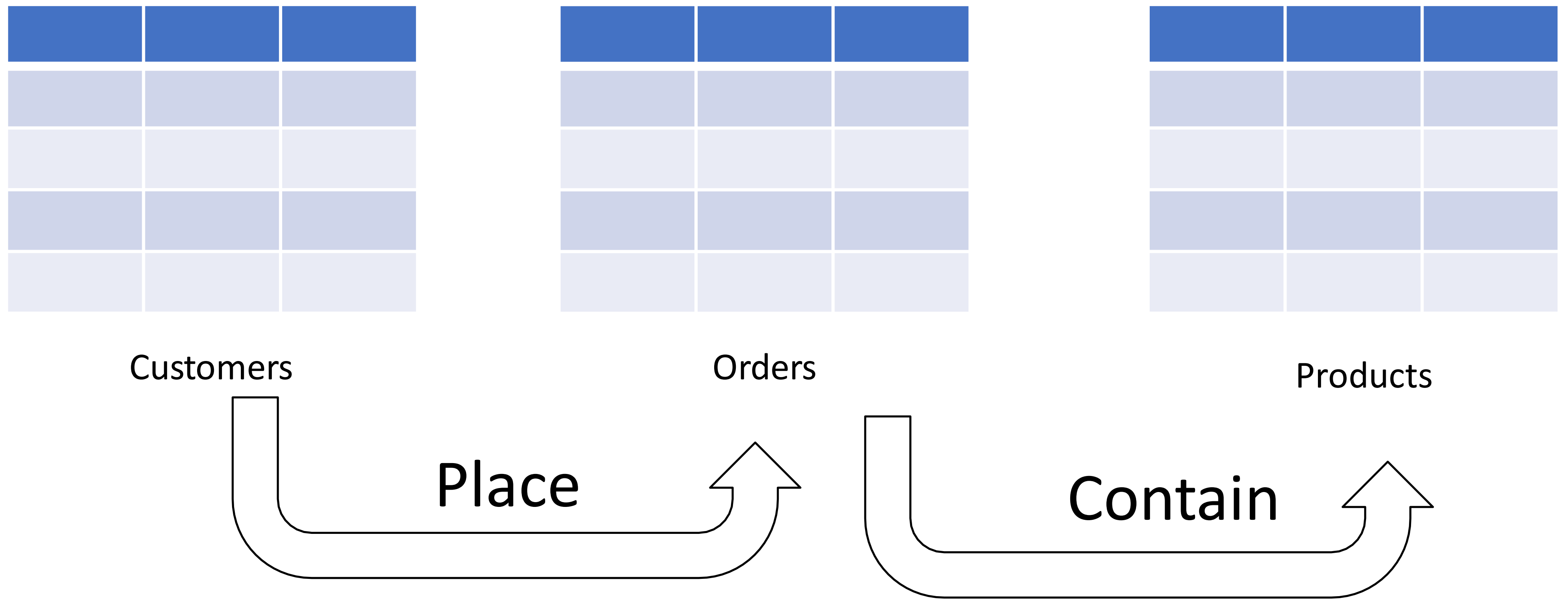
Introduction to NoSQL Databases

CS306 - Recitation 9

Relational Databases

- MySQL, MariaDB, PostgreSQL, SQLite
- Good at keeping data consistency
- Has to maintain these relationships which is an intensive process requiring high memory and compute power



Relational Databases

- Hard to scale
- Resource Intensive
- Relational databases can scale vertically but not horizontally
- NoSQL databases can scale both vertically and horizontally

Why does NoSQL scale well?

