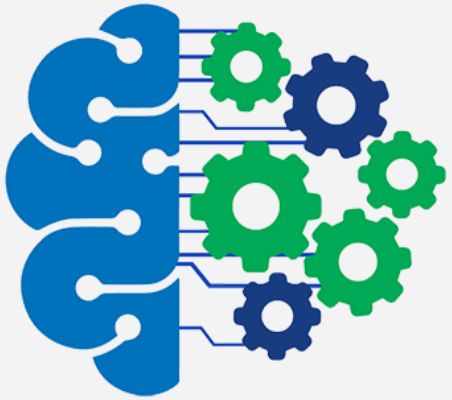
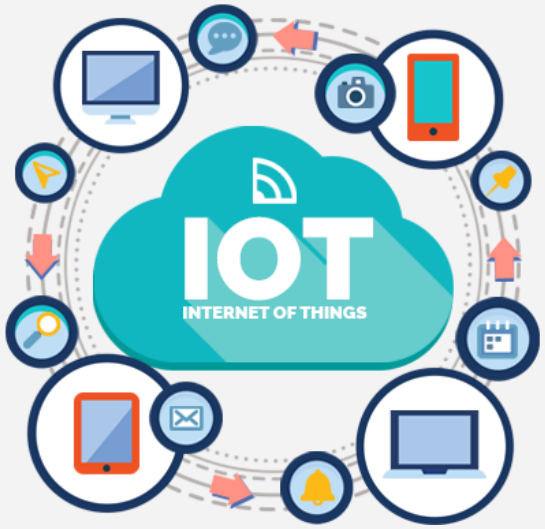
Introduction

By Eman Fathi


MACHINE
LEARNING

EVERYWHERE


IOT
INTERNET OF THINGS


JS
node
METEOR
ember
polymer
BACKBONE.JS
Knockout.
aurelia
ext js

THERE IS NOTHING YOU CAN'T DO WITH JAVASCRIPT (WELL, ALMOST...)

Front-end apps

Back-end apps

THIS COURSE

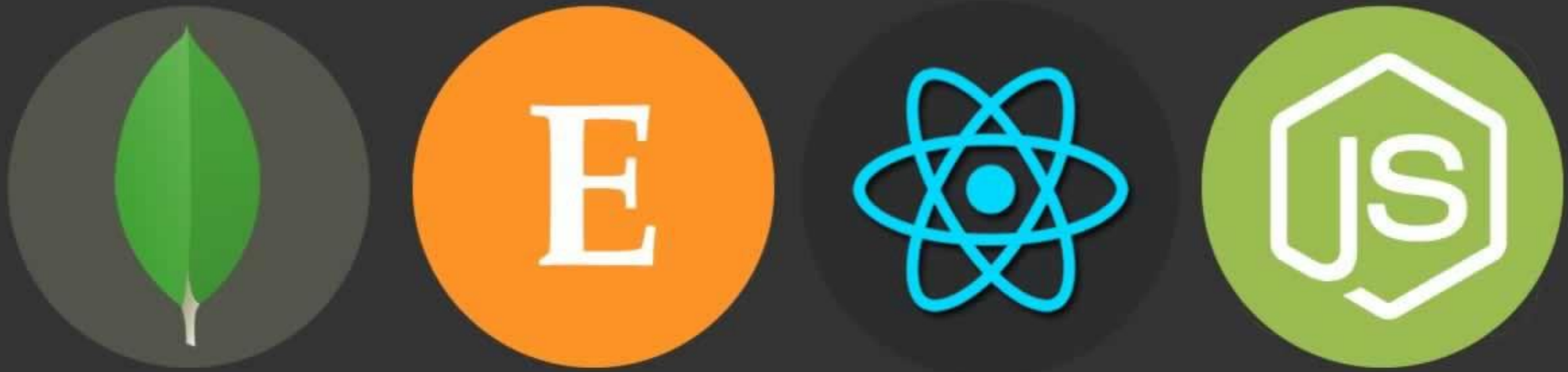
Dynamic effects and web applications in the browser 😍

Web applications on web servers

100% based on JavaScript. They might go away, **but JavaScript won't!**

Native mobile applications

Native desktop applications



M E R N

What is Mongo DB ?

