TOPIC: Neo4J

GRAPH DATABASE MANAGEMENT SYSTEM

GROUP MEMBERS

- JAVERIA AHMED
- DILKASH
- BATOOL EHSAN
- AQSA JAWAID
- ANEEQA ASIF
- HAFSA ABBASI
- BAKHTAWAR SIRAJ
- JAWERIA EJAZ

Contribution of Each Member

Presentation by:

Javeria Ahmed:(slide no 4-18)

- WHAT IS NeoJ4?, What is a Graph Database?
- Why Graph Databases?
- RDBMS vs Graph Database,
- Advantages, Features, Neo4j Customers

Dilkash:(slide no 19-37)

- USE CASES OF NEO4J(Social networks, fraud detection, recommendation engines and knowledge grpahs
- ARCHITECTURE OF NEO4J:
- Storage engine (native graph databases)
- Query engine (cypher language)
- API layer(cypher, bolt, java, rest)

Batool Ehsan:(slide no 38-55)

- How Neo4j works?
- How Information stored in Neo4j
- What is Cypher?
- Clauses in Cypher Query Language
- Write clauses, Read clauses
- String Functions
- Create Index, Create Constraint

Aqsa Jawaid:(slide no 56-67)

- Comparison of Neo4j with Other Databases ?
- Neo4j in Data science
- Future Scope of Neo4j in Data sciences

Contribution of Each Member

Project demonstration by:

Jaweria Ejaz:

- Different ways of using Aura db

Aneeqa Asif:

- Types of CQL queries
- How to create graph database project using CQL queries

Bakhtawar Siraj:

- How to download and create account
- How to create new instance on neo4j

Hafsa Abbasi:

- Types of CQL queries
- How to Import CSV file
- What is the working of different panels

WHAT IS NeoJ4?

Neo4j is the world's leading open source Graph Database which is developed using Java technology. It is highly scalable and schema free (NoSQL)

NO SQL

01

Not Only SQL

NoSQL databases (aka "not only SQL") are non-tabular databases and store data differently than relational tables.

02

Variety Of Types

NoSQL databases come in a variety of types based on their data model. The main types are document, key-value, wide-column, and graph

03

Flexible Schemas

They provide flexible schemas and scale easily with large amounts of data and high user loads.

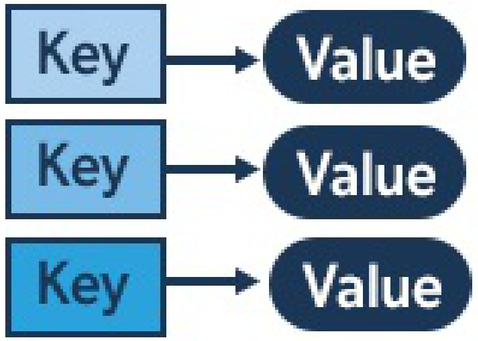
04

Used For

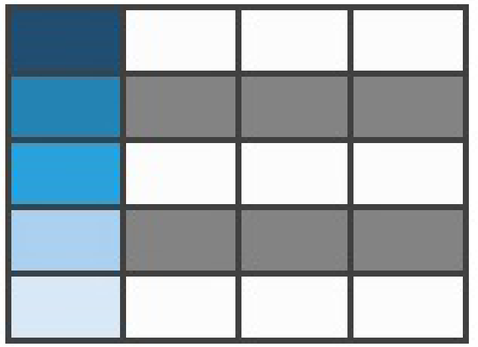
NoSQL is used for Big data and real-time web apps.

