MongoDB & Neo4j Guide

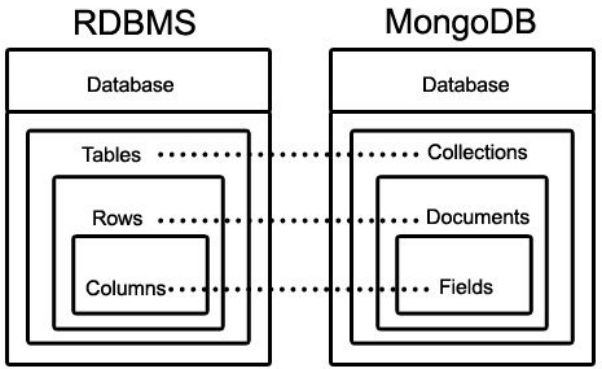
MongoDB Installations

- OS recommendation: Linux Ubuntu 20.04 x64
- Canvas→ Modules→ MongoDB, download the following files(For Ubuntu 20.04 x64 Only):
  *mongodb-org-server_**4.4.3**_amd64.deb, mongodb-org-shell_**4.4.3**_amd64.deb*
- MacOS, Debian, Ubuntu < 20.04. Download the server and shell version 4.4.3 from here:
  https://www.mongodb.com/download-center/community/releases/archive
- Install the .deb package:
  $ sudo apt install /path/to/file
- Start the Mongodb server:
  $ sudo systemctl start mongod
- Test MongoDB connection
  $ mongo

```
kali@ubuntu:~/Desktop/MongoDB/Example$ mongo
MongoDB shell version v4.4.3
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("5e535f8f-15f0-410e-9c35-77636cd6a089") }
MongoDB server version: 4.4.3
```

MongoDB

- Collections & Documents
- Document format: BSON (Binary-JSON)
  - Binary encoding of JSON documents
  - Support more data types: Date, binary byte array.
- Documents v.s Rows
  - Document can have different structure
  - Document can have embedded documents



Student Collection #1

```
{
 "name" : "student1",
 "type" : "boarder",
 "room#" : "West Hall 130"
},

{
 "name" : "student2",
 "type" : "day student",
 "homeAddress" : "Socorro"
},
```

Student Collection #2

```
{
 "name" : "student1",
 "Transcript": {
                    "course1": {
                                  "grades" : "A",
                                     .
                                     .
                                  },
                    "course2": ...
                  }
}
```

MongoDB Example With Python

- To simplify the data import, data are stored as python list, with dictionaries embedded.

- Python dictionary
  - A dictionary is as simple as placing items inside curly braces **{ }** separated by commas.
  - An item has a key and a corresponding value that is expressed as a pair (key: value).
  - Values: can be of any data object and can repeat.
  - Keys: immutable object, can not repeat
    - `my_dict = {1: 'Hello', 2: 'World'}`
    - `my_dict = {1: {'A': 2}}`

- Python list
  - A list is created by placing all the elements inside square brackets **[ ]**, separated by commas.
    - `my_list = [1, 2, 3, 4]`
    - `my_list = [1, "Hello", 3.4]`
    - `my_list = [{1: 'Hello', 2: 'World'}, {1: 'Hi', 2: 'World'}]`

MongoDB Example With Python

- Example datasets: Canvas→ Modules→ MongoDB→ data.py

department

```
[
      {
       "name": "",
       "id": int,
       "phone#": "",
       "course_offer": []
      }
]
```

course

```
[
      {
       "name": "",
       "id": "",
       "requisite": {
                    "co-requisite": [],
                    "prerequisite": []
                    }
       "department_id": int,
       "credit": "",
       "description": ""
      }
]
```

MongoDB Example With Python

- Example python script Canvas→ Modules→ MongoDB→ mongoDB.py
- Line 5-23:
  - Establish a connection to MongoDB,
  - Create database "myDB"
  - Create two collections "departmentCol" and "courseCatalogCol".

- Import "departmentList" and "courseList" from data.py, and insert to collections.
  departmentCol.insert_many(departmentList)
  courseCatalogCol.insert_many(courseList)

