Lecture 11: Introduction to non-relational databases

Quan Nguyen, Ph.D.
Assistant Professor
Department of Computing Science

THOMPSON RIVERS UNIVERSITY

Oct 15, 2024

Learning goals

The goal of the 2nd half of this course is to equip students with a comprehensive understanding and practical skills in working with NoSQL databases, specifically **MongoDB**, and its integration with Python through PyMongo. Additionally, students will learn to leverage **PySpark** for large-scale data processing and analysis. Finally, students will learn about the **key principles in data governance, privacy, and security**.

data engineer in Canada
1,835 results

Set alert

Viewed · Promoted

Data Engineer
BCAA
Burnaby, BC (Hybrid)
19 company alumni work here
Viewed · Promoted

Senior Data Scientist/Economist
Deloitte
Toronto, ON
698 school alumni work here
Promoted

Data Engineer, Ads Data
TikTok
Vancouver, BC
17 company alumni work here
Viewed · Promoted

Data Engineer II - Temporary
Electronic Arts (EA)
Vancouver, BC (Hybrid)

Data Engineer
BCAA · Burnaby, BC (Hybrid)

Apply Save

demonstrated experience building data warehouses, scalable data pipelines, provisioning self-serve

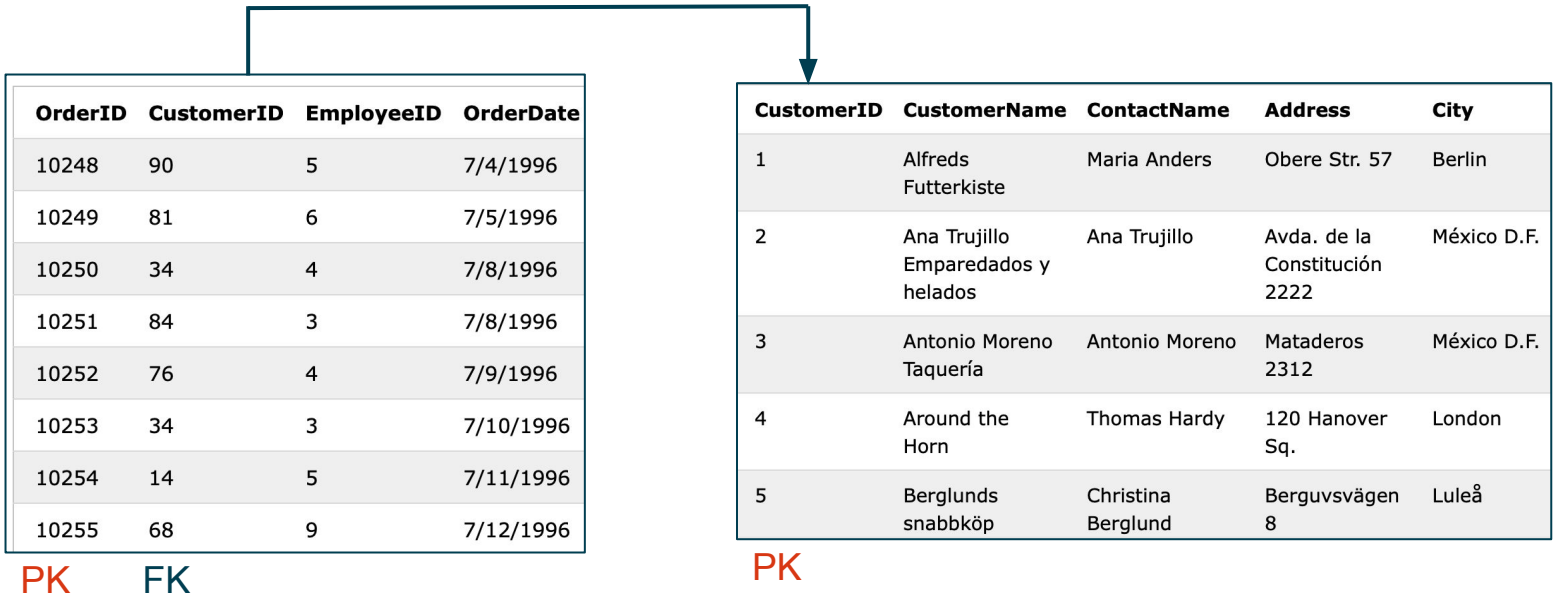
- You have experience leading data projects from problem identification to production.
- You have experience supporting ML and AI experiments, business intelligence, data science models and reporting.
- You are proficient in handling large and complex data sets in all stages from extraction to modelling, structuring, filtering, discovery, and development of visualizations.
- You have experience developing, optimizing, and implementing machine learning models in a production environment.
- You have a solid understanding of foundational statistical concepts.
- Ability to connect systems using API's, Python code or Bash scripts.
- Strong working knowledge of source and target data structures, ETL and CDC processes and products, analysis, troubleshooting, and code review.
- You have demonstrated experience with quantitative data analysis and utilizing data to drive decision-making, specifically related to measuring and demonstrating effectiveness.
- Working understanding of high-volume data processing and analysis, relational and NoSql Databases, and Machine Learning Tools/Frameworks.
- Capability to learn and develop new engineering techniques as required.
- Experience with the following tools/languages for data management and manipulation: Informatica, MapReduce, Rest APIs, Attunity Qlik-Replicate, Scala
- Proficient in:
  - Python, SQL, NoSQL
  - AWS stack (certification preferred)