- First, they do away with costly relationships
- Every item on NoSQL stands on its own
- They are essentially **Key-Value** stores

Key	Value
134567632335	Macbook Pro
845737529208	Samsung Galaxy S4
576945764948	Hand Sanitizer
487539539754	Blanket

Products

NoSQL Databases

- Values can be JSON documents

Key	Value
134567632335	Macbook Pro
845737529208	Samsung Galaxy S4
398749234729	**{ name: "Playstation 5", price: "$499", description: "game console" }**
576945764948	Hand Sanitizer
487539539754	Blanket

Products

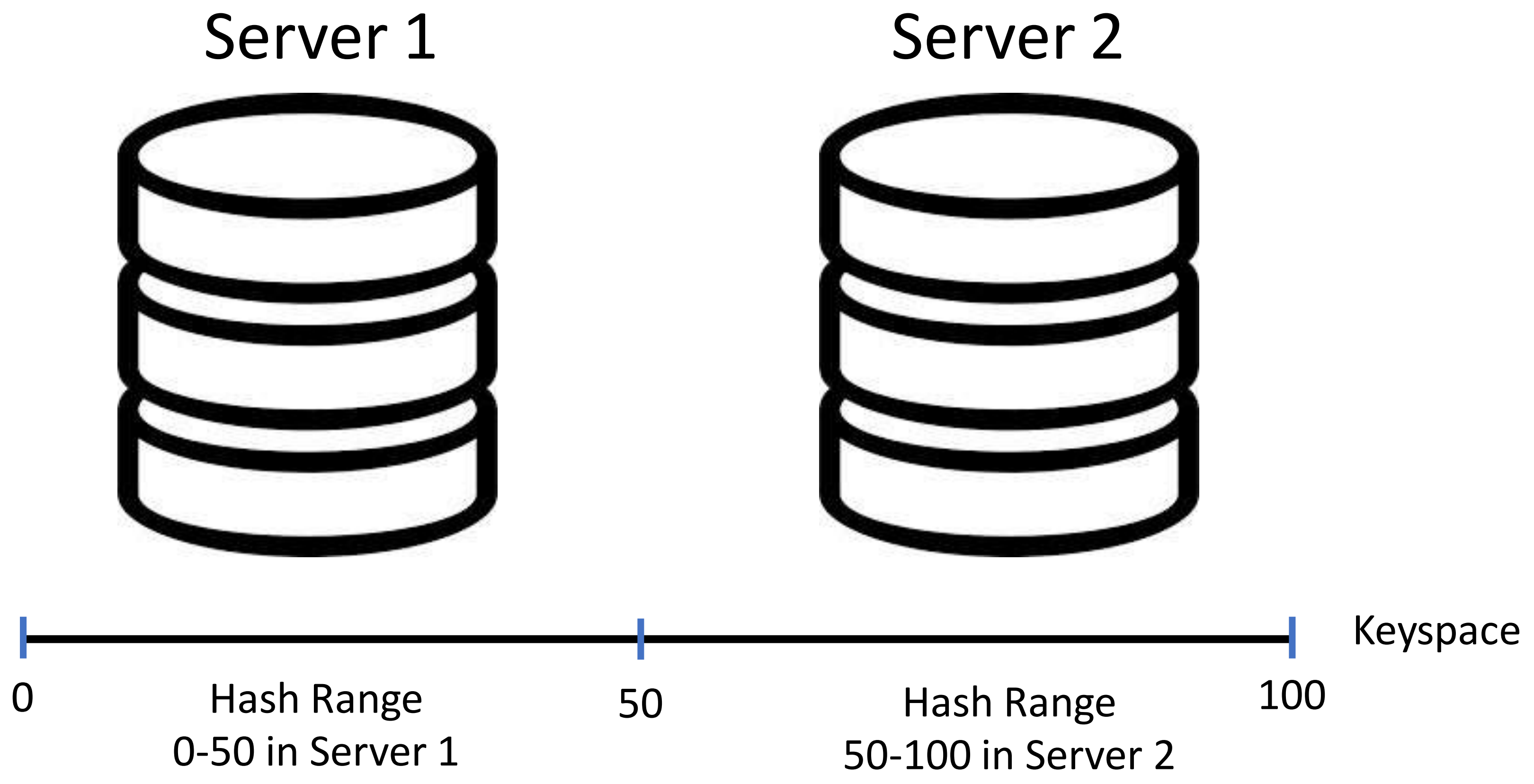
Partitions

- If our database is splitted across multiple partitions, how do we know where an item is stored?
  - Behind the scenes primary key is mapped into a hash that determines the server
  - Let’s say if hash is between 0-100 we put into a single server

Key	Hash	Value
134567632335	23	Macbook Pro
845737529208	87	Samsung Galaxy S4
576945764948	23	Hand Sanitizer
487539539754	100	Blanket

Partitions

- If that server becomes overloaded, it can be added a secondary server, which means the range will be splitted half



Schemaless

- Items in NoSQL does not have to be in the same structure.

```
{
  name: "Playstation 5",
  price: "$499",
  description: "game console"
}
```

```
{
  name: ”Macbook Pro",
  price_usd: "$999",
  price_eur:  "₺13766"
}
```

Products

- Can be useful if database is constantly evolving

Retrieving Data

- NoSQL data is only retrieved by their primary key

  Get Products with id “product-1905” -> **FAST**
  Get Orders With amount > $100 -> **SLOW**

NoSQL Databases

elasticsearch

mongoDB®

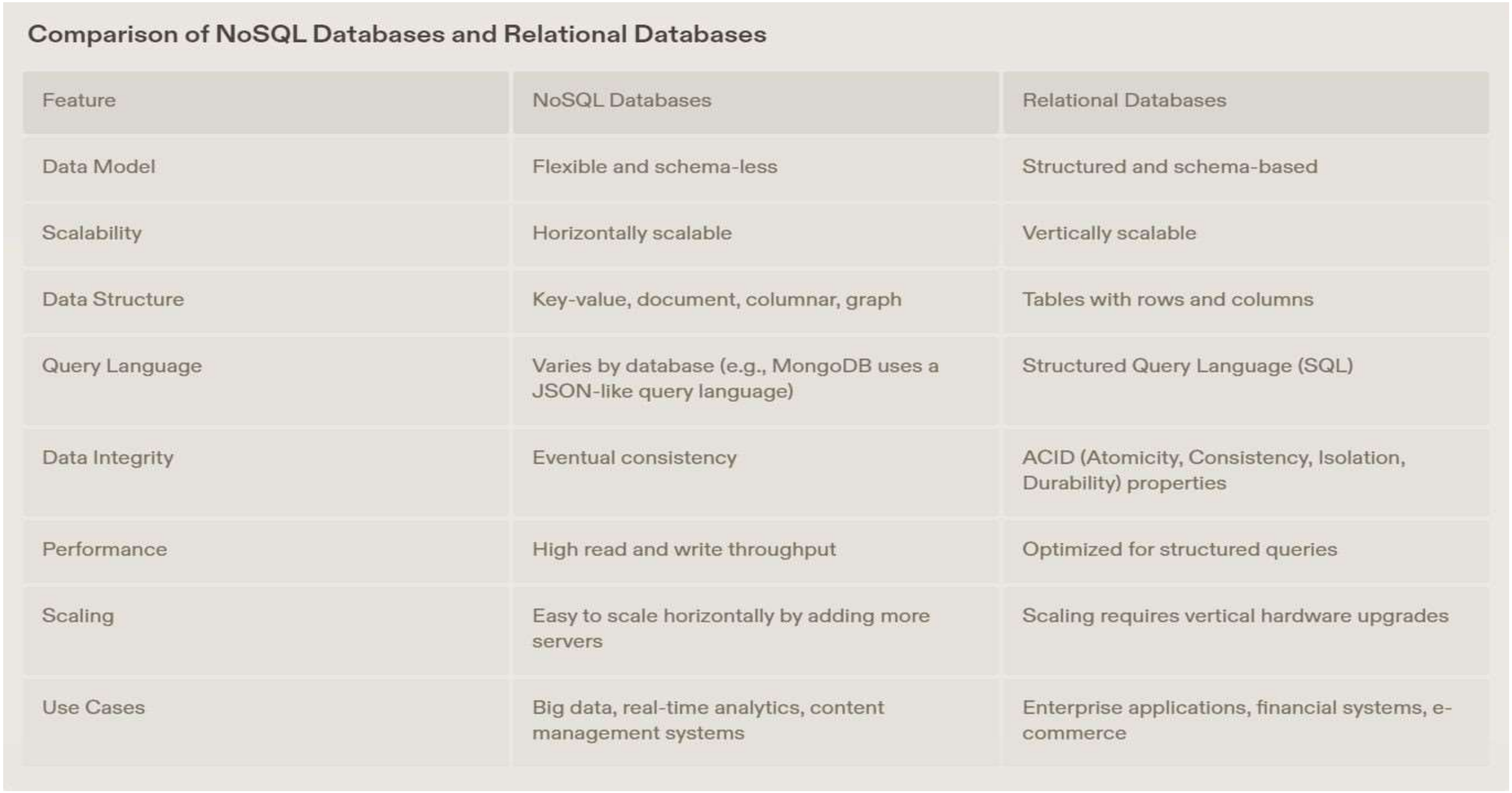
cassandra

amazon DynamoDB

redis

APACHE HBASE

What is a NoSQL



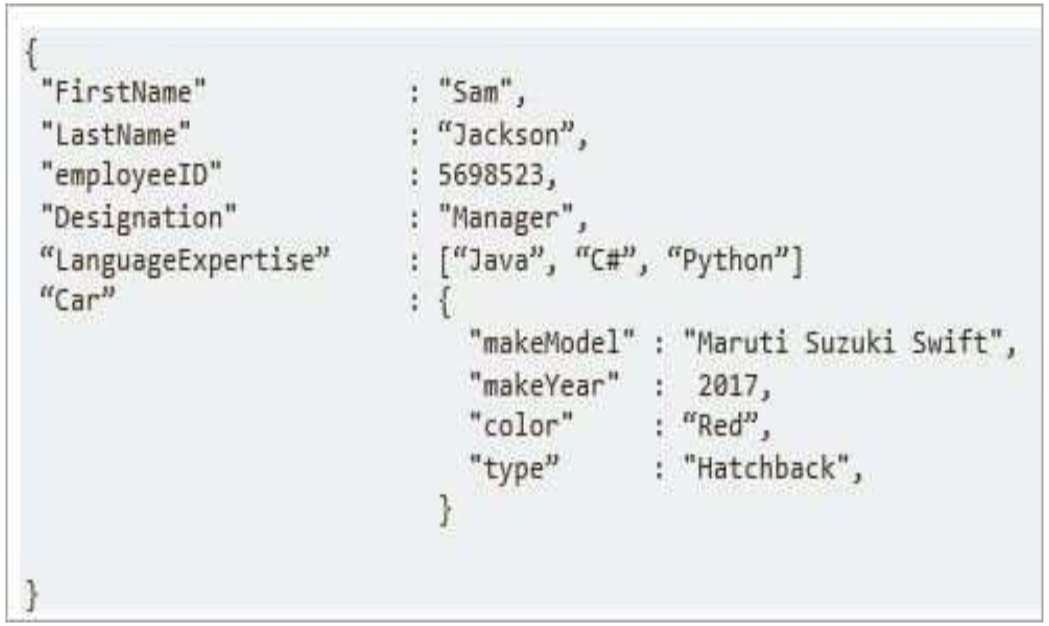
Comparison of NoSQL Databases and Relational Databases

Feature	NoSQL Databases	Relational Databases
Data Model	Flexible and schema-less	Structured and schema-based
Scalability	Horizontally scalable	Vertically scalable
Data Structure	Key-value, document, columnar, graph	Tables with rows and columns
Query Language	Varies by database (e.g., MongoDB uses a JSON-like query language)	Structured Query Language (SQL)
Data Integrity	Eventual consistency	ACID (Atomicity, Consistency, Isolation, Durability) properties
Performance	High read and write throughput	Optimized for structured queries
Scaling	Easy to scale horizontally by adding more servers	Scaling requires vertical hardware upgrades
Use Cases	Big data, real-time analytics, content management systems	Enterprise applications, financial systems, e-commerce

Advantages of NoSQL Databases

Key Advantages

- **Scalability**: NoSQL databases are highly scalable and can handle large amounts of data without sacrificing performance.
- **Flexibility**: NoSQL databases allow for flexible data models, making it easier to adapt to changing requirements and schema-less data.
- **High Performance**: NoSQL databases are designed for high-speed data retrieval and processing, making them ideal for applications that require real-time data access.
- **Horizontal Scaling**: NoSQL databases can be easily scaled horizontally by adding more servers to distribute the data load, ensuring high availability and fault tolerance.



```
{
 "FirstName"                : "Sam",
 "LastName"                 : “Jackson”,
 "employeeID"               : 5698523,
 "Designation"              : "Manager",
 “LanguageExpertise”        : [“Java”, “C#”, “Python”]
 “Car”                      : {
                                "makeModel" : "Maruti Suzuki Swift",
                                "makeYear"  :  2017,
                                "color"     : “Red”,
                                "type”      : "Hatchback",
                              }

}
```

Disadvantages of NoSQL Databases

Key Disadvantages

- **Lack of Standardization**: NoSQL databases lack a standardized query language, making it difficult to switch between different databases or integrate them into existing systems.
- **Limited Querying Capabilities**: NoSQL databases often have limited querying capabilities compared to SQL databases, making complex queries and joins more challenging.
- **Potential Data Inconsistency**: NoSQL databases prioritize scalability and performance over data consistency, which can lead to potential data inconsistencies in certain scenarios.
- **Complexity and Additional Development Effort**: NoSQL databases can be more complex to work with and may require additional development effort compared to traditional SQL databases.

Examples of NoSQL Databases: MongoDB

E-commerce

MongoDB is widely used in e-commerce applications for its ability to handle large amounts of product data and provide real-time inventory management.

Social Media

MongoDB is utilized by social media platforms to store and retrieve user-generated content, such as posts, comments, and user profiles, due to its flexible schema and scalability.

Should I Always Use NoSQL?

Of course no, instead:

- **Identify your data model:** Is it highly structured and relational, or is it more flexible and document-oriented?
- **Evaluate scalability needs:** Does horizontal scaling (NoSQL) or vertical scaling (SQL) make more sense for your application?
- **Don't be afraid to mix and match:** Sometimes, the optimal solution involves using a combination of SQL and NoSQL databases, depending on specific data needs.

NYPD
SQL FOR A SIMPLE KEY-VALUE STORE
NOSQL FOR A COMPLEX RELATIONAL DATASET

How to Connect to a MongoDB Database?

- Go to https://www.mongodb.com/cloud/atlas/register
- Create your account
- Create a database cluster as a shared cluster, which is free
- Set a user for your database from Database Access tab on the left panel
- Go back to your database section from left panel and click connect
- Select your development environment and copy the connection string
- You may also check out this video: https://www.youtube.com/watch?v=3wNvKybVyaI

MongoDB

- Installation

  Windows

  - Download MongoDB from https://www.mongodb.org/downloads
  - Go to location where mongo DB is installed "C:\Program Files\MongoDB\Server\3.4\bin" and open command prompt at that location (mongod)

  macOS

  - Open your terminal and type the followings:

```
brew tap mongodb/brew

brew install mongodb-community@5.0

brew services start mongodb-community@5.0

mongosh
```

MongoDB Commands

`db.help`

Get list of commands that you can execute

`show dbs`

Show all databases

`use db_name`

Create a database with a name of

`db_name use cs306`

`db`

Show your current database

`db.dropDatabase()`

Drops the selected database

`db.createCollection("collection_name")`

Create a new collection with name of

`collection_name db.createCollection("users")`

`show collections`

Get the list of collections

`db.collection_name.drop()`

Drop the selected collection

`db.users.drop()`

MongoDB Commands

INSERTION

db.collection_name.insert(document)

To insert single document in selected collection

Examples:

```
db.users.insert({name:'Emir Alaattin Yilmaz', city:'Istanbul'})
db.users.insert({name:'Yucel Saygin', city:'Ankara'})
```

Insert Multiple Documents:

```
db.users.insertMany([
{ name: 'Ragga Oktay', city: 'Rotterdam' },
{ name: 'Mansur Ark', city: 'Mersin' }
])
```

Get collection documents

db.collection_name.find()

```
db.users.find()
```

UPDATE

db.COLLECTION_NAME.update(SELECTION_CRITERIA, UPDATED_DATA)

```
db.users.update(
          {'name':'Emir Alaattin Yilmaz'},
          {$set:{'name':'Yahya Kemal'}}
          )
```

DELETION

```
db.collection_name.remove(DELLETION_CRITTERIA)
db.users.remove({'name':"Yucel Saygin"})
```