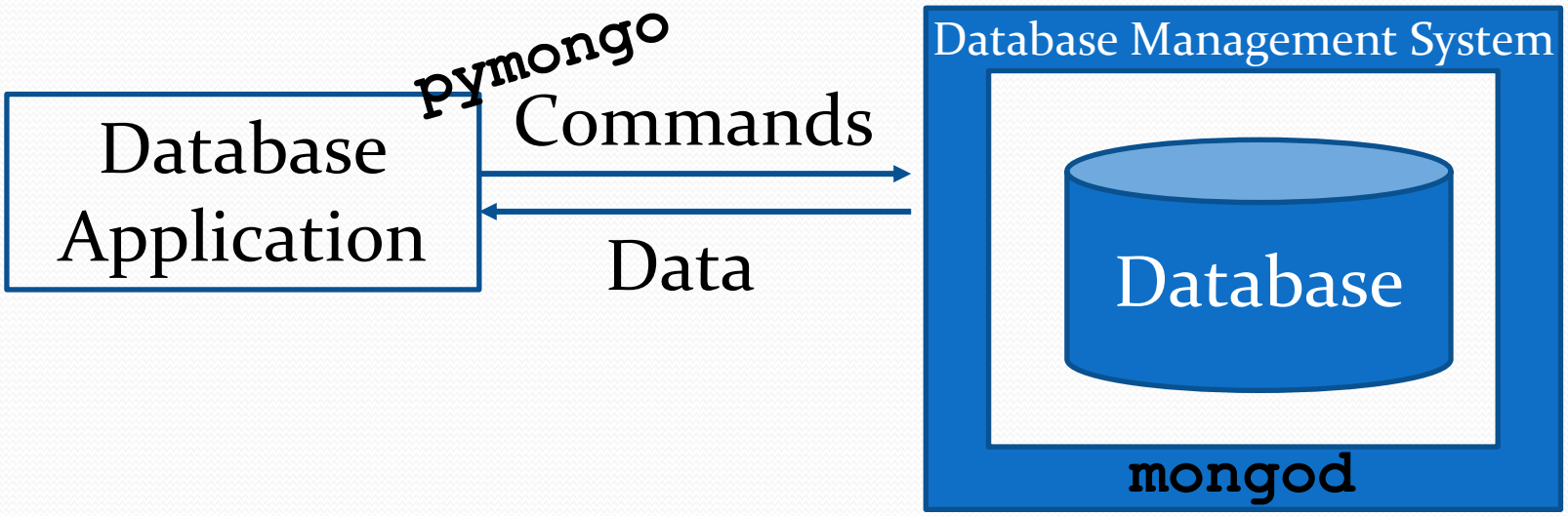
NoSQL

Introduction

What is NoSQL

- Databases can be divided in 2 types:
  1. RDBMS (Relational Database Management System)
  2. NoSQL
     - key-value databases
     - document databases
     - wide-column databases
     - graph databases
- NoSQL Database is also to refer a non-SQL or non relational database
  - CouchDB, MongoDB etc.

Database, Database Application and DBMS



Relational vs NoSQL

- Fixed Schema
  - Structure must be define before data can be inserted.
- Normalised data, prevent I/U/D anomalies and reduce redundancy
- Complex queries using JOINs

- Dynamic Schema
  - Structure is not fixed
- Unnormalized data, may introduce I/U/D anomalies and redundancy
- "Simple" queries using filters

*A schema defines the structure of data in a database and includes its

Organisation

Data types

Constraints and Relationships

Relational vs NoSQL

- Scaled Vertically
  - grow by increasing capacity of the DBMS/Server
  - Single Point of Failure
- Scaled Horizontally
  - grow by adding more servers
  - No Single Point of Failure