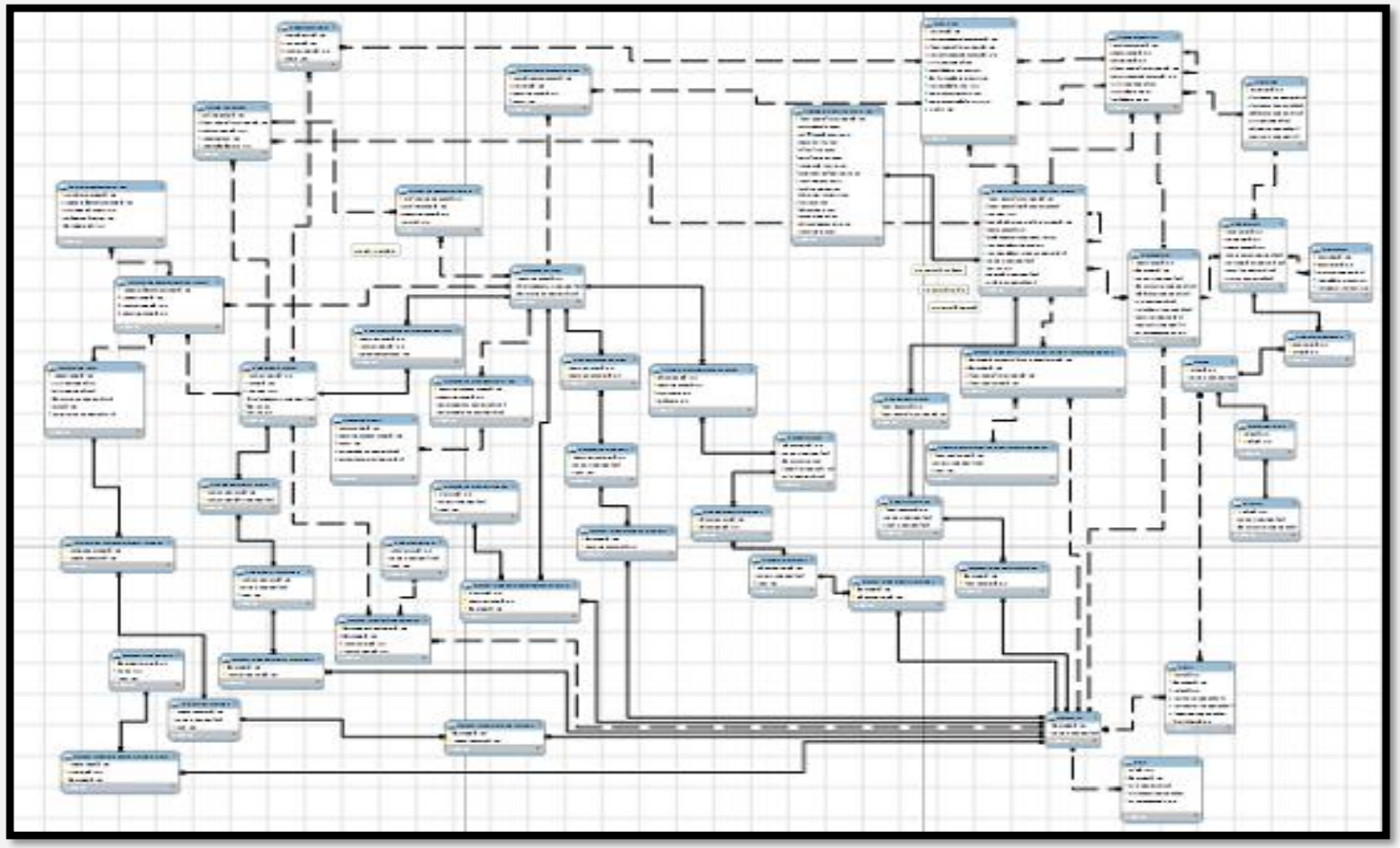
Hu*mongo*us

It can stores lots and lots of Data

CONCEPT OF NOSQL DATABASES GREW WITH INTERNET GIANTS

Gigantic volume of data

CONCEPT OF NOSQL DATABASES GREW WITH INTERNET GIANTS



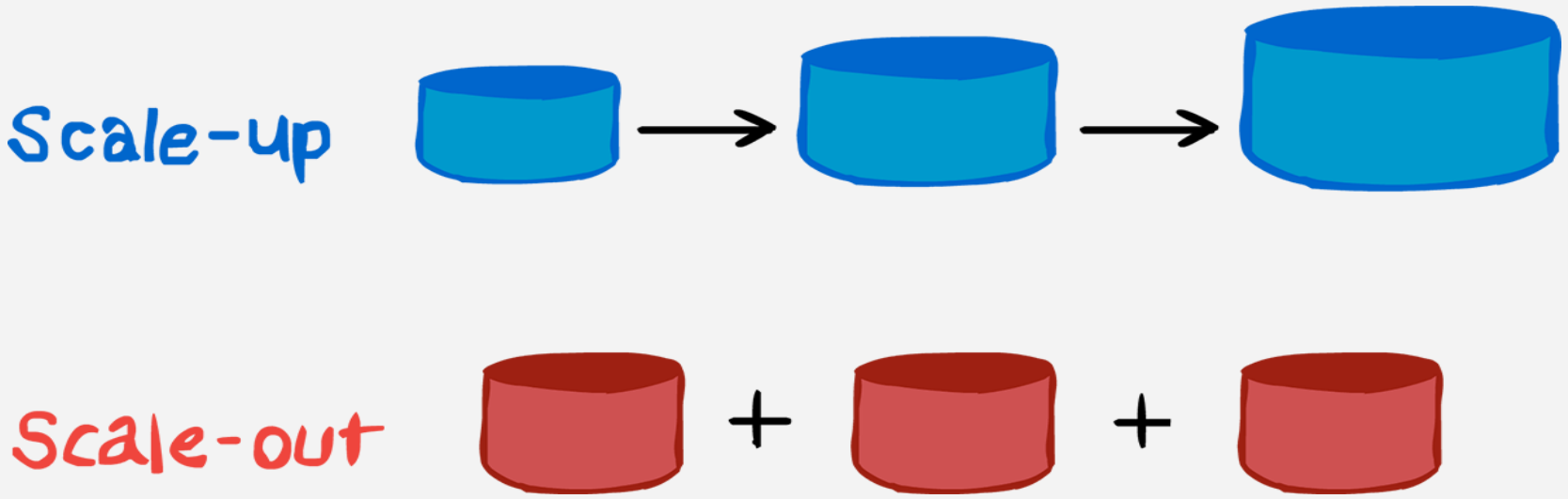
CONCEPT OF NOSQL DATABASES GREW WITH INTERNET GIANTS

