements and to suggest MongoDB products and deployment options to you.

        To enable free monitoring, run the following command: db.enableFreeMonitoring()
        To permanently disable this reminder, run the following command: db.disableFreeMonitoring()

---
>
```

Mongo CRUD Operations

- CRUD (Create Read Update Delete)
- Create a database
  - Type *use [database name]* and press *enter*. If the database exists the MongoDB will switch to database else it will create a brand-new database for you
  - type *use inventory* to create a database named *inventory*

Start

- To display the database you are using, type *db*
- To switch databases, type *use <database>*
- To list the databases available to the user, type *show dbs*
  - Empty databases are not shown
  - https://jira.mongodb.org/browse/SERVER-18313

Create a Database

- You can switch to non-existing databases. When you first store data in the database, such as by creating a collection, MongoDB creates the database. For example, the following creates both the database myNewDatabase and the collection myCollection during the insertOne() operation:

```
use myNewDatabase
db.myCollection.insertOne( { x: 1 } )
```

- Print a list of all collections for current database

```
show collections
```

Drop a collection/Database

- Removes a collection or view from the database

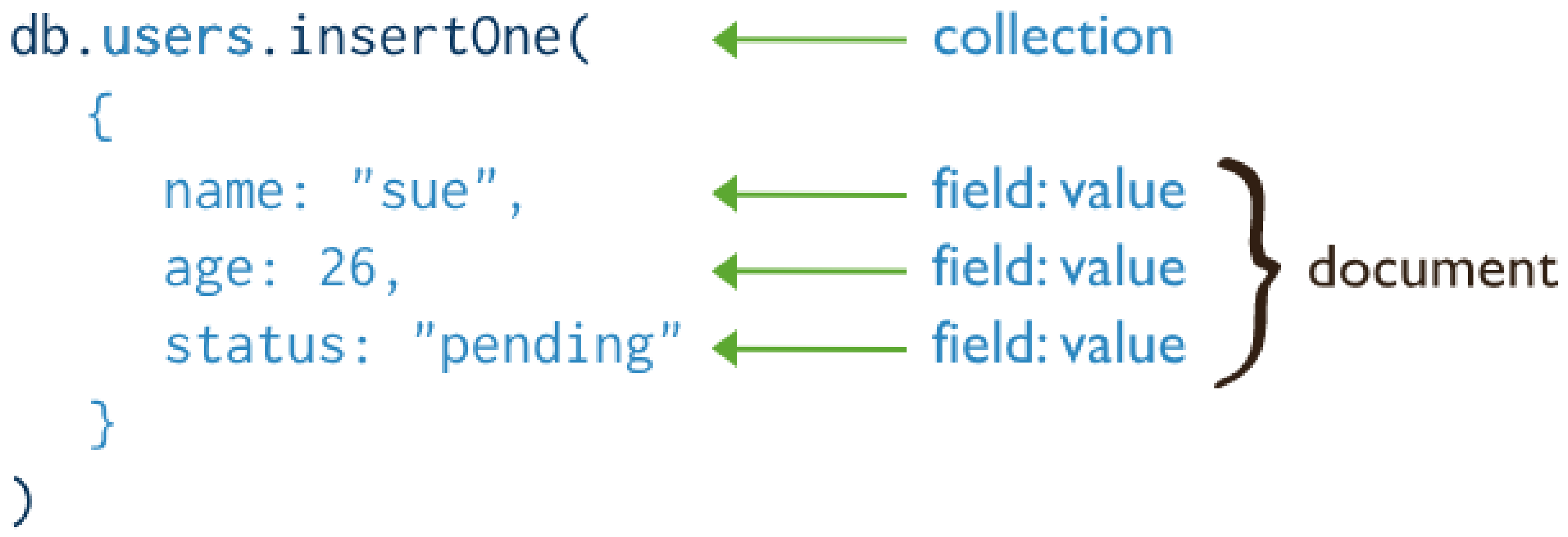
  db.collection.drop()

- Removes the current database, deleting the associated data files

  db.dropDatabase()

Insert

- In MongoDB, insert operations target a single collection. All write operations in MongoDB are atomic on the level of a single document



```
db.users.insertOne(
   {
      name: "sue",
      age: 26,
      status: "pending"
   }
)
```

- If the collection does not currently exist, insert operations will create the collection

Insert a Single Document

- db.collection.insertOne() inserts a single document into a collection

```
db.inventory.insertOne(
   { item: "canvas", qty: 100, tags: ["cotton"], size: { h: 28, w: 35.5, uom: "cm" } }
)
```

- If the document does not specify an _id field, MongoDB adds the _id field with an ObjectId value to the new document.
  - _id Field: In MongoDB, each document stored in a collection requires a unique _id field that acts as a primary key. If an inserted document omits the _id field, the MongoDB driver automatically generates an ObjectId for the _id field
- To retrieve the document that you just inserted, query the collection:

```
db.inventory.find( { item: "canvas" } )
```

Insert Multiple Documents

- db.collection.insertMany() can insert multiple documents into a collection. Pass an array of documents to the method.

