NoSQL with MongoDB

Objectives

- Compare and contrast SQL and NoSQL
    - Explain use cases for NoSQL
    - Explain use cases for SQL
    - Discuss why we need both options
- Develop basic familiarity with MongoDB

SQL Review

- What is SQL?
    - Structured Query Language
- SQL allows us to interact with Relational Database Management Systems (RDBMS)
    - Model **relations** in the data
        - Stores data about one object across multiple tables
            - {student_id, course_id}, {student_id, student_name} -> join on student_id and filter on course_id for names of all students in a course
        - Query data and relations efficiently
    - Maintain data consistency and integrity

SQL Review

- Tables, Columns (fields), Rows (records)
    - Each column is of a certain data type
    - Each row is an entry in the table
        - It holds values for each one of the columns
- Tables are specified by a **schema** that defines the structure of the data
    - We specify the schema ahead of time

NoSQL

- NoSQL
  - “Non SQL”, “Non relational”, “Not Only SQL”
  - *“Not Only SQL”* because stacks often use both NoSQL and SQL for different purposes
- Many NoSQL databases (though not all) are **document-oriented**
  - Each object (row/document) is stored in one place
  - Each object can be completely different from all others
- There is **no schema** (actually, there is, but it’s *implicit*)
  - Each document can have or not have whatever fields are appropriate for that particular document