Gigantic volume of data

RDBMS

Slow Response Time

CONCEPT OF NOSQL DATABASES GREW WITH INTERNET GIANTS




APACHE
HBASE
cassandra
Neo4j
Bigdata
No SQL
mongoDB
CouchDB
redis

NoSQL
Not Only SQL

Key Value

Usage:
Briskly changing data and high availability

Popular DBs:
Riak, Redis

1

Column Based

Usage:
Read/write extensions

Popular DBs:
HBase, Cassandra

2

Document Database

Usage:
Working with occasionally changing consistent data

Popular DBs:
Couchbase, MongoDB

3

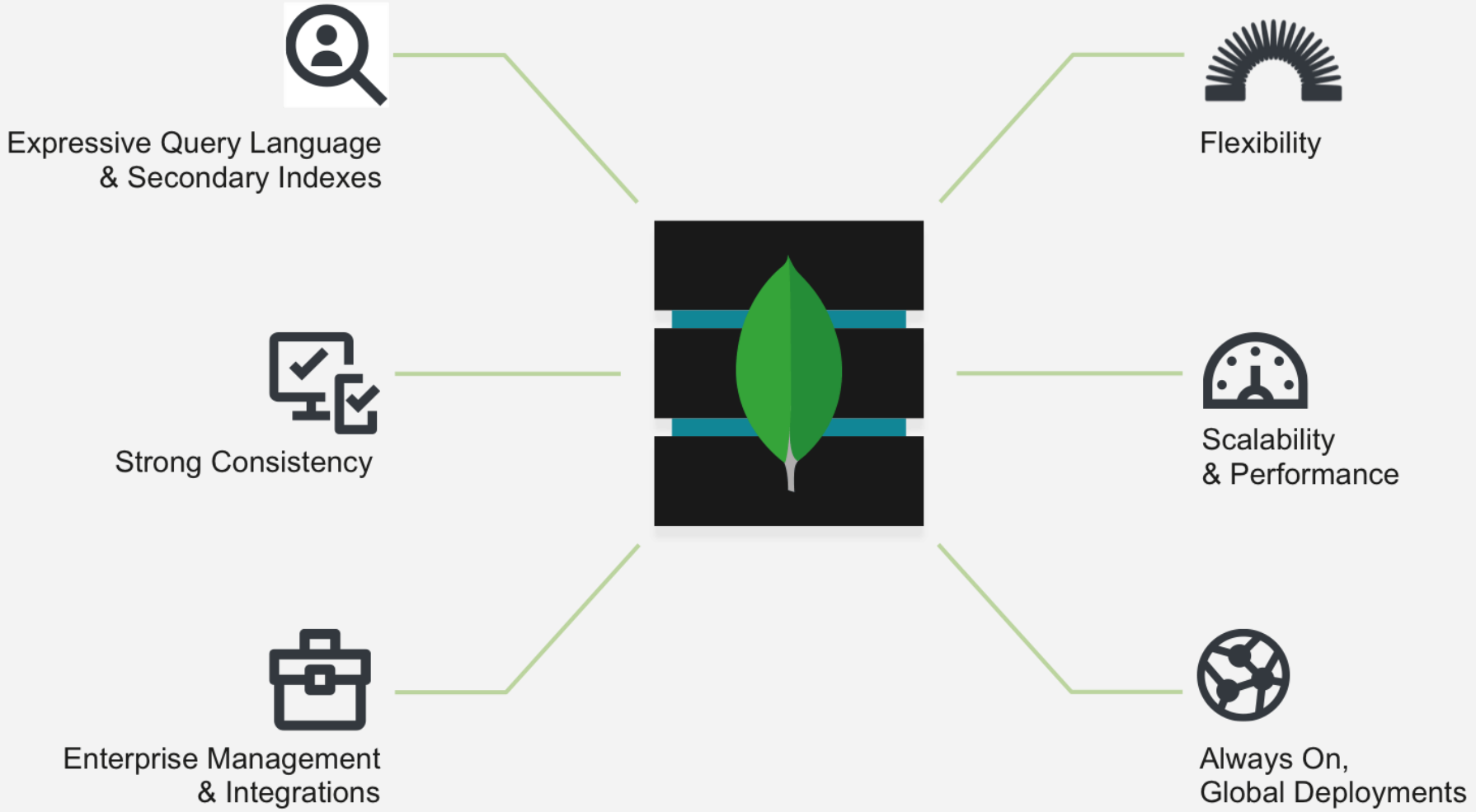
Graph Database

Usage:
Spatial data storage

Popular DBs:
Neo4J, Big data

4

NoSQL Features



Humongouse

Produced by 10gen company In 2007. In 2013 10gen renamed itself to MongoDB.

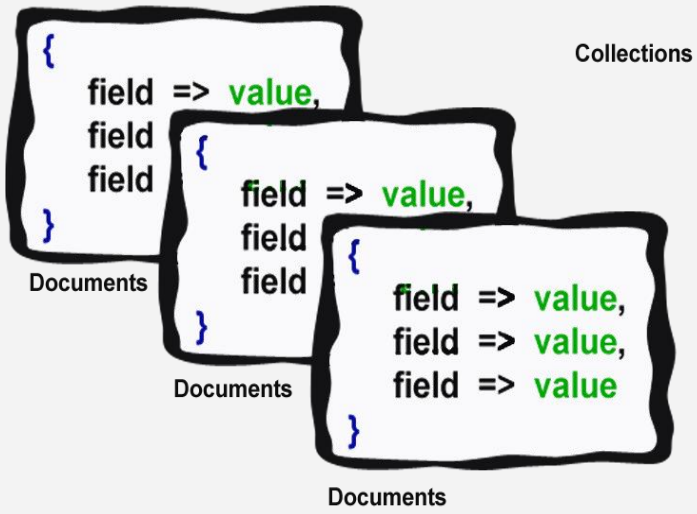
MongoDB is a **cross-platform**, document oriented database that provides, **high performance**, **high availability**, and **easy scalability**. MongoDB works on concept of collection and document.

MongoDB is a database management system designed to rapidly develop web applications and internet infrastructure. The data model and persistence strategies are built for high read-and-write throughput and the ability to scale easily with automatic failover. Whether an application requires just one database node or dozens of them, MongoDB can provide surprisingly good performance.

Humongouse

Collections

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table .Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.



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.

