Not Only SQL (noSQL)

Before Starting

Required Software: To complete this lecture, you will need to install the following software:

1. Mongosh: The command line interface used to interact with the database
2. MongoDB Compass: The GUI interface to interact with the database
3. Community services: The actual database

As you will see NoSQL offers something similar to Structured Query Language (SQL).

Types of NoSQL databases

- There are many types of NoSQL databases e.g.
    1. Document
    2. Key Value Pair
    3. Graph
- For our discussion we will focus on the **document** TYPE

NoSQL VS Relational DATABASE

NoSQL

- No Schema
- Flexible
- Scales Horizontally:
    - When system grows and reaches a certain threshold
    - To increase performance and storage capacities
        - Is achieved by adding more low-end computers to satisfy increased demands,
        - Results in no downtime.
- Data can still be related in this model
- No Foreign Keys (FKs)
- No JOINS: information that needs be displayed together is stored together
- No datatypes defined during object creation

Relational

- Schema
- Rigid
- Scales Vertically:
    - When system grows and reaches a certain threshold
    - Increase performance and storage capacities is done by upgrading hardware
    - e.g., increasing CPUs, storage (more SSDs)
- Uses FKs
- JOINS
- Datatypes must be defined

Use Cases:

Relational

- Business model/requirements well understood and defined
- Data is structured
- Developers work in Object Relational Mapping *(relating tables)*
- Increased Cost of ownership

NoSQL (Document db)

- Not much is well understood or finalized about the database requirements or requirements changing frequently
- Data is unstructured
- Developers don't use Object Relational Mapping in development
- Lower Cost of ownership (Open Source)

Terms

SQL Terms	NoSQL Terms
Database	Database
ROW/Record	Document
Primary key	Primary Key
Table/Relation	Collection
Column/Field	Field

Ways to interact with MongoDB

1. **Mongo Shell**: Accessed via Windows Command Prompt - type cmd into Windows Search
2. **MongoDB Compass:** Desktop Based GUI
3. **MongoDB Atlas**: Database as a service *(Db stored in cloud)*

- Let's start with #1. Windows Command Prompt
  - **Try it!** Type **mongosh** into cmd.
    - **Trouble?** If an error occurs
      - ✓ Open **services** by clicking on Start and type **'Services'**
      - ✓ Check if the service called **'MongoDB Server'** is running. If necessary, start the service.

Basic commands

- **Mongosh**: this will load mongo shell
- **cls**: clear screen
- **help**: Help command to show which options are available

Some comments available through help menu include:

- **db**: to show current database context
- **show dbs**: to show all databases
- **show collections**: display current collections in database context
- **show users**: show users in database context

Create the Database

- Using our sample **JR_Movie** database we will create an equivalent NoSQL Database for it

- **Try it!** Type **use JR_Movie** to create a *temporary* database
  - **Note:** No CREATE statement in Mongo, use Context will create it
- To show the current database context: Type **db**
  - JR_Movie is displayed

- To display all of the databases that we have presently: Type **show dbs**
  - JR_Movie not shown, why?
    - Presently all we have a is temporary database
    - Mongo won't create database until data is inserted into it

- **Best Practice:** When working with Mongo SQL Language (MSL) it's good idea to write statements in separate text or json FILE and then paste into CMD

To Drop a databasedb.dropuse

- **Try it!** To Drop JR_Movie Database type in the following
  - use JR_Movie
    - Ensures that you are presently using the database that you want to drop
  - db.dropDatabase()
    - **Trouble?** This command is case sensitive. Try again but pay attention to case
  - If successful should receive { "ok" : 1 }

Inserting a single document (row) into the collection (table)

- Copy the code below and go to the command prompt and right click
  - **Note:** A single right click in the cmd will paste

```
db.Customer.insertOne({
CUSTID: 23,
FNAME:"Mike",
LNAME: "Poitras",
STREETNO: "123",
STREET: "Main St",
CITY: "Halifax",
PROVINCE: "NS",
PCODE: "B3N2J2",
PRIMARYCUSTID: null
})
```

- **Customer** refers to a collection (called a table in relation databases)

- **Try it!** Does our database appear now?
  - Type**: show dbs**
  - Yes! JR_Movie now appears as a database, since data was written; size appears next to database name