Learning objectives

- Understand how relational and non-relational database differ in terms of data structure, schema, and query language.
- Explain the pros & cons of each type of database and its use case
- Explain different types of non-relational databases (i.e., column-wide, key-pair, graph, document)

Recap of relational database

- Relational databases are structured as a table, with rows and columns



OrderID	CustomerID	EmployeeID	OrderDate
10248	90	5	7/4/1996
10249	81	6	7/5/1996
10250	34	4	7/8/1996
10251	84	3	7/8/1996
10252	76	4	7/9/1996
10253	34	3	7/10/1996
10254	14	5	7/11/1996
10255	68	9	7/12/1996

PK FK

CustomerID	CustomerName	ContactName	Address	City
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå

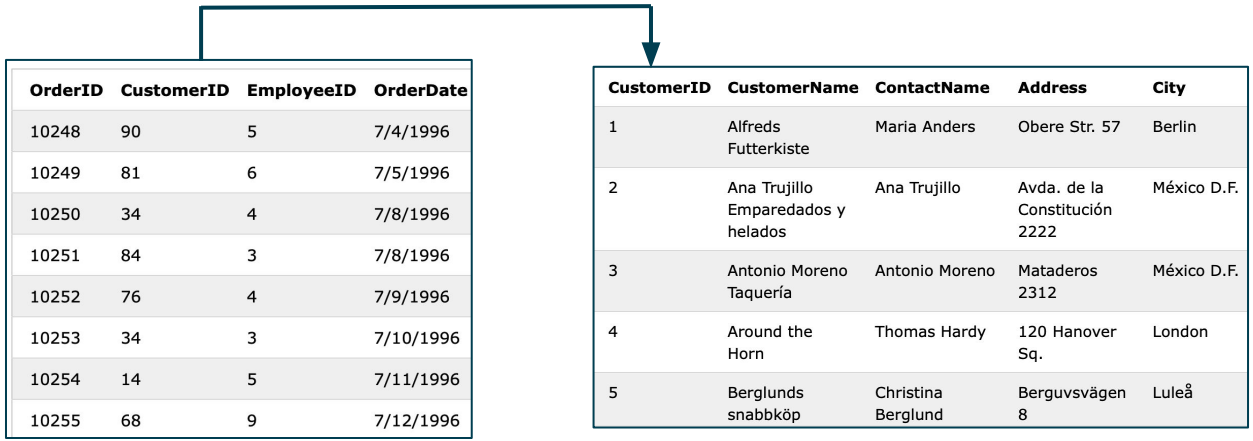
PK

Advantages of relational databases

- Relational/Normalized data → reduce duplication, improve consistency
- Pre-defined schema and data format
- ACID compliant transactions
- Standardized query language across different RDBMS

Use case: When data accuracy and transactional reliability is prioritized over flexibility and speed

Banking, finance, insurance



OrderID	CustomerID	EmployeeID	OrderDate
10248	90	5	7/4/1996
10249	81	6	7/5/1996
10250	34	4	7/8/1996
10251	84	3	7/8/1996
10252	76	4	7/9/1996
10253	34	3	7/10/1996
10254	14	5	7/11/1996
10255	68	9	7/12/1996

CustomerID	CustomerName	ContactName	Address	City
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå

Limitations of relational database

Relational databases are structured as a table, with rows and columns

_id	first_name	last_name	address	email	phone
1	Quan	Nguyen	Kamloops	lnguyen@tru.ca	778-123-456

What if we want to add a second email?

_id	first_name	last_name	address	email1	email2	phone
1	Quan	Nguyen	Kamloops	lnguyen@tru.ca	quan@ubc.ca	778-123-456

Limitations of relational database

Let’s add another record

_id	first_name	last_name	address	email1	email2	phone
1	Quan	Nguyen	Kamloops	lnguyen@tru.ca	quan@ubc.ca	778-123-456
2	Ajay	Dhruv	Kamloops	ajay@tru.ca		778-456-789
3						
4						
5						
6						

Limitations of relational database

Let’s add another column for second phone number for Ajay

_id	first_name	last_name	address	email1	email2	phone1	phone2
1	Quan	Nguyen	Kamloops	lnguyen@tru.ca	quan@ubc.ca	778-123-456	
2	Ajay	Dhruv	Kamloops	ajay@tru.ca		778-456-789	123-456-789
3							
4							
5							
6							

As you can see, the database gets bloated with empty fields

Limitations of relational database

SQL handle this problem by separating emails & phone into separate tables. These tables are related to each other by user_id

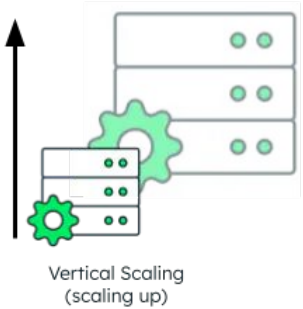
_id	first_name	last_name	address
1	Quan	Nguyen	Kamloops
2	Ajay	Dhruv	Kamloops
3			
4			

_id	user_id	email
1	1	q@ubc.ca
2	1	q@tru.ca
3	2	a@tru.ca
4	3	b@gmail.com

_id	user_id	phone
1	1	778-332-123
2	1	305-233-233
3	2	123-456-789
4	3	444-555-131

To update a record, we have to access multiple tables and join the data back together → inefficient

Limitations of relational database



Schemas must be created in advance! → long time to set up, not suited to handle unstructured data or format is unknown.

Vertical Scaling: Typically scaled by increasing the resources of a single server (e.g., more CPU, RAM).

You can read more about the pros/cons of vertical versus horizontal scaling here

Filling in the Gap

1970s

Relational Databases

SQL was developed by IBM as a way to interact with the new relational databases

1990s

World Wide Web

Need for data storage explodes

2000s

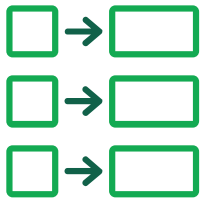
NoSQL

Unstructured data storage to mitigate costs and increase efficiency

Image Credit: MongoDB Educators' resources

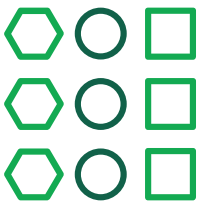
Non-Relational Database Types

- **Flexible schema** that accommodates unstructured data (JPGs, text, audio, video) without the need to pre-define the structure
- **High performance**: Low latency, optimized for specific task
- **Horizontal scaling** by adding more nodes (rather than scaling up a single server)

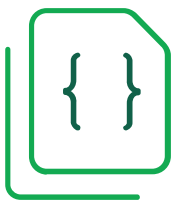


Key/Value

Graph



Column



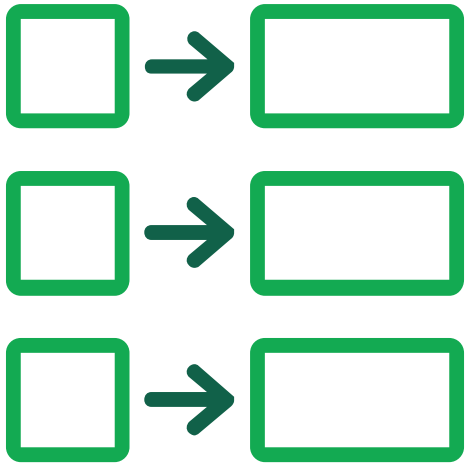
Document

Structure

- A unique key is paired with a collection of values, where the values can be anything from a string to a large binary object.

Strength

- Simple data model.



Key/Value Database

Key/Value: Example

Key	Value
Name	Sherlock Holmes
Age	40
Address	221B Baker Street

They are highly optimized for scenarios where quick lookups based on a key are required.

Example: Caching, shopping cart

Tools: Redis, Amazon DynamoDB, Riak

Structure

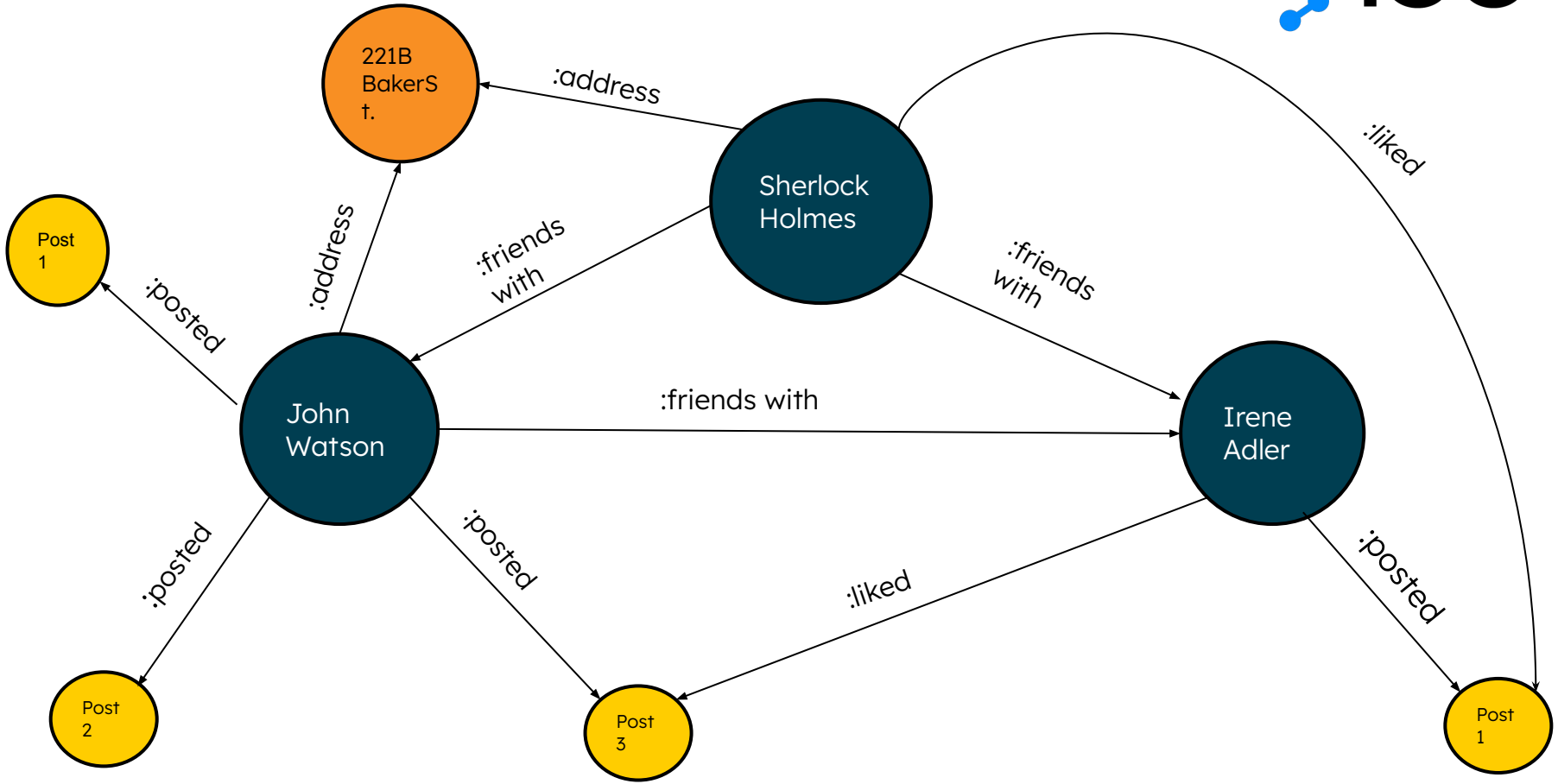
- Captures connected data.
- Each element is stored as a node.
- Connections between nodes are called links or relationships.

Strength

- Traverses the connections between data rapidly.