NoSQL

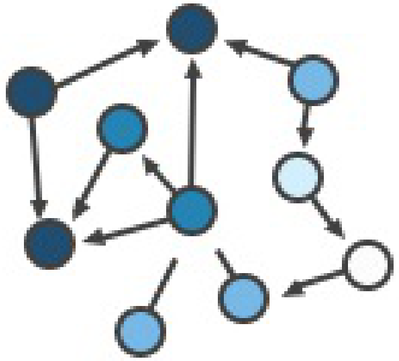
Key-Value



Column-Family



Graph



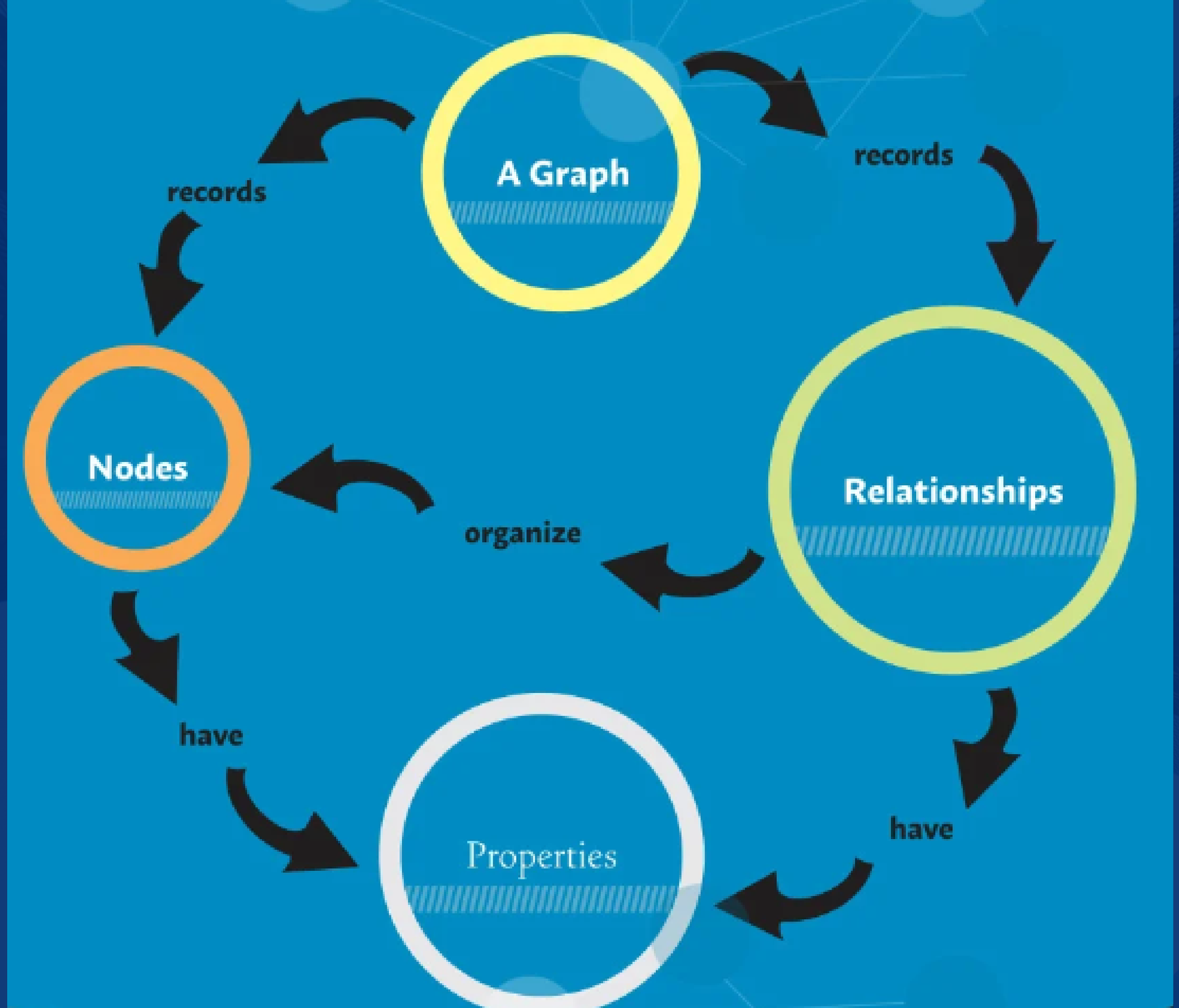
Document

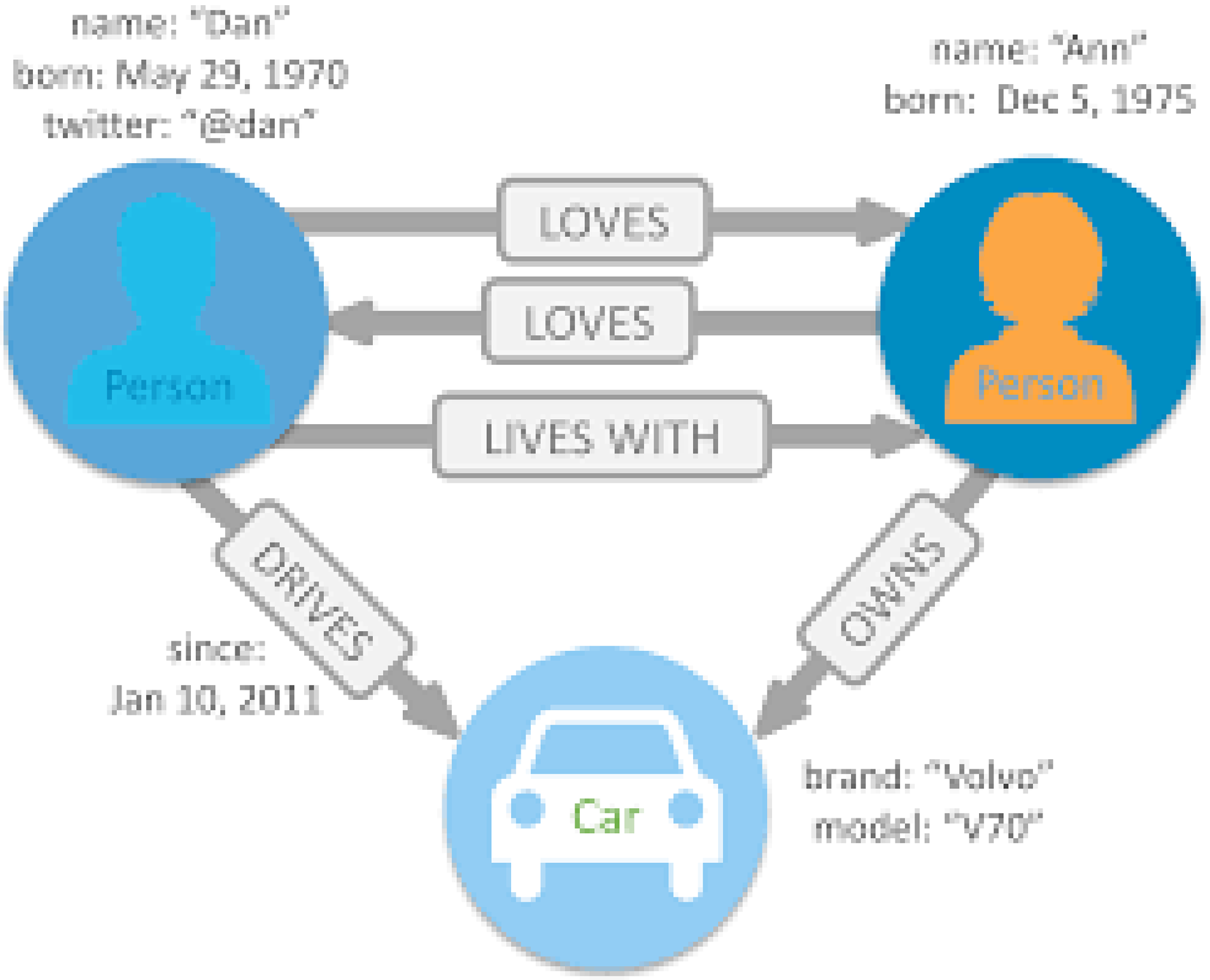
STUDIO SHODWE

What is a Graph Database?

A graph is a pictorial representation of a set of objects where some pairs of objects are connected by links. It is composed of two elements - nodes (vertices) and relationships (edges).

Graph database is a database used to model the data in the form of graph. In here, the nodes of a graph depict the entities while the relationships depict the association of these nodes.


A Graph
records
records
Nodes
Relationships
organize
have
have
Properties


name: "Dan"
born: May 29, 1970
