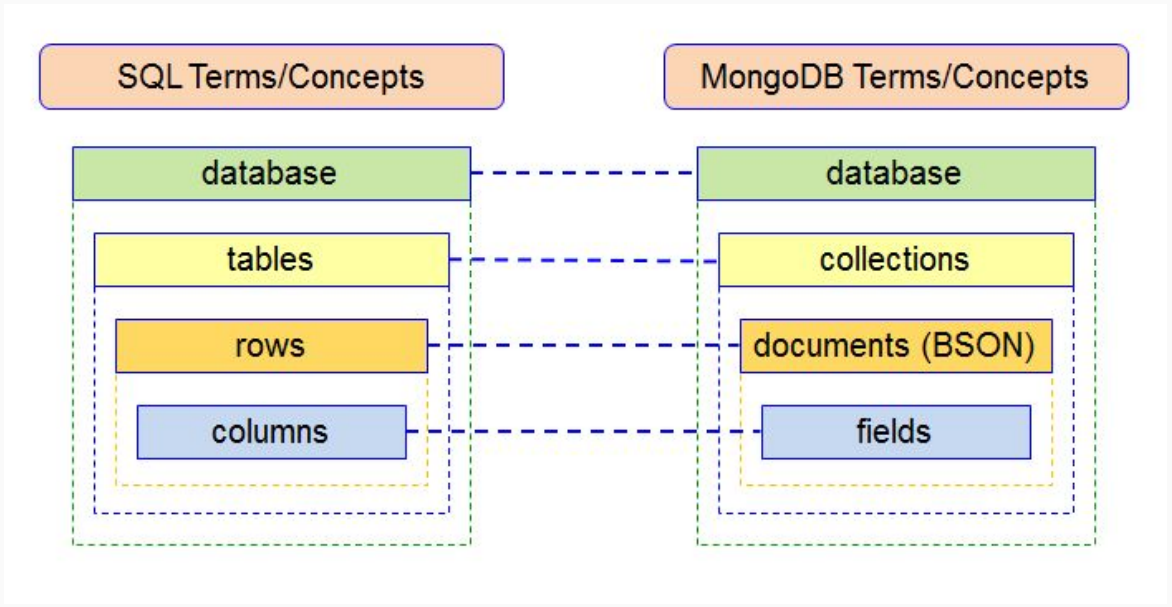
NoSQL

- NoSQL stands for Not only SQL. MongoDB is a flavor of NoSQL, like PosgreSQL is a flavor of SQL.
    - A NoSQL paradigm may be preferable to SQL because it is schemaless.
    - Great for storing unstructured data, as we may find on the web!
    - MongoDB is a document-oriented DBMS.



General Commands for Inspecting Mongo

galvanize

```
help                          // List top level mongo commands

db.help()                     // List database level mongo commands

db.<collection name>.help()   // List collection level mongo commands.

show dbs                      // Get list of databases on your system

use <database name> *          // Change current database

show collections              // List collections in current database
```

* mongo will make a database if you use the wrong name

Inserting

galvanize

Once you're using a database you refer to it with the name db. Collections within databases are accessible through dot notation.

```
db.users.insert({ name: 'Jon', age: '45', friends: [ 'Henry', 'Ashley']})

db.getCollectionNames()  // Another way to get collection list

db.users.insert({ name: 'Ashley', age: '37', friends: [ 'Jon', 'Henry']})
db.users.insert({ name: 'Frank', age: '17',
                  friends: [ 'Billy'], car : 'Civic'})

db.users.find()
```

* Note: The three documents that we inserted into the above database didn't all have the same fields.
    * Note: Mongo creates an _id field for each document if one isn't provided.

Querying

When querying from mongo the first parameter can be thought of like the WHERE in a SQL query. The second parameter is like the SELECT

```
db.users.find({ name: 'Jon'}).limit(5)       // find by single field

db.users.find({ car: { $exists : true } })  // find by presence of field

db.users.find({ friends: 'Henry' })         // find by value in array

db.users.find({}, { name: true })   // field selection (only return name)
```

A quick way to figure out how to write a Mongo query is to think about how you would do it in SQL and check out a resource like this Mongo endorsed conversion guide
Or use something like a query translator

Updating

galvanize

```
// replaces friends array
db.users.update({name: "Jon"}, { $set: {friends: ["Phil"]}})

// adds to friends array
db.users.update({name: "Jon"}, { $push: {friends: "Susie"}})

// upsert
db.users.update({name: "Stevie"}, { $push: {friends: "Nicks"}}, true)

// multiple updates
db.users.update({}, { $set: { activated : false } }, false, true)
```

https://docs.mongodb.com/manual/reference/method/db.collection.update/

Aggregation

Aggregations in Mongo end up being way less pretty than in SQL/Pandas. Let's just bite the bullet and take a look:

```
db.users.aggregate( [ {
     $group :
     {
         _id: "$name",
         count: { $sum: 1 }
     }
}])
```

Aggregation

A more complex example:

```
db.users.aggregate([
     { $match: { name: {$in : ["Dan", "Sam"] } } },
     { $group: { _id: "$name", total: { $sum: "$age" } } },
      { $sort: { total: 1 } }
   ])
```

Think of the parts of this as:
$match == WHERE in SQL
$group == GROUP BY and aggregation function
$sort   == SORT BY

https://docs.mongodb.com/manual/reference/sql-aggregation-comparison

Mongo Updates

galvanize

```
# Update the **first instance** with name Mulan,
# setting their age to 29.
db.users.update({name: "Dan"}, {$set : {age: 29}})

# Update all instances.
db.users.update({name: "Sam"}, {$set : {age :29}},
                                {multi: true})

# Update all instances found, or insert a document
# if not found.
db.users.update({name: "Peter"}, {$set : {age :29}},
                                {multi: true, upsert: true})
```

Mongo Shell and JavaScript

galvanize

The Mongo Shell is technically a JavaScript shell, meaning it allows any valid JavaScript (JS) code. That includes loops:

```
#  Update all those with name "Dan" to have name
# "Daniel"

db.users.find({name: "Dan"}).
                forEach(function(doc) {
    {
       id = doc._id;
       new_name = doc.name.replace("Dan", "Daniel");
       db.users.update({ _id : id}, {$set: {name: new_name}})
    };
 });
```

To keep data in a Mongo Docker container after shutting it down:

```
docker run --name mongoserver -d
-p 27017:27017
-v $HOME/docker/volumes/mongo:/home/data
mongo
```

To connect without loading mongo shell on local

```
docker exec -it mongoserver mongo
```