NoSQL

Not only SQL

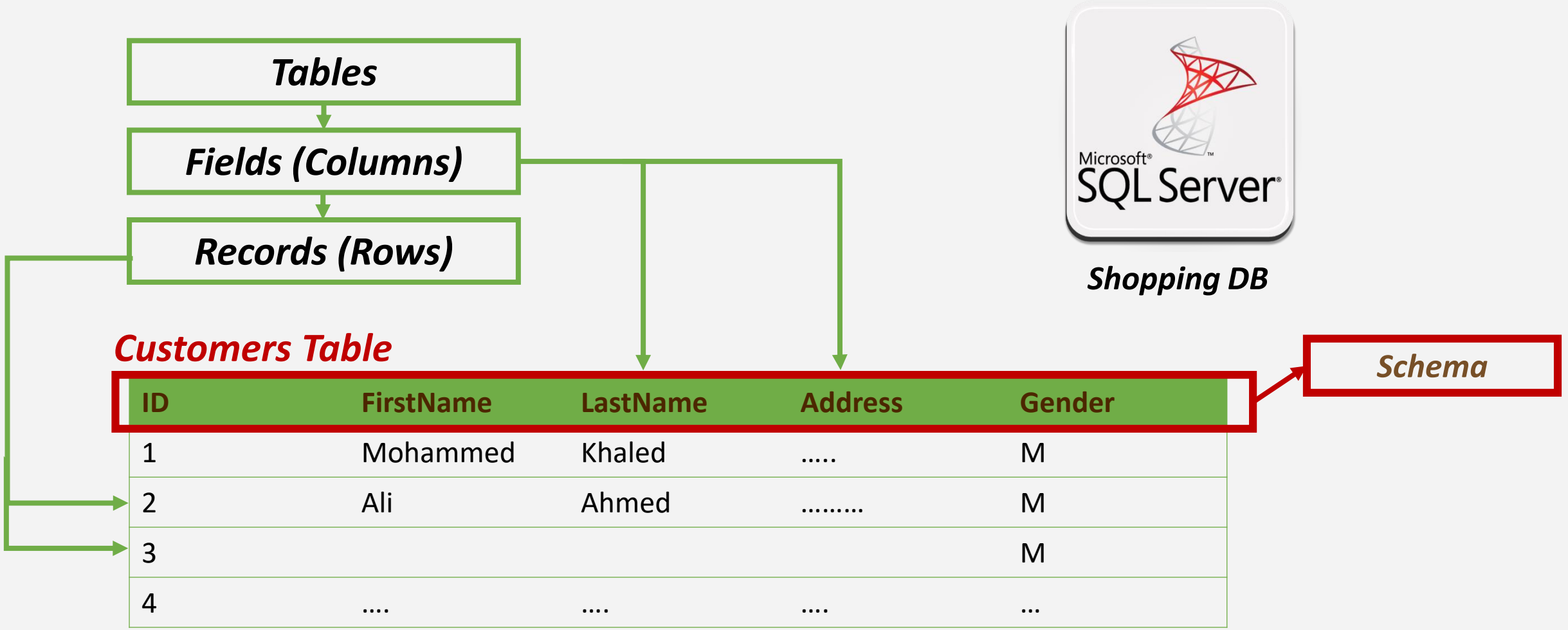
SQL

Relational Data Model

NoSQL

Document Data Model

SQL (Structure Query Language)



ID	FirstName	LastName	Address	Gender
1	Mohammed	Khaled	…..	M
2	Ali	Ahmed	……	M
3				M
4	….	….	….	…

SQL (Structure Query Language)

Relations

Customers

ID	*FirstName*	*LastName*	*Address*	*Gender*
1	Ali	Ahmed	……..	M
2	Hana	Mohamed	……	F
3				M

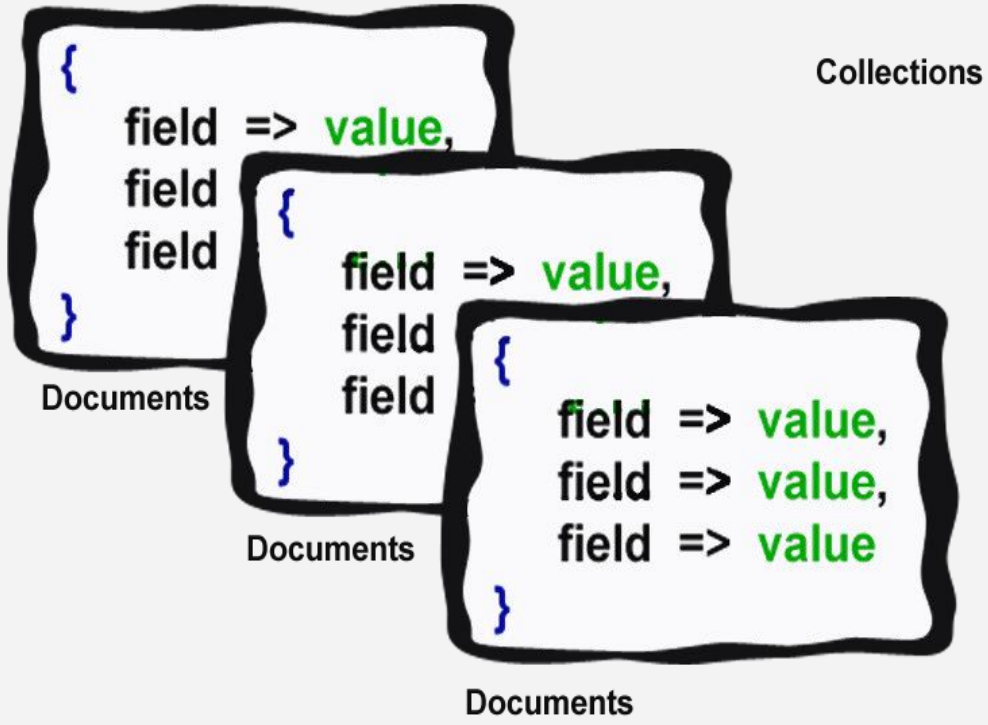
Orders

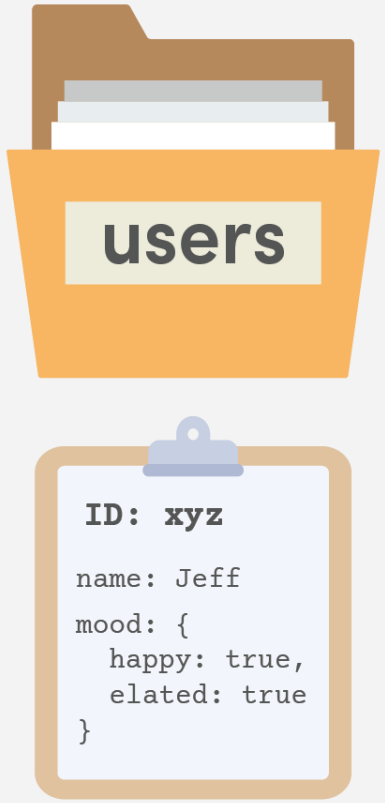
ID	*CustomerId*	*ProductId*
1	1	1
2	1	2
3	2	1