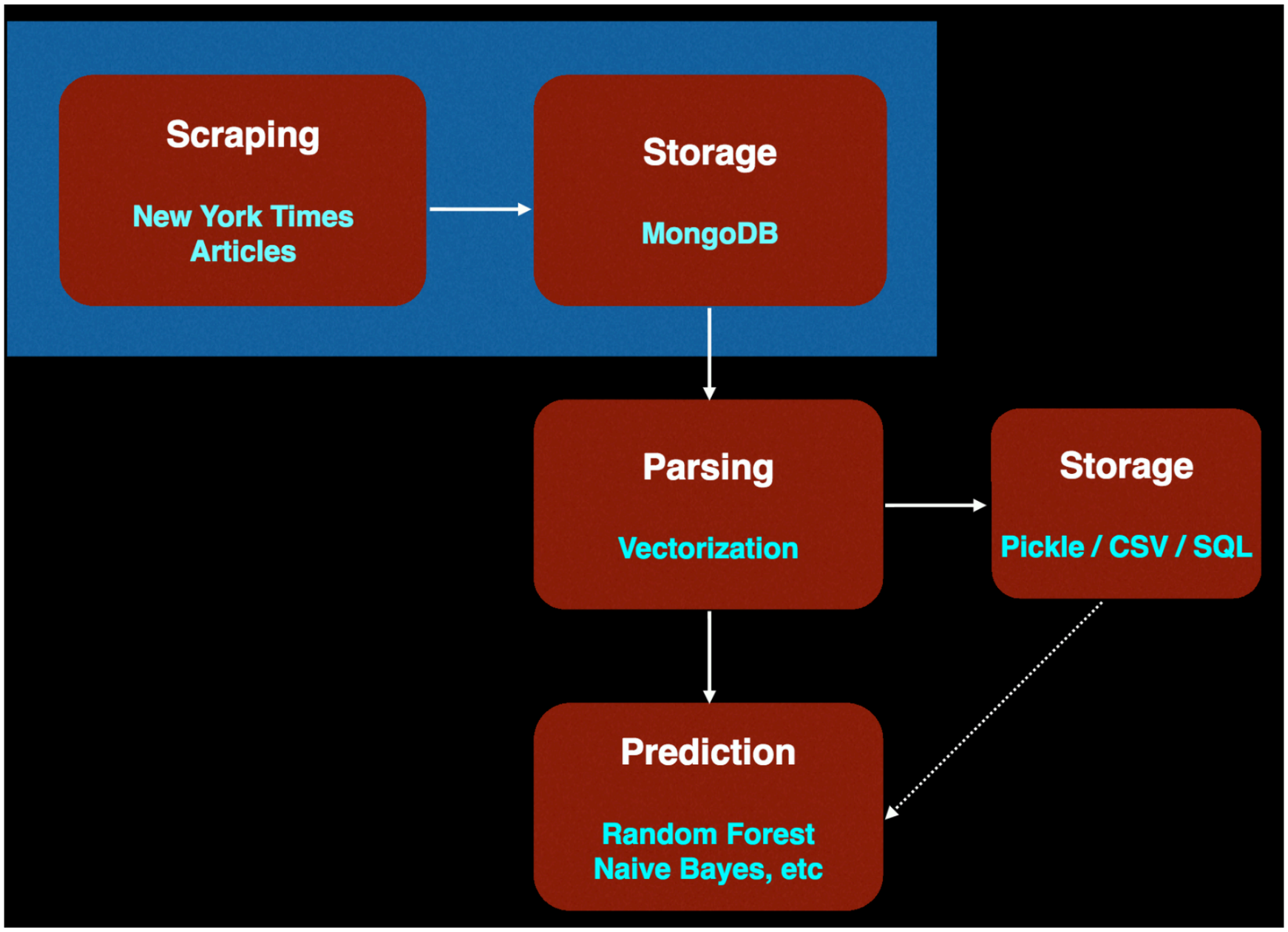
MongoDB

- MongoDB is a document-oriented database, an alternative to RDBMS
- Used for storing semi-structured data
- JSON-like objects form the data model, rather than RDBMS tables
- No schema, No joins, No transactions
- Sub-optimal for complicated queries
- MongoDB is made up of databases which contain collections (analogous to tables)
- A collection is made up of documents (analogous to rows or records)
- Each document is made up of key-value pairs (analogous to columns)
- ***RDBMS defines columns at the table level, document oriented database defines its fields at a document level.***
- CURSOR: When you ask MongoDB for data, it returns a pointer to the result set called a cursor.
- Actual execution is delayed until necessary.

Roughly Analogous:

- NoSQL
  - MongoDB
    - Database
      - Collection
        - Field
          - Document
- SQL
  - PostgreSQL
    - Database
      - Table
        - Column
          - Row
- Spreadsheet
  - Excel
    - File
      - Sheet
        - Column
          - Row

Typical Pipeline



<live coding>

connecting to MongoDB running locally via Pymongo
saving documents
finding documents
updating documents
counting documents

<live coding>
Class Exercise: self description

- Install pymongo with: pip/conda install pymongo
- Get mongodb_uri from Slack message now to connect to remote shared database

```
import pymongo

mongodb_uri = 'your_mongodb_uri_here (or set to None for localhost)'
mc = pymongo.MongoClient(mongodb_uri)  # Create MongoClient instance

db = mc[pymongo.uri_parser.parse_uri(mongodb_uri)['database']]  # Specify the
database to work with

collection_name = 'mongodb_lecture'  # Specify the collection name to work with

coll = db[collection_name]  # Get a collection instance

# Now we have a collection instance, ready to play with...
```

- Connect in Python: Client->Database->Collection
- Save into the collection a document describing yourself with whatever fields are appropriate:
    - coll.save({‘name’: ‘Dan’, ‘quest’: ‘Teach Data Science’, ‘favorite_color’: ‘Purple’})

<live coding>
Class Exercise: news articles

- Use the same starter code, change collection_name to: ‘class_news_articles’

- Each student: find a news article about technology or healthcare

- Save into the collection a document describing the news article with fields for ‘base_url’, and a count of each word:

  - coll.save({‘base_url’: ‘http://www.nyt.com/….’, ‘Elon’: 4, ‘Musk’: 5, ‘Tesla’: 2, ‘power’: 12, ‘solar’: 6})

  - Note: count the words in Python and create a dictionary object to save, then save that dictionary object

Pros and Cons of NoSQL

- Pros:
- No Schema!
  - Quick to iterate on data shape
  - No need to commit to a schema before starting
- No joins!
- Simple queries tend to be quick
- Culture stereotype of using NoSQL: Sip your hipster latte while working on your MacBook in a cafe and checking your Insta, your latest app is about to go viral (jk!)

- Cons:
- No Schema!
  - Well… 'implicit schema'
  - How will you communicate with other devs about choices you made?
- No joins!
- Complex queries tend to be slower
- Typically not as much ACID (Atomicity, Consistency, Isolation, Durability)
- Typically takes more storage
- Culture stereotype of avoiding NoSQL: Sip free office sludge coffee while working on your Dell in a cubicle and checking your 401k, your rock-solid system is ready for the upcoming audit (jk!)