twitter: "@dan"
name: "Ann"
born: Dec 5, 1975
LOVES
LOVES
LIVES WITH
Person
Person
DRIVES
since:
Jan 10, 2011
OWNS
Car
brand: "Volvo"
model: "V70"

WHY GRAPH DATABASES?

01

Nowadays, most of the data exists in the form of the relationship between different objects and more often, the relationship between the data is more valuable than the data itself.

02

Relational databases store highly structured data which have several records storing the same type of data so they can be used to store structured data and, they do not store the relationships between the data.

03

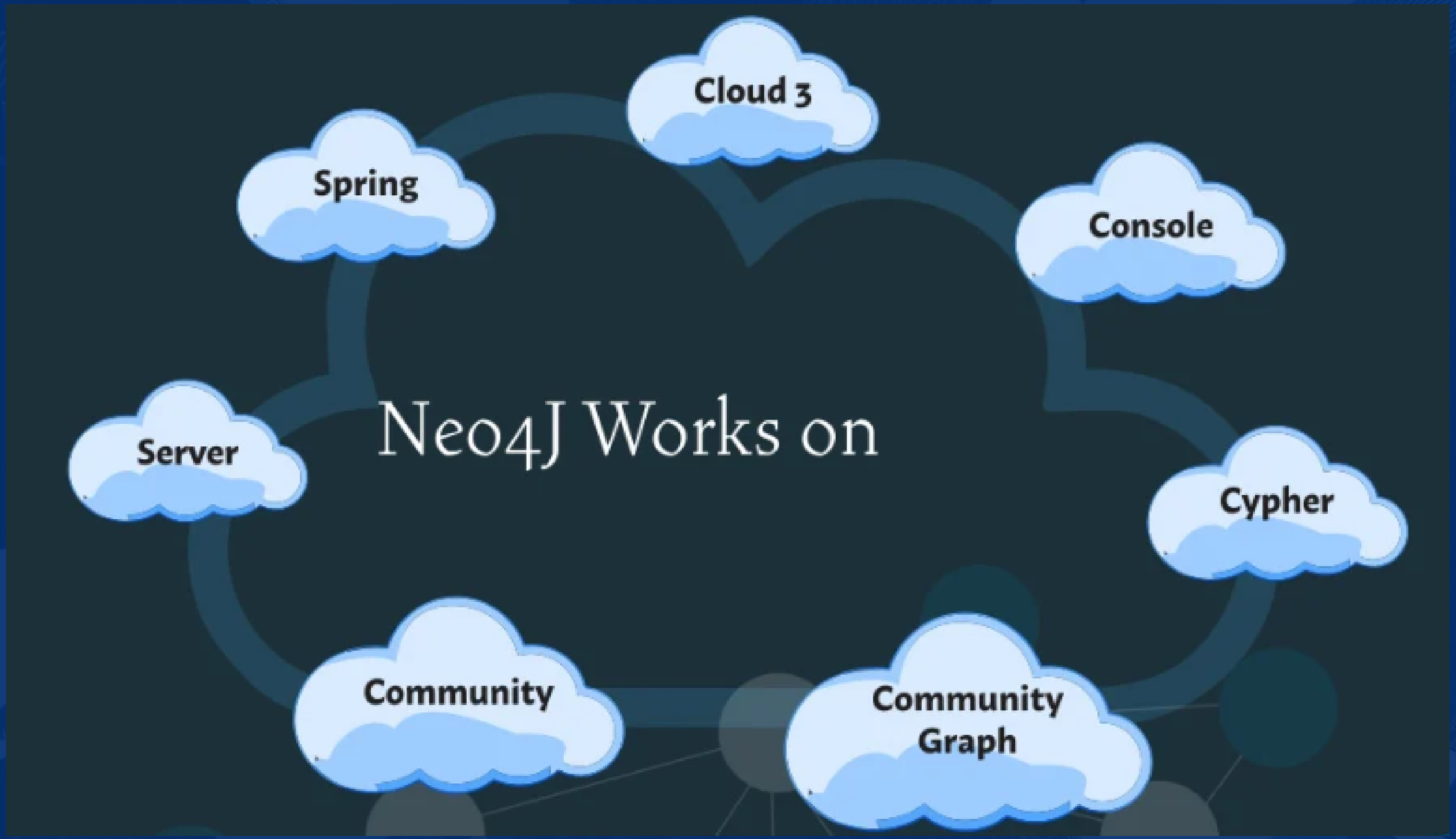
Unlike other databases, graph databases store relationships and connections as first-class entities.