To query a collection

- **Try it!**
  - To retrieve all rows in Customer collection, similar to SELECT * from customer;
    - db.Customer.find()
  - To retrieve rows where lname = "Poitras", similar to SELECT * from customer where LNAME="Poitras"
    - db.Customer.find({LNAME: "Poitras"})
  - **Trouble?** Case matters! I.E. A collection named **customer** is not the same as a collection named **Customer**

Inserting many documents (rows) into the collection (table)

Try it! Copy/Paste the following 3 documents into the Customer collection

```
db.Customer.insertMany(
[
{
        CUSTID: 23,
        FNAME:"Al",
        LNAME: "Dente",
        STREETNO: "17",
        STREET: "Rosedale Ave",
        CITY: "Halifx",
        PROVINCE: "NS",
        PCODE: "B3N2J2",
        PRIMARYCUSTID: null
        },
{
        CUSTID: 24,
        FNAME:"Zac",
        LNAME: "Oleeskey",
        STREETNO: "5501",
        STREET: "Leeds St",
        CITY: "Halifx",
        PROVINCE: "NS",
        PCODE: "B3K2T3",
        PRIMARYCUSTID: null
        },
{
        CUSTID: 29,
        FNAME:"Alan",
        LNAME: "Celone",
        STREETNO: "2230",
        STREET: "Gottingen Rd",
        CITY: "Halifx",
        PROVINCE: "NS",
        PCODE: "B3K3C6",
        PRIMARYCUSTID: null
        },
]
)
```

Repeat our queries of the Customer collection

- **Try it!**
  - Retrieve all rows in Customer collection, similar to SELECT * from customer;

db.Customer.find()

  - Retrieve rows where lname = "Celone", similar to SELECT * from customer

db.Customer.find({LNAME: "Celone"})

Introduction to MongoDB Compass (GUI for Mongo DB)

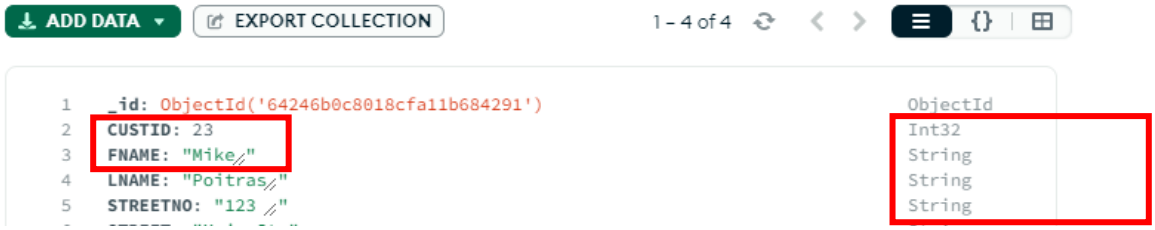
Procedure.

1. Type Mongo into Windows Search and open MongoDB Compass
2. For local connection, verify that the URI has something similar **mongodb://localhost:27017**
3. Press **CONNECT**
4. In the left pane: Click on **Databases**, then on **JR_Movie** Database
5. Click on the **Customer** Collection (table)

- You should now see 4 documents that had been added to the customer collection

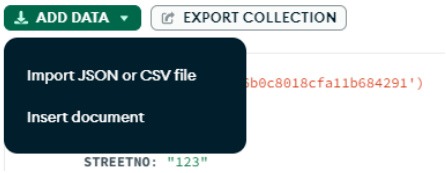
To learn which datatypes are used for each field in the document *(list view)*

- Double click any **field** name
- Datatypes will be shown on the right

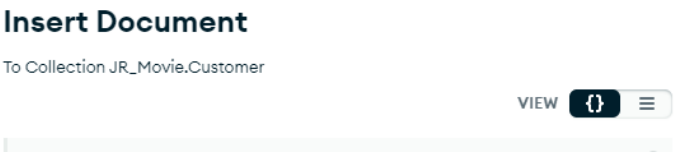


To Add Data

- Click on **ADD Data** button, then choose **Insert Document**

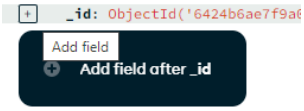


- Two options become available in the top right corner next to **VIEW**
  1. json OBJECT
  2. List type