Relational vs NoSQL

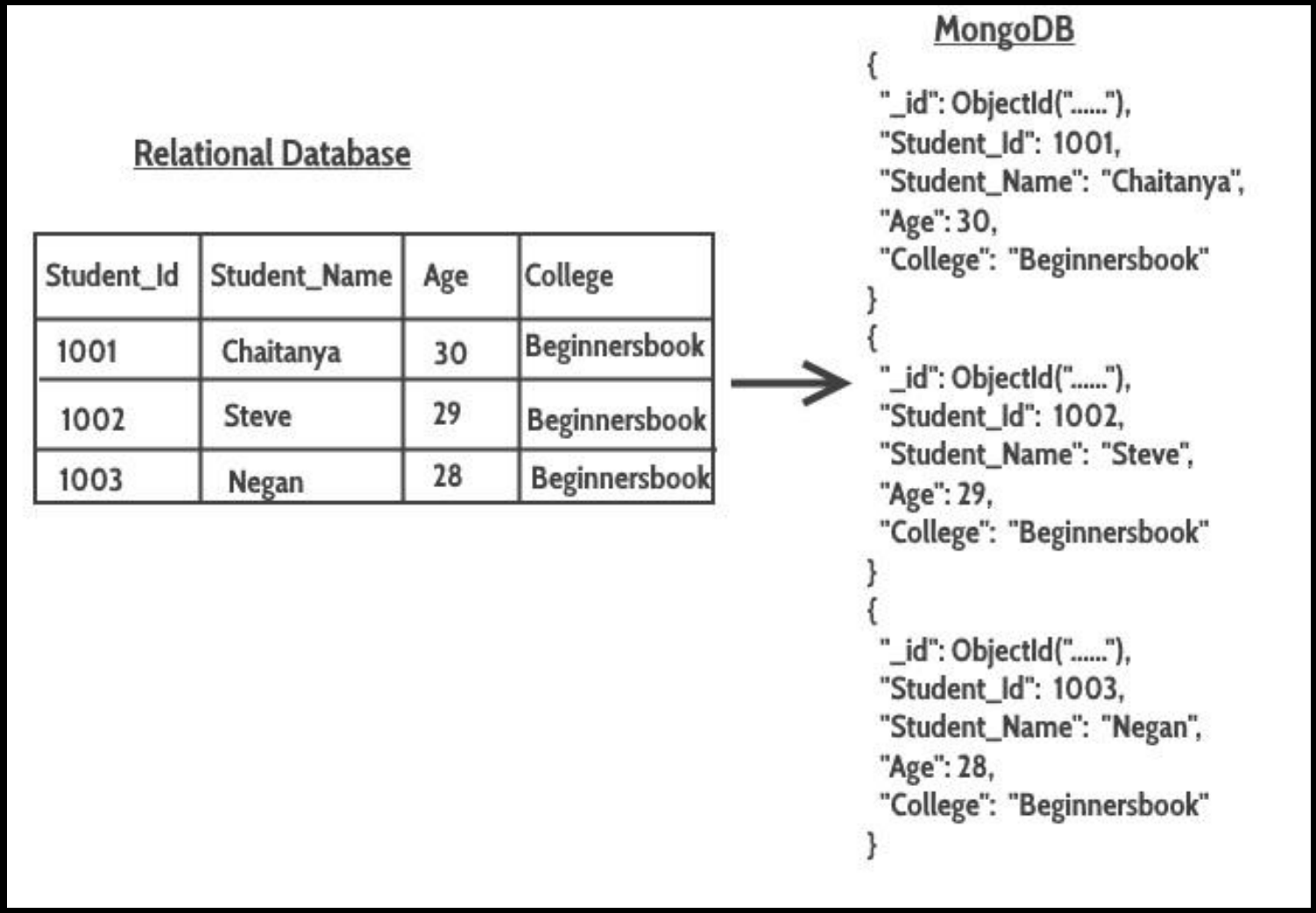
- Use Cases:
  - Schema is static and needs to known before hand.
  - Business model is mature and does not change
    - Order Processing and Flight Reservations.
    - Most Line of Busines Applications

- Use Cases:
  - Schema changes often
  - Unstructured data
    - Social Media Apps/ Personalised Content
    - IoT / Real Time Data
    - AI, Machine Learning
  - Data Archiving

Difference in Terminology

SQL Lite	**MongoDB**	**Python**
Database	Database	
Table	Collection	List
Row	Document	Dictionary object
Column	Field	Key
Joining	N.A	N.A

Example


Relational Database
Student_Id
Student_Name
Age
College
1001
Chaitanya
30
Beginnersbook
1002
Steve
29
Beginnersbook
1003
Negan
28
Beginnersbook
MongoDB
{
"_id": ObjectId("....."),
"Student_Id": 1001,
"Student_Name": "Chaitanya",
"Age": 30,
"College": "Beginnersbook"
}
{
"_id": ObjectId("....."),
"Student_Id": 1002,
"Student_Name": "Steve",
"Age": 29,
"College": "Beginnersbook"
}
{
"_id": ObjectId("....."),
"Student_Id": 1003,
"Student_Name": "Negan",
"Age": 28,
"College": "Beginnersbook"
}

Un-Normalised Form(UNF) to Relational Model

OrderID	OrderDate	CustomerName	CustomerContact	Products
0100	9/12/2019	Joe Wang	91223344	(Bag,12.00), (Book,19.50), (Camera,455.00)