Graph Database

Graph: Example



Use case

- Social networks
- Recommendation systems

Tools

- Neo4j
- Amazon Neptune

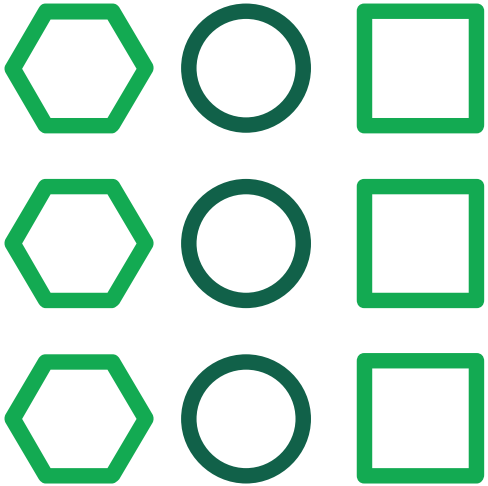
Graph Database

Structure

- Data is stored using key rows that can be associated with one or more dynamic columns

Strengths

- Highly performant queries
- Designed for analytics



Column Oriented or Wide Column

Column Oriented Example

Name	ID
Sherlock	001
John	002
Irene	003

Age	ID
40	001
45	002
43	003

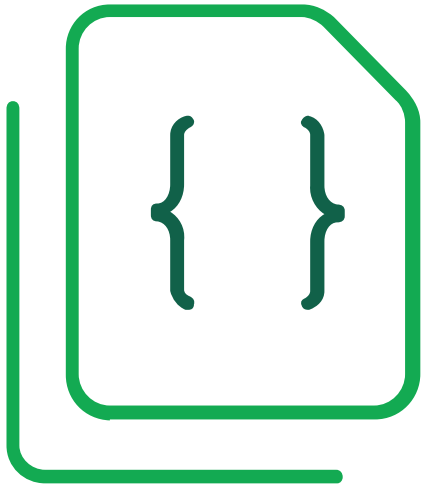
Height	ID
6’2	001
5’9	002
5’7	003

Structure

- Polymorphic data models
- Each document contains markup that identifies fields and values.

Strengths

- Obvious relationships using embedded arrays and documents
- No complex mapping



Document Database

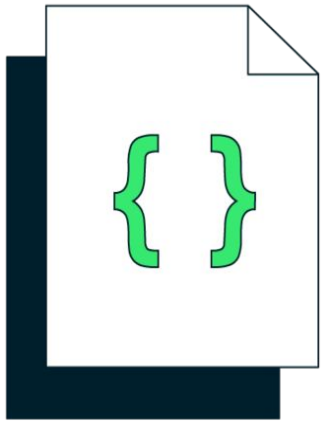
mongoDB®

Document Model Example

```
{
  "_id":
ObjectId("5ef2d4b45b7f11b6d7a"
),
  "user_id": "Sherlock
Holmes",
  "age": 40,
  "address":
     {
       “Country: “England”
       “City”: “London”,
       “Street”: “221B Baker
     St.”
     },
  “Hobbies”:[ violin,
crime-solving ]
}
```

```
{
  "_id":
ObjectId("6ef8d4b32c9f12b6d4a")
,
  "user_id": "John Watson",
  "age": 45,
  "address":
     {
       “Country: “England”
       “City”: “London”,
       “Street”: “221B Baker
     St.”
     },
  “Medical license”: “Active”
}
```

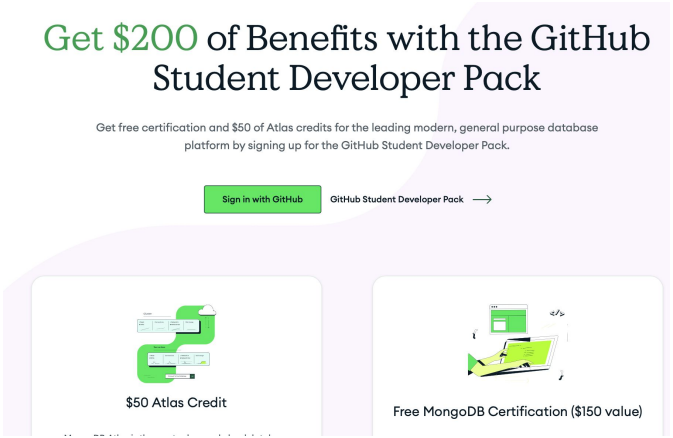
The Document Model



For general purpose use, the document model prevails as the preferred model by developers and database administrators.

Next steps

Register a free account with MongoDB Atlas using your Github account



https://www.mongodb.com/students