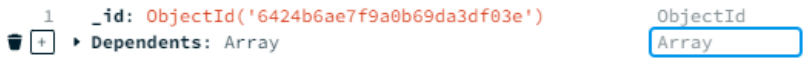


- Let's use **list type** here to create a new document (the 2nd option)

1. Click on **+** sign left of column name and click **Add field after id** to add a new FIELD



2. Enter the word **Dependents**
3. Change the datatype on the right to Array



4. Click the plus sign and click **Add item to Dependents**. Enter "Amy Smith" as item #0
5. Enter "Ben Smith" as item #1
6. Click **Insert**
   - Finished it should look like this...

```
_id: ObjectId('6424bb197f9a0b69da3df042')
▾ Dependents: Array
    0: "Amy Smith"
    1: "Ben Smith"
```

Advantage using NoSQL

- NoSQL offers flexibility that we can't get with a relational database

- **Try it!** Insert the **JSON** object below:
  - What is JSON? *(JavaScript Object Notation)* Uses a **key/Value** pair that is separated by comma
  - Click **Add Data**
  - Select **Insert Document**
  - Select **JSON** if necessary


```
{
      "_id": {
         "$oid": "6283e069a26d49b2dc76c3a5"
      },
        "CUSTID": 100,
        "FNAME": "Joe",
        "LNAME": "Smith",
        "DEPENDENTS": [
              "Amy Smith",
              "Ben Smith"
        ],
        "Projects": {
              "name": "Mars"
      }
}
```

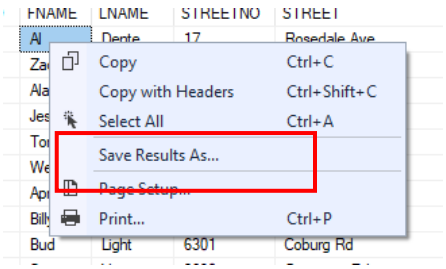
- **Try it!** Let's view our new document using Mongosh
  - Go back to command prompt and type **db.Customer.find({LNAME: "Smith"})**
    - **Trouble?** Did you remember to type the collection name (Customer) with a capital 'C'?

  - Now type **db.Customer.find({LNAME: "Smith"}).pretty()**
    - **Note:** The **pretty()** function is used here to make the JSON more readable
    - *It's supposed to but I personally don't see the difference.* ☺

Importing with MongoDB Compass

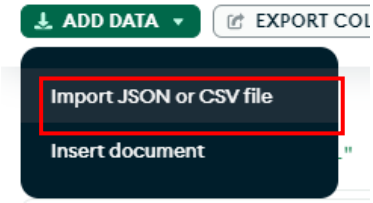
- MongoDB Compass can only insert ONE document at a time UNLESS you are importing from file
  - E.G. JSON or csv

- **Try it!** Import a csv file into MongoDB Compass
  - **Export the customers table from SSMS**
    1. Open SSMS and query all of the customer table: **SELECT * FROM Customer**
    2. Right click on the result tab and select **Save Results As**

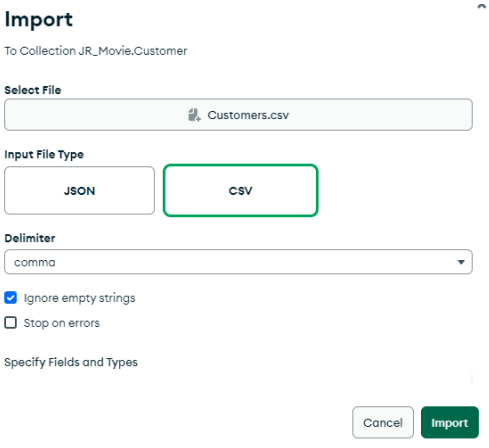


    3. From the Save As dialog select a file type of **csv** and name the file customers

- Import the data from the csv file into the Mongo DB
  1. From MongoDB Compass click on Add Data >> Import JSON or CSV file



  2. Select the Customer.csv file. From the Import dialog ensure that file type is **csv**, the delimiter is **comma** and **Ignore empty strings** is selected. Click on **Import**



*/

Confirming a successful import

- **Try it!** Running the following commands in cmd should reflect all newly imported documents in the database

```
db.Customer.countDocuments()
db.Customer.find()
```

  - **Trouble?** Remember it is case sensitive: **customer** is not the same as **Customer**