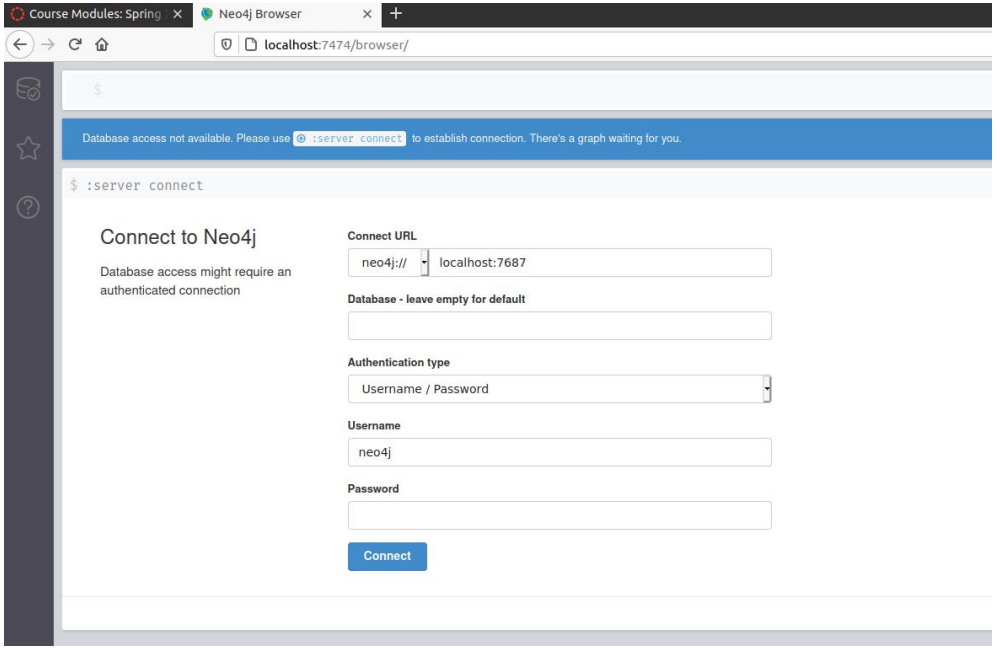
MongoDB Example With Python

- Select collections
  - ***collection.find_one()***
  - ***collection.find()***
  - ***Parameters:***
    - ***P1(Optional): query object***
    - ***P2(Optional): object describing which fields to include in the result***
  - When no parameters given, ***find()*** similar to **SELECT *** in Oracle DB**.**

- Example:
  - Find phone number 575-555-444 belongs to which department
    - *result = departmentCol.find_one({"phone#":"575-555-444"}, {"_id":0, "name":1})*
    - *‘result’ is a dictionary: {'name': 'Math'}*
  - More examples in the mongoDB.py

Neo4j Installations

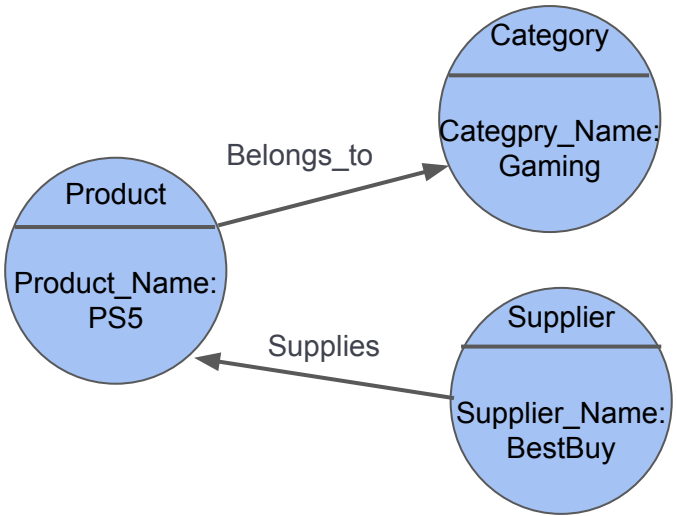
- Canvas→ Modules→ Neo4j, download the following files:
  neo4j-community-4.2.3-unix.tar.gz
- Extract the tar archive
  $ tar -xf /path/to/file
- Install java
  `$ sudo apt install default-jre`
- Start neo4j server
  `$ <NEO4J_HOME>/bin/neo4j start`
- Go to http://localhost:7474
- Default password ‘neo4j’



Neo4j

- Graph databases
- Consider the example with following tables: “Product”, “Categories” and “Supplier”.

- **Node:** Each row of the each table.
- **Edges:** Relationship between each node.



Product

Product_ID	Product_Name	Category_ID	Supplier_ID
1	PS5	1	1

Categories

Category_ID	Category_Name
1	Gaming

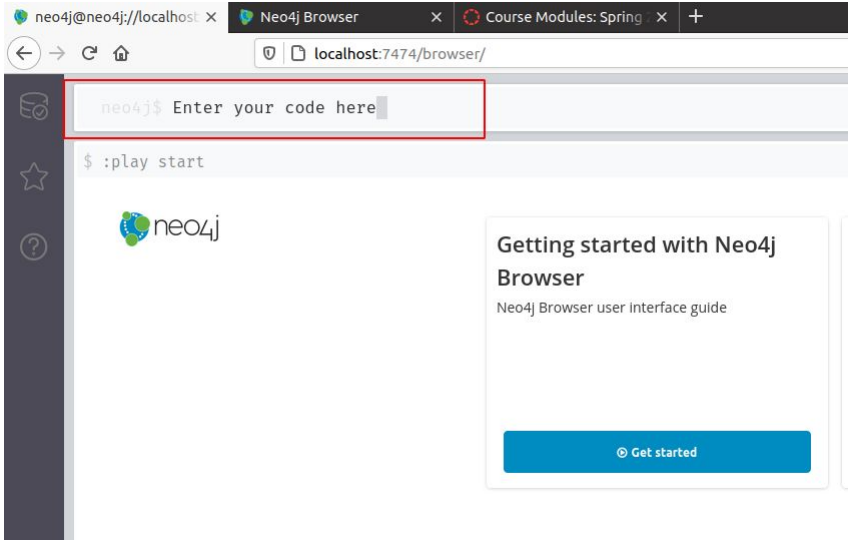
Supplier

Supplier_ID	Supplier_Name
1	BestBuy

Neo4j Example

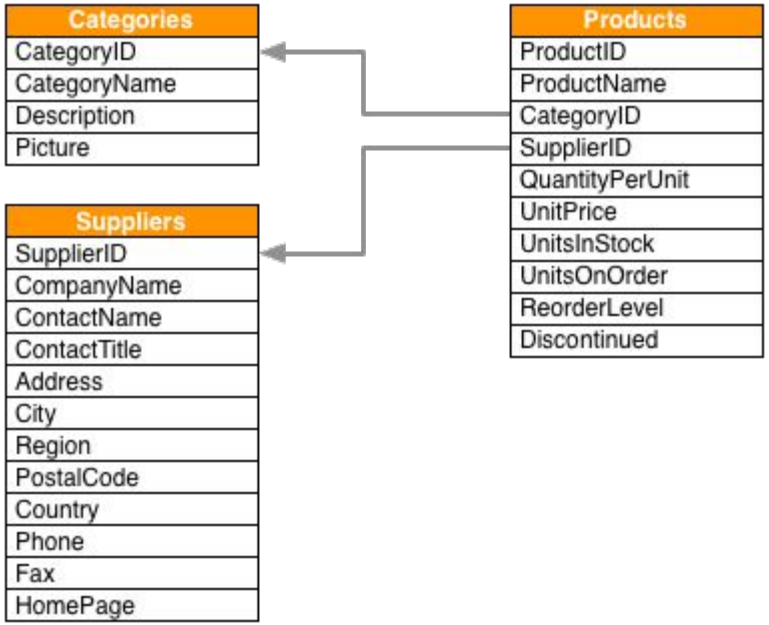
- Import CSV file
  - Download *products.csv*, *categories.csv*,*suppliers.csv* from Canvas→ Modules→ Neo4j
  - Copy all csv file into import folder: *<NEO4J_HOME>/import/*
  - Open Neo4j web interface http://localhost:7474, connect to default database with username 'neo4j' and default password 'neo4j'.
  - Load CSV

```
LOAD CSV WITH HEADERS FROM "file:///products.csv" AS
row
CREATE (n:Product)
SET n = row

LOAD CSV WITH HEADERS FROM "file:///categories.csv"
AS row
CREATE (c:Category)
SET c = row

LOAD CSV WITH HEADERS FROM "file:///suppliers.csv" AS
row
CREATE (s:Supplier)
SET s = row
```



Neo4j Example

- Display nodes
  MATCH (p:Product) return p LIMIT 5
- Create "*Part_of*" Relationship
  MATCH (p:Product),(c:Category)
  WHERE p.categoryID = c.categoryID
  CREATE (p)-[:Part_of]->(c)
- Create "*Supplies*" Relationship
  MATCH (p:Product),(s:Supplier)
  WHERE p.supplierID = s.supplierID
  CREATE (s)-[:Supplies]->(p)



Neo4j Example

- Visualize nodes with relationship
  Nodes with Part_of relationship: `MATCH p=()-[r:Part_of]->() RETURN p LIMIT 100`
  Nodes with Supplies relationship: `MATCH p=()-[r:Supplies]->() RETURN p LIMIT 100`
  Overall graph Visualization: `MATCH p=()-->() RETURN p LIMIT 200`

- Query to find the supplier ID of product Chocolade
  `MATCH (choc:Product {productName:'Chocolade'}) RETURN choc.supplierID`

- Visualize the “Confections” node cluster. Including all the product belong to the category confections, and suppliers of those products.
  ```
  MATCH (c:Category {categoryName:'Confections'})<-[:Part_of]-(p:Product),
  (s:Supplier)-[:Supplies]->(p)
  return c,p,s
  ```

Question ?

Office Hour: Friday 2-3pm
Email: anhao.xiang@student.nmt.edu