04

The data model for graph databases is simpler compared to other databases and, they can be used with OLTP systems. They provide features like transactional integrity and operational availability.

RDBMS vs Graph Database

1	Tables	Graphs
2	Rows	Nodes
3	Columns and Data	Properties and its values
4	Constraints	Relationships
5	Joins	Traversal


Cloud 3
Spring
Console
Server
Neo4J Works on
Cypher
Community
Community
Graph

Businesses require information to make decisions. These decisions rely on numerous factors that surround their business. Although, companies collect large amounts of data, this data is usually found in large blocks that require sorting and allocation. This is where Neo4J can help by completely changing the way you get your data!

Neo4J is an open source NoSQL graph database that connects information and shows a trend in the data. The world is connected, where everything is related to something in some way. This is what Neo4J focuses on, it embraces relationships and uses it to store, process, and query connections efficiently.

This differs from other databases which only compute relationships at query, Neo4J stores connections in a manner which are readily available for any “join-like” navigation operation. This saves a lot of time and also gives more efficient results. Neo4J stores results in nodes and connects multiple nodes using relationships, both of which have arbitrary properties.

Neo4J is currently used by thousands of companies for many mission-critical production applications including scientific research, matchmaking, routing, organizational, network management, software analytics, and project management, social networks, recommendations, and more.

Advantages

Easy to represent connected data.

Does not require complex joins

Neo4j CQL query language commands are in human readable format and very easy to learn.

HIgh Availablity and Easy Retrieval

Features

01 SQL Like easy query language Neo4j CQL

02 It supports exporting of query data to JSON and XLS format

03 It supports full ACID(Atomicity, Consistency, Isolation and Durability) rules

04 It supports UNIQUE constraints

Features

05 It uses Native graph storage with Native GPE(Graph Processing Engine)

06 Neo4j facilitates you to scale the database by increasing the number of reads/writes, and the volume without affecting the data integrity and the speed of query processing.

07 Neo4j follows a data model called graph model. The graph contains nodes and the nodes are connected to each other. Nodes and relationships store data in key-value pairs known as properties.

08 Neo4j also provides a built-in Neo4j browser web application which can be used to create and retrieve your graph data.