```
db.inventory.insertMany([
   { item: "journal", qty: 25, tags: ["blank", "red"], size: { h: 14, w: 21, uom: "cm" } },
   { item: "mat", qty: 85, tags: ["gray"], size: { h: 27.9, w: 35.5, uom: "cm" } },
   { item: "mousepad", qty: 25, tags: ["gel", "blue"], size: { h: 19, w: 22.85, uom: "cm" } }
])
```

- To retrieve the inserted documents

```
db.inventory.find( {} )
```

Query Documents

- First let’s populate the database *demoDB*

```
db.inventory.insertMany( [
  { item: "canvas", qty: 100, size: { h: 28, w: 35.5, uom: "cm" }, status: "A" },
  { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },
  { item: "mat", qty: 85, size: { h: 27.9, w: 35.5, uom: "cm" }, status: "A" },
  { item: "mousepad", qty: 25, size: { h: 19, w: 22.85, uom: "cm" }, status: "P" },
  { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "P" },
  { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },
  { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },
  { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" },
  { item: "sketchbook", qty: 80, size: { h: 14, w: 21, uom: "cm" }, status: "A" },
  { item: "sketch pad", qty: 95, size: { h: 22.85, w: 30.5, uom: "cm" }, status: "A" }
] );
```

Select All Documents in a Collection

- Use db.inventory.find({}) to select all documents
- This operation corresponds to the following SQL statement:

  SELECT * FROM inventory

- Display results in an easy-to-read format.

  db.inventory.find().pretty()

- Count the number of documents

  db.inventory.count()

Specify Equality Condition

- To specify equality conditions, use <field>:<value> expressions
  - *db.inventory.find({ status: "D" })*

- This operation corresponds to the following SQL statement:

  SELECT * FROM inventory WHERE status = "D"

Specify Conditions Using Query Operators

- A query filter document can use the query operators to specify conditions in the following form:

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

Comparison

$eq	Matches values that are equal to a specified value.
$gt	Matches values that are greater than a specified value.
$gte	Matches values that are greater than or equal to a specified value.
$in	Matches any of the values specified in an array.
$lt	Matches values that are less than a specified value.
$lte	Matches values that are less than or equal to a specified value.
$ne	Matches all values that are not equal to a specified value.
$nin	Matches none of the values specified in an array.

Logical

$and Joins query clauses with a logical AND returns all documents that match the conditions of both clauses.

$not Inverts the effect of a query expression and returns documents that do not match the query expression.

$nor Joins query clauses with a logical NOR returns all documents that fail to match both clauses.

$or Joins query clauses with a logical OR returns all documents that match the conditions of either clause.

More at https://docs.mongodb.com/manual/reference/operator/query/#query-and-projection-operators

Specify AND conditions

- The following example retrieves all documents in the inventory collection where the status equals "A" and qty is less than ($lt) 30:

  db.inventory.find({ status: "A", qty: { $lt: 30 } })

- The operation corresponds to the following SQL statement:

  SELECT * FROM inventory WHERE status = "A" AND qty < 30

Specify OR Conditions

- The $or operator performs a logical OR operation on an array of two or more <expressions> and selects the documents that satisfy at least one of the <expressions>. The $or has the following syntax:

  { $or: [{ <expression1> }, { <expression2> }, ... , { <expressionN> }] }

- The following example retrieves all documents in the collection where the status equals "A" or qty is less than ($lt) 30:

  db.inventory.find({ $or: [{ status: "A" }, { qty: { $lt: 30 } }] })

Specify AND as well as OR Conditions

- In the following example, the compound query document selects all documents in the collection where the status equals "A" and either qty is less than ($lt) 30 or item starts with the character p:

```
db.inventory.find( {
     status: "A",
     $or: [ { qty: { $lt: 30 } }, { item: /^p/ } ]
} )
```

SELECT * FROM inventory WHERE status = "A" AND (qty < 30 OR item LIKE "p%")

$in Operator

- The $in operator selects the documents where the value of a field equals any value in the specified array

- The following example retrieves all documents from the inventory collection where status equals either "A" or "D":

  db.inventory.find({ status: { $in: ["A", "D"] } })

- The operation corresponds to the following SQL statement:

  SELECT * FROM inventory WHERE status in ("A", "D")

https://docs.mongodb.com/manual/reference/operator/query/in/

Update a Single Document

- The following example uses the db.collection.updateOne() method on the inventory collection to update the first document where item equals "paper":

```
db.inventory.updateOne(
   { item: "paper" },
   { $set: { "size.uom": "cm", status: "P" }},
)
```

- Check the result

```
db.inventory.find({item: "paper"})
```

Update Multiple Documents

- The following example uses the db.collection.updateMany() method on the inventory collection to update all documents where qty is less than 50:

```
db.inventory.updateMany(
  { "qty": { $lt: 50 } },
  { $set: { "size.uom": "in", status: "P" } }
)
```

Replace a Document

- The following example replaces the first document from the inventory collection where item: "paper":

```
db.inventory.replaceOne(
   { item: "paper" },
   { item: "paper", instock: [ { warehouse: "A", qty: 60 }, { warehouse: "B", qty: 40 } ] }
)
```

Delete All Documents

- The following example deletes all documents from the inventory collection:

  db.inventory.deleteMany({})

- The following example removes all documents from the inventory collection where the status field equals "A":

  db.inventory.deleteMany({ status : "A" })

Delete Only One Document that Matches a Condition

- The following example deletes the ***first*** document where status is "D":

  db.inventory.deleteOne({ status: "D" })

SQL to MongoDB Mapping Chart

- https://docs.mongodb.com/manual/reference/sql-comparison/#sql-to-mongodb-mapping-chart

- Very important!