Order

OrderID	OrderDate	CustomerID
0100	9/12/2019	012

Product

ProductID	ProductName	Price
09	Bag	12.00
19	Book	19.50
10	Camera	244.00

Order_Product

OrderID	ProductID
0100	09
0100	19
0100	10

Customer

CustomerID	CustomerName	CustomerContact
012	Joe Wang	91223344

Un-Normalised Form(UNF) to Relational Model

OrderID	OrderDate	CustomerName	CustomerContact	Products
0100	9/12/2019	Joe Wang	91223344	(Bag,12.00), (Book,19.50), (Camera,455.00)


```
[ {
  “OrderID”: “0100”,
  “OrderDate”:”9/12/2019”,
  “CustomerName”: “Joe Wang”,
  “CustomerContact”:”91223344”,
  “Products”: [
                {
                        “ProductName”:”Bag”,
                        “Price”:12.00
                },
                {
                        “ProductName”:”Book”,
                         “Price”:19.50
                },
                {
                        “ProductName”:”Camera”,
                        “Price”:455.00
                 }
          ]
 }]
```

Demo/Code

The code shown in the following slides are javascript that are executed in the mongo shell.

A Level software does not include mongo shell, so NO NEED to learn.

We have to use pymongo in Python.

Database Operations

on Javascript shell

- Show Databases/Collections
  - `show dbs`
  - `show collections`
- Connect / Create Database (if database_name not found
  - `use <database_name>`
- Create a collection/document
  - `db.createCollection("person")`

  OR by inserting a document, the collection will be automatically created!
  - `db.person.insert({`

  `“name”:"John",”class”:"18S01",`
  `“hobbies”:["running","kayaking","gaming"],`
  `“PM”:{“name”:”Chan”,”age”:34}  })`

Database Operations on Javascript shell

Query

- db.person.find(<query>, <projection>)
- db.person.find()
  #SELECT * FROM person

- db.person.find({"name":"John"})

#SELECT * FROM person WHERE name= “John”

#implicit AND

- db.person.find({"name": "John", "class":"18S01"})

#SELECT * FROM person WHERE name= “John” AND class = "18S01"

#explicit OR

db.person.find({"$or": [{ "name":"John"}, {"class":"18S02" }] })

```
{
“name”:"John","class":"18S01",
“hobbies”:["running","kayaking","gaming"],
“PM”:{“name”:”Chan”,"age":34}
}
```

Database Operations on javascript shell

Query on non-atomic value attribute

- db.person.find({"hobbies": "kayaking})

- # find person who has kayaking or running as a hobby
- db.person.find({"hobbies": {"$in":["running", "kayaking"]} })

nested attribute

- db.person.find({“PM.name”: “Chan”})

```
{
“name”:"John","class":"18S01",
“hobbies”:["running","kayaking","gaming"],
“PM”:{“name”:”Chan”,"age":34}
}
```

Database Operations
on Javascript shell

- query : find all PMs above 30 years old
  - db.person.find({“PM.age”: { "$gt":30 } })
    # WHERE PM.age > 30

find person with no hobbies

- db.person.find({"hobbies": { "$exists":0 } })

Projection

- db.person.find({ }, {"name":1,"_id:0})
  SELECT name FROM person

```
{
“name”:"John",”class”:"18S01",
“hobbies”:["running","kayaking","GAMING"],
“PM”:{“name”:”Chan”,"age":34}
}
```

Database Operations
on Javascript shell

- delete ## in PyMongo
  - `db.person.drop() #DROP TABLE person`
  - `db.person.remove({}) #DELET FROM person`
  - `db.person.remove({“name”:"John"})`
    `# DELETE FROM person WHERE name=“John”`

Database Operations
on Javascript shell

- update
  - db.person.updateOne({“name”:"John"}, {$set:{“name”:"NewJohn"} })
    # UPDATE person SET name =“NewJohn” WHERE name=“John”

- update PM's Chan age to 35
  - db.person.updateMany({“PM.name”:"Chan"}, {"$set":{"PM.age":35} })

```
{
“name”:"John",”class”:"18S01",
“hobbies”:["running","kayaking","GAMING"],
“PM”:{“name”:”Chan”,"age":34}
}
```

Database Operations

on Javascript shell

- update on an array element
  - Change "GAMING" in hobbies array to "gaming"
  - `db.person.updateMany({"hobbies":"GAMING"}, {"$set":{"hobbies":"gaming"} } )`
  - `db.person.updateMany({"hobbies":"GAMING"}, {"$set":{"hobbies.$":"gaming"} } )`
- remove a field in a document
  - Remove the John's PM attribute from the document
  - `db.person.updateMany({“name”:"John"}, {"$unset":{“PM”:""}} )`

```
{
“name”:"John",”class”:"18S01",
“hobbies”:["running","kayaking","GAMING"],
“PM”:{“name”:”Chan”,"age":34}
}
```

PyMongo

- see `pymongo.ipynb`