Products

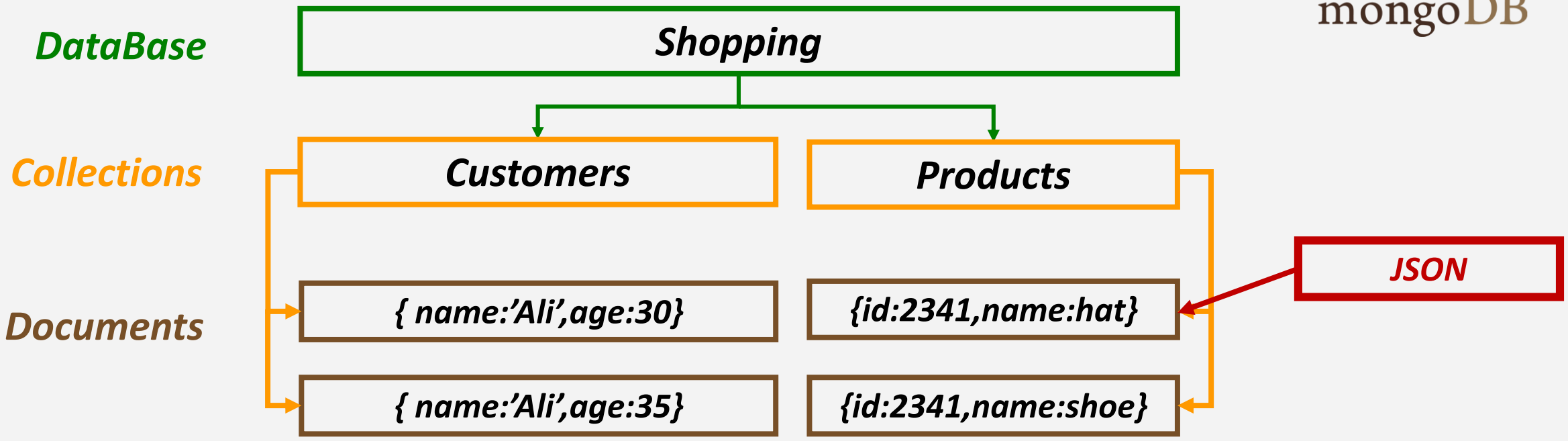
ID	*Price*	*Description*	*Quantity*
1	20.05	……	…….
2	597	…….	……
3	342.8	…….	……..

NoSQL (Not Only SQL)





NoSQL (Not Only SQL)



NoSQL (Not Only SQL)

No Schema

Documents

{id:1 name:'Ali',age:30}

{id:2 name:'Ahmed',age:35}

{id:4 name:'Khaled',email:Khaled@gmail.com}

NoSQL (Not Only SQL)

No/Few Relations

Customers

{id:1 name:'Ali',age:30}
{id:2 name:'Khaled',age:35}
{id:3 name:'Lara',age:30,email:lara@gmail.com}
{……}

Products

{id:1,title:'hat',price:120}
{id:2,title:'shoe',price:320}
{……}

Orders

{id:14 , customer : {id:3,email:lara@gmail.com} , product : 3}
{id:15 , customer : {id:2,name:"Khaled"} , product : 3}
{……}

SQl Server	MongoDB
Tables	Collections
Rows	Documents
coulmns	Fields
Joins	Embeded Documents
Primary Keys	PrimaryKey providd by mongodb itself

SQl

Data Schemas

Data distributed across multiple tablesd

Relations

Horzontal scalling is difficult, vertical is possible

Limitations for huge numbers of read/write queries /second

NoSQL

Schemas less

Data merged in few collections

No/few Relations

Both horizontal & vertical scalling are possible

Great performance for huge read/write requests

https://www.mongodb.com/

Community Server

Installation

Running Through Mongo Shell
All instances of MongoDB come with a command line program we can use to interact with our database using Javascript

MongoDB server (mongod)

After Installation

mongod is the basic process for mongoDB system. It handles data requests ,manages data access and performs background management operations.

By default it listens to port 27017.

mongod runs with some options as:
--dbpath
--port
--maxconns

{JSON}

Documents are JSON-Like Objects

MongoDB save Data in documents in a format called BSON

BSON { 01010100 11101011 10101110 01010101 }

Binary JSON

MongoDB BSON

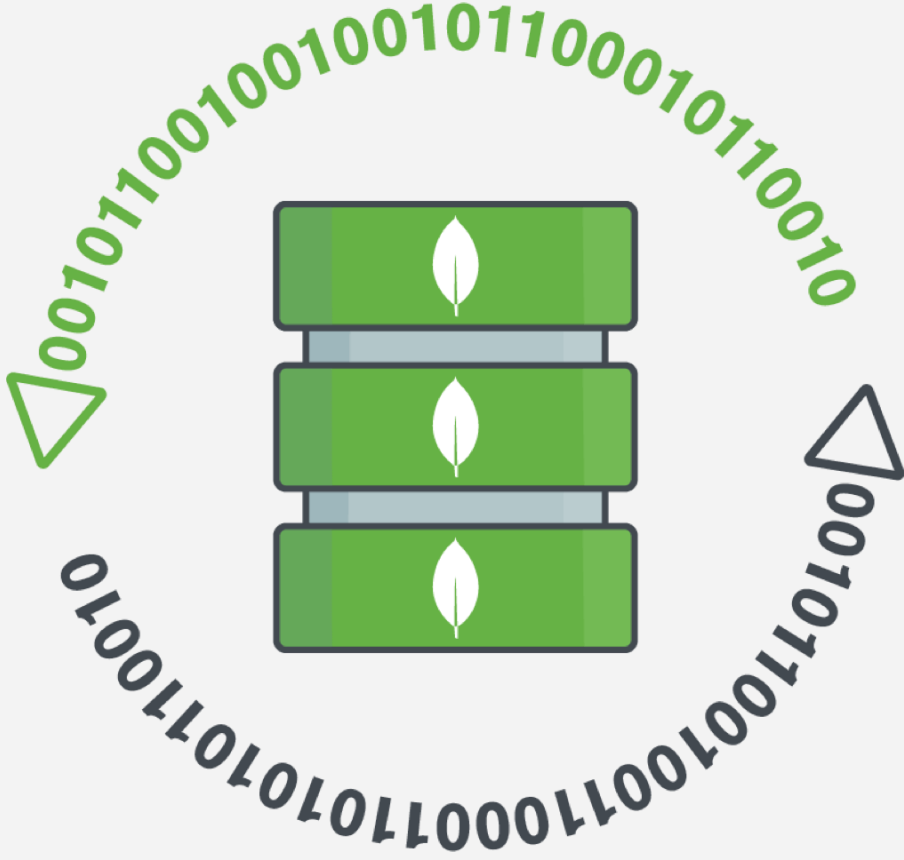
How mongodb respresents data and how mongo shell interpret that data comming from database?

Mongogdb does not use stringly format for storing or retrieving data.

Instead it uses binary representation to store data inside documents BSON

BSON is a binary representation for json and support Data types not in JSON like BinData , ObjectID and Timestamp.

bsonspec.org



Collections

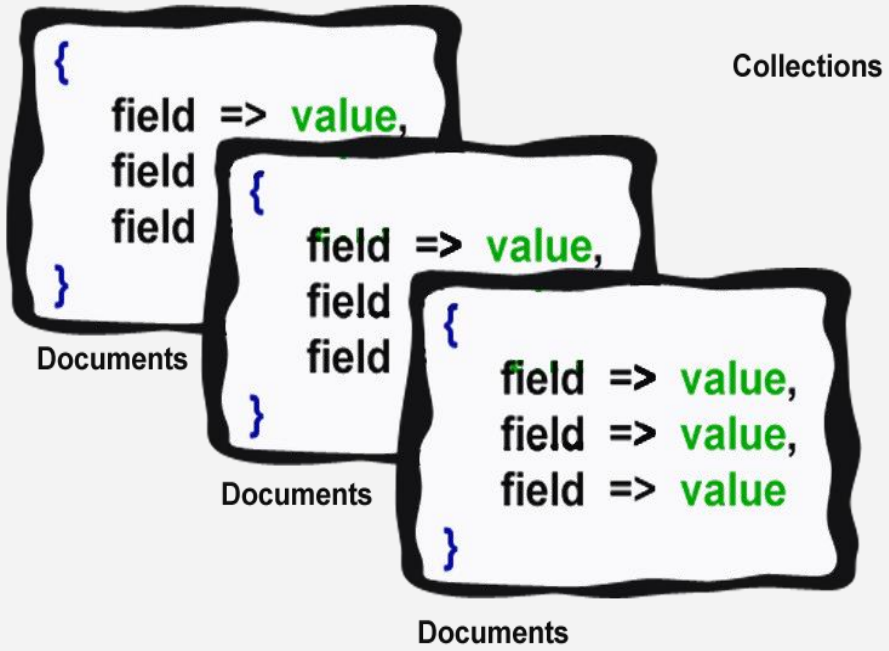
```
db.createCollection("customers");
```

```
show collections
```

```
db.getCollectionNames()
```

```
db.getCollectionInfos({name:'employees'})
```

```
db.customers.drop()
```



CREATE READ UPDATE DELETE

C R U D

CREATE

db.collection.insert();

insert(): insert document or documents into a collection

insertOne(): insert only one document into a collection

insertMany(): insert multiple documents into a collection

```
db.employees.insert({name:"Maha",salary:1000});
db.employees.insert({name:"Maha",salary:1000},{name:"Ali",age:33});
db.employees.insert([{name:"Maha",salary:1000},{name:"Ali",age:33}]);
```

```
db.employees.insertOne({name:"Maha",salary:1000});
db.employees.insertMany([{name:"Maha",salary:1000},{name:"Ali",age:33}]);
```

ObjectId

The ObjectId Class is the default primary key for a MongoDB document and is found in _id field in an inserted document.

```
{
    "_id": ObjectId("54759eb3c090d83494e2d804")
}
```

- ✓Immutable
- ✓Unique
- ✓BSON DataType
- ✓12 byte Value

DELETE

db.collection.remove();

Remove all documents

```
db.customers.remove({});
```

Remove a document according to specified condition

```
db.customers.remove({salary:{$gt:1000}});
```

READ

db.collection.find(query,projection);

find(): returns all matched documents , if there is no condition returns all collections documents

findOne(): return only one matching document , the first matching document

```
db.employees.find();
```
All collection documents

```
db.employees.find({name:"eman"})
```
all documents with name =‘eman’

```
db.employees.find({age:{$lt:20}})
```
all documents with age<20

```
db.employees.findOne({age:{$lt:20}})
```
first document with age<20

```
db.employees.find({}).pretty()
```
result as readable json objects

Operators

Comparisons operators

Array operators

logical operators

element operators

Update operators

Comparisons Query operators

Name	Description
$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.

Comparisons Query operators

All documents that have salaries less than 2000

```
db.employees.find({salary:{$lt:2000}});
```

All documents that have salaries greater than or equal 2000

```
db.employees.find({salary:{$gte:2810}});
```

All documents that have salaries between 1000 and 3000

```
db.employees.find({salary:{$lt:3000,$gt:1000}});
```

Any document has age value inside $in array

```
db.employees.find({age:{$in:[22,35,25,40]}});
```

All documents that have salaries equal 2810

```
db.employees.find({salary:{$eq:2810}});
```

Logical Query operators

Name	Description
$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.

Logical Query operators

```
db.employees.find({$or:[{name:"hassan"},{salary:3030}]});
```

```
db.employees.find({$and:[{name:"hassan"},{salary:{$gte:30}}]});
```

```
db.employees.find({$nor:[{salary:2810},{name:"hassan"}]});
```

```
db.employees.find({salary:{$not:{$gt:3000}}});
```

Element Query operators

Name	Description
$exists	Matches documents that have the specified field.
$type	Selects documents if a field is of the specified type.

```
db.employees.find({count:{$type:"string"}});
```

```
db.employees.find({count:{$exists:""}});
```

Array Query operators

Name	Description
$all	Matches arrays that contain all elements specified in the query.
$elemMatch	Selects documents if element in the array field matches all the specified $elemMatchconditions.
$size	Selects documents if the array field is a specified size.

```
db.employees.find({dresses:{$all:["black","red"]}});
```

```
db.employees.find({data:{$elemMatch:{$gt:3,$lt:6}}});
```

```
db.employees.find({dresses:{$size:5}})
```

Projection

projection means selecting only the necessary data rather than selecting whole of the data of a document.

```
db.employees.find({name:"eman"},{salary:0});
```

```
db.employees.find({name:"eman"},{salary:false});
```

```
db.employees.find({name:"eman"},{salary:true});
```

```
db.employees.find({name:"eman"},{salary:1});
```

UPDATE

db.collection.update(query,update,option);

update(): updates one documents or documents in a collection

updateOne(): updates single document in a collection

updateMany(): updates multiple documents in a collection

```
db.employees.update({name:"eman"},{$set:{salary:5600}});
```

```
db.employees.update({name:"eman"},{$set:{salary:5600,age:20}});
```

```
db.employees.update({name:"eman"},{$set:{address:"Mansoura"}},{upsert:true});
```

```
db.employees.update({age:{$gt:20}},{$set:{count:1}},{multi:true});
```

operators

Field Operators

Name	Description
$inc	Increments the value of the field by the specified amount.
$min	Only updates the field if the specified value is less than the existing field value.
$max	Only updates the field if the specified value is greater than the existing field value.
$mul	Multiplies the value of the field by the specified amount.
$set	Sets the value of a field in a document.
$unset	Removes the specified field from a document.

UPDATE

Field Operators

```
db.employees.update({name:"eman"},{$set:{salary:5600},$inc:{count:1}});
```

```
db.employees.update({name:"eman"},{$max:{count:3}});
```

```
db.employees.update({name:"eman"},{$min:{count:3}});
```

```
db.employees.update({name:"eman"},{$mul:{count:3}});
```

```
db.employees.update({},{$unset:{color:"red"}},{multi:true});
```

```
db.employees.update({},{$rename:{"count":"counter"}},{multi:true});
```

operators

Array Operators

Name	Description
$	Acts as a placeholder to update the first element that matches the query condition.
$[]	Acts as a placeholder to update all elements in an array for the documents that match the query condition.
$pop	Removes the first or last item of an array.
$pull	Removes all array elements that match a specified query.
$push	Adds an item to an array.

operators

Array Operators

```
db.employees.update({name:"eman"},{$set:{"books.2":"JQuery"}});
```

```
db.employees.update({name:"eman"},{$set:{"books.$":"ES"}})
```

```
db.employees.update({name:"eman"},{$set:{"books.$[]":"ES"}})
```

```
db.employees.update({name:"eman"},{$push:{books:"nodejs"}});
```

Introduction

Eman Fathi

Information Technology Institute