Neo4J Customers

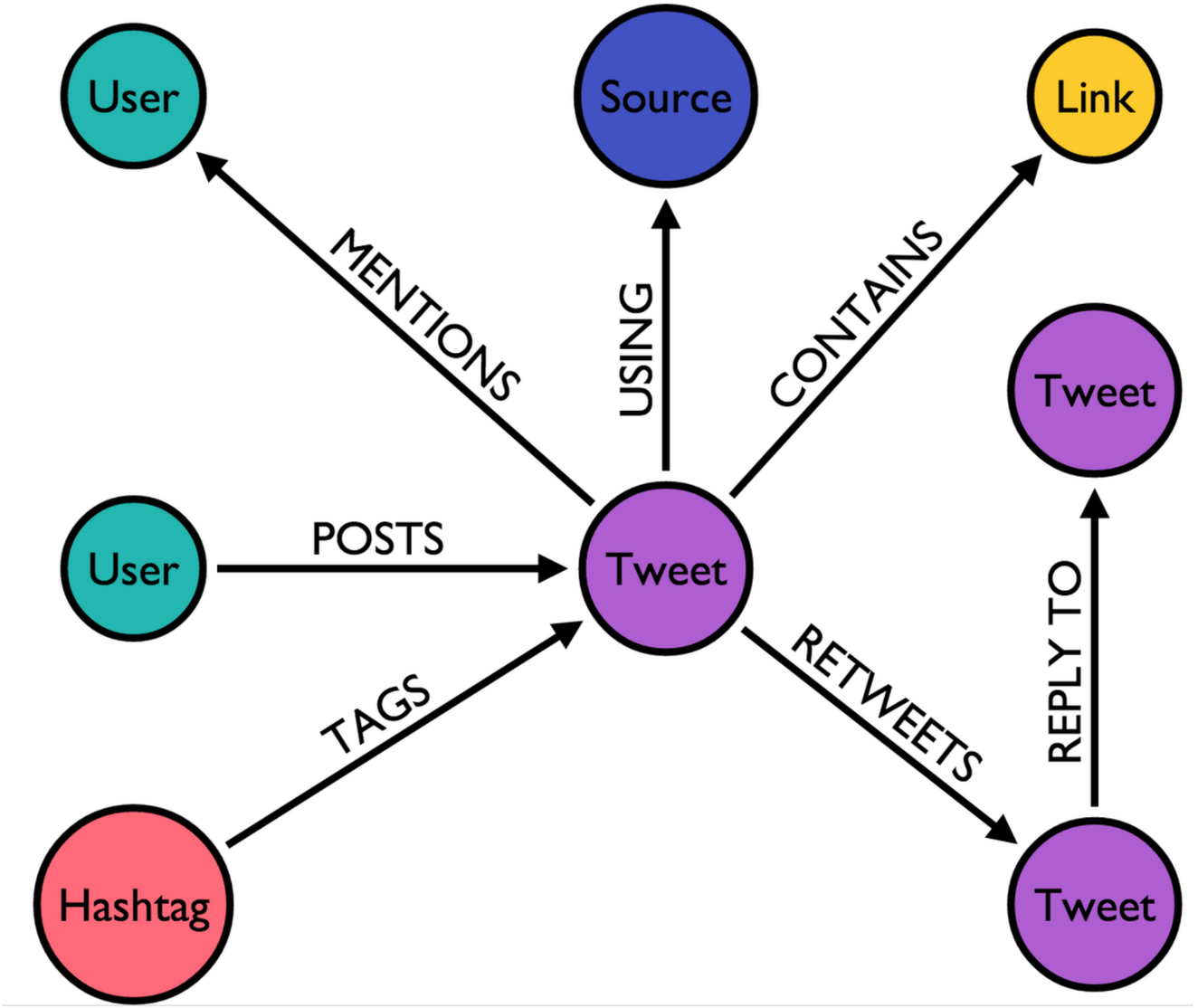
USE CASES OF NEO4J

NEO4J is widely used in various industries, including social media, e-commerce, finance, and healthcare, among others.
Here are some real-world use cases of Neo4j:

1.Social Networks

- Neo4j is used extensively by social networking sites such as LinkedIn, Facebook, and Twitter to manage and analyze the vast amount of data generated by their users.
- For example, Neo4j is used to store and manage the connections between users (such as friends and followers), as well as the interactions between them (such as likes, comments, and shares).

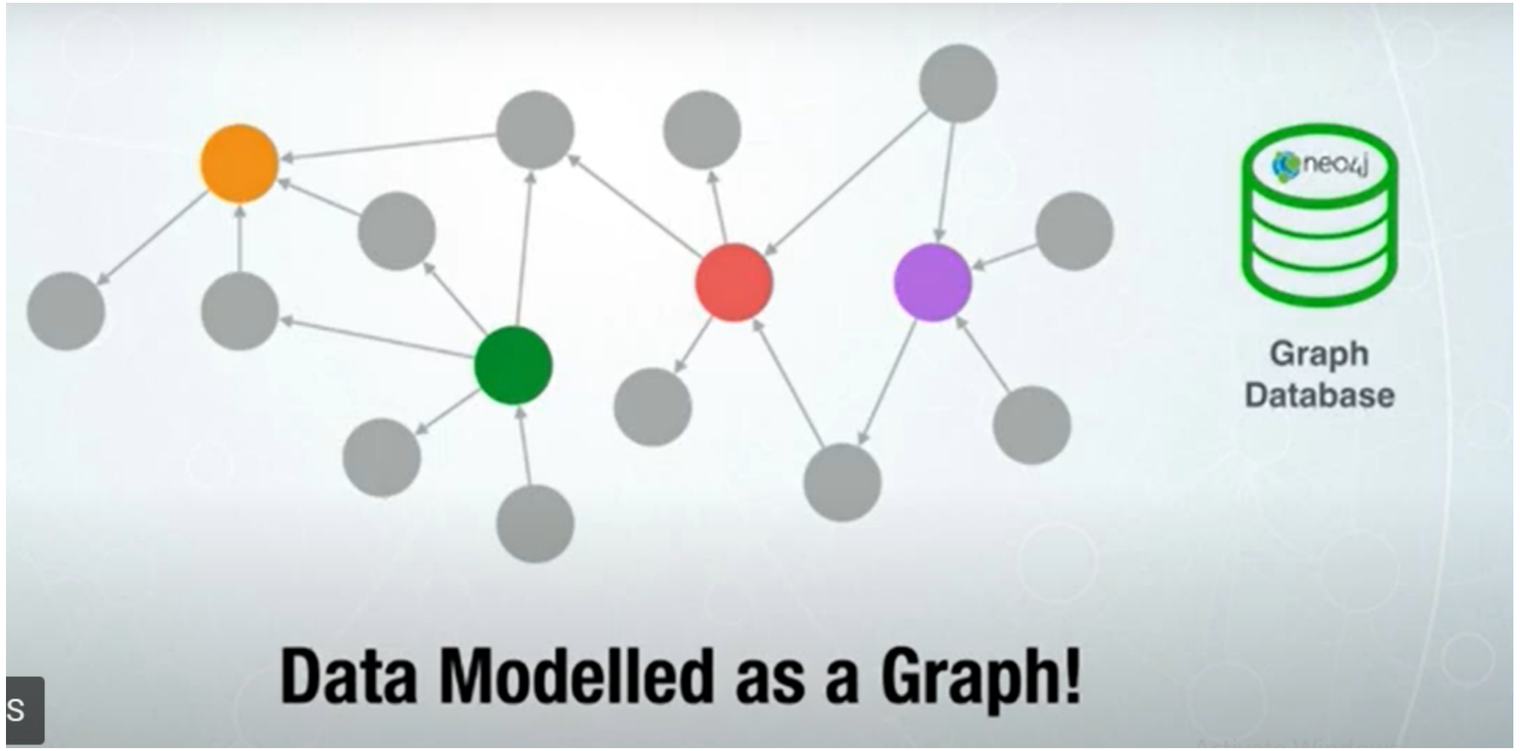
1.Social Networks



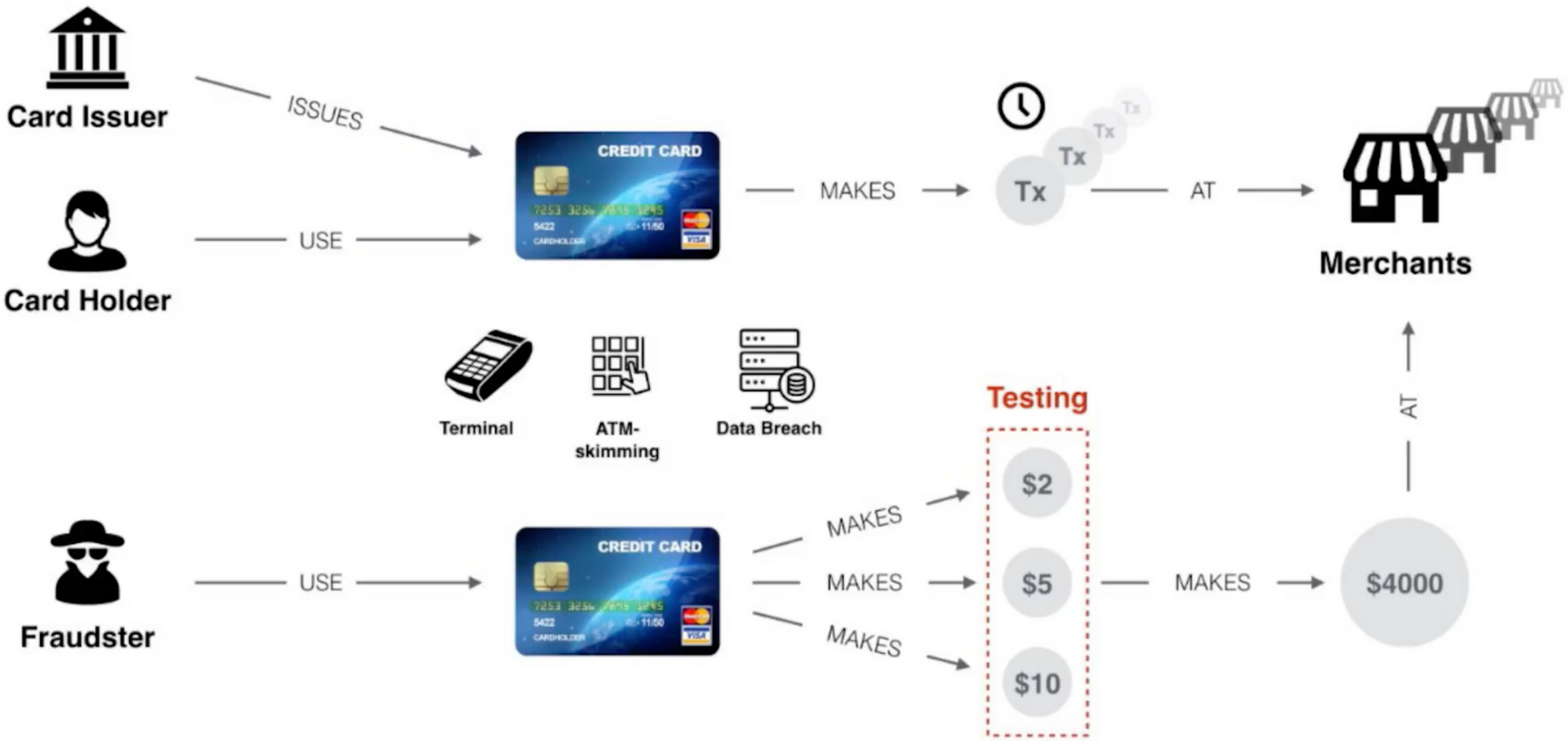
2.Fraud Detection:

- Neo4j is used by financial institutions and e-commerce platforms to detect fraudulent activities, such as credit card fraud, money laundering, and online scams.
- Neo4j can identify patterns of suspicious behavior and alert investigators to potential fraud. For example, if a large number of transactions originate from the same IP address, or if a user suddenly changes their behavior, this may be an indication of fraud.

2.Fraud Detection:



2.Fraud Detection:



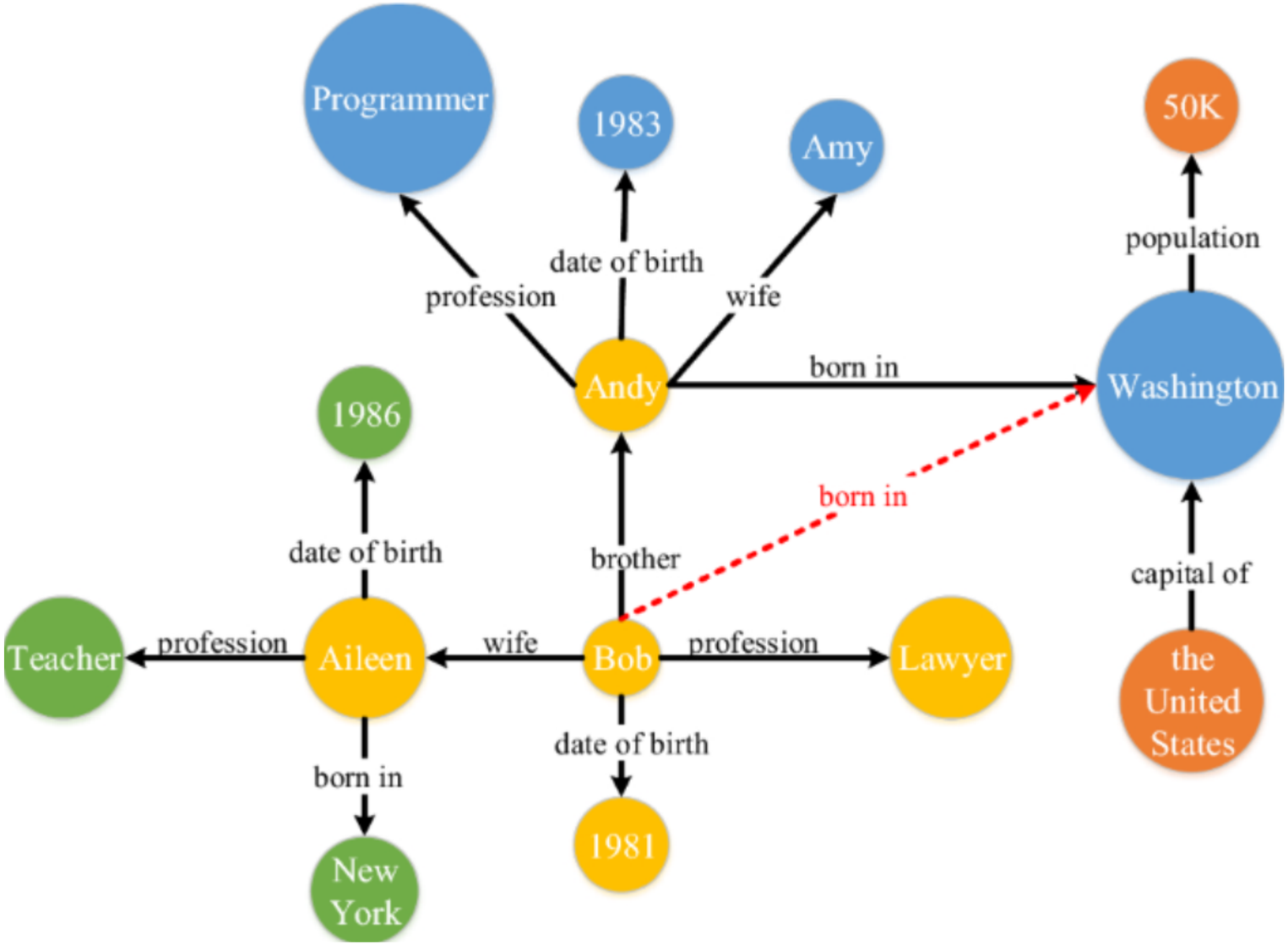
3.Recommendation Engines:

- Neo4j is often used to build recommendation engines, which are systems that make personalized recommendations based on a user's behavior and preferences.
- For example, **Netflix** uses Neo4j to analyze the relationships between movies and TV shows, and to recommend new content to users based on their viewing history.
- Similarly, **Amazon** uses Neo4j to analyze the relationships between products and to suggest new items to customers based on their purchase history.

4.Knowledge Graphs:

- A knowledge graph is a type of graph database that represents knowledge as a network of entities (or nodes) connected by relationships (or edges). In a knowledge graph, the nodes represent objects or concepts, and the edges represent the relationships between them.
- Knowledge graphs are designed to capture and represent complex relationships between entities and can be used for a wide range of applications, including natural language processing, recommendation systems, and search engines.
- A knowledge graph gets richer as new data is added. Through a combination of data, graph, and semantics (meaning), you get a knowledge graph with deep, dynamic context.

4.Knowledge Graphs:



4.Knowledge Graphs:

Types of Knowledge Graphs

Actioning Knowledge Graph

A data management knowledge graph that aims to drive action by providing data assurance, discovery, or insight.

Decisioning Knowledge Graph

A knowledge graph used for analytics, machine learning or data science where the aim is to improve decisions.

ARCHITECTURE OF NEO4J

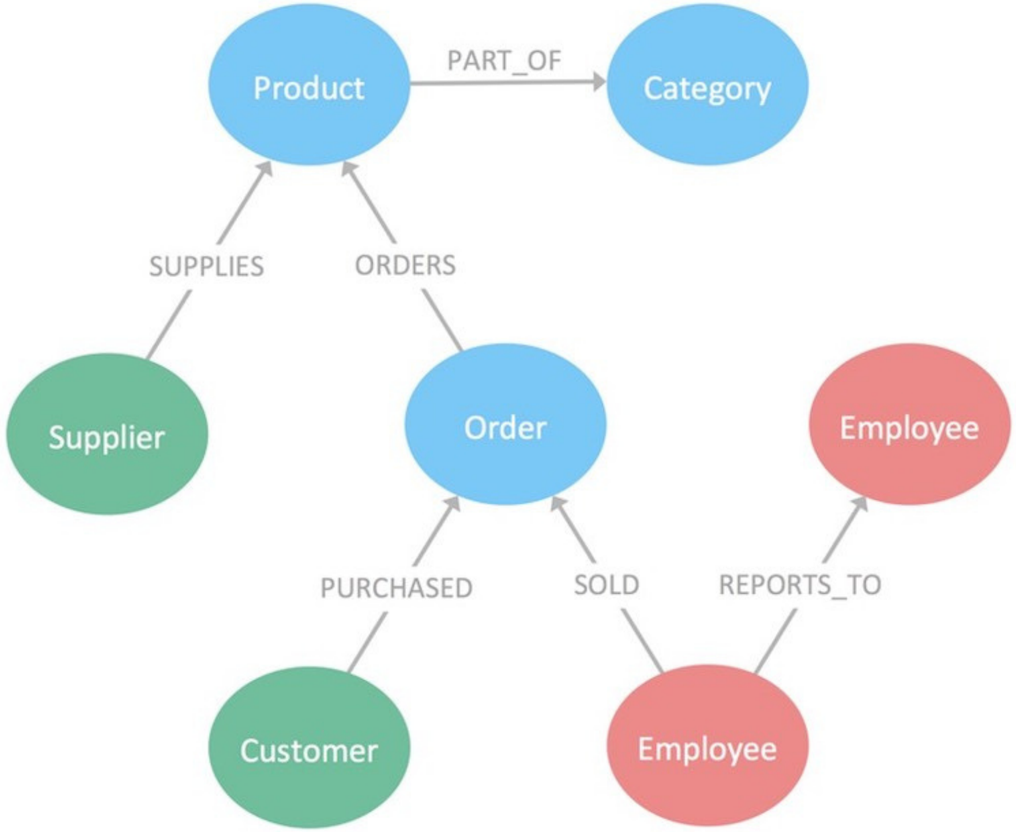
The architecture of Neo4j can be divided into three main components:

1.Storage Engine:

- The storage engine is responsible for managing the storage and retrieval of data in Neo4j.
- It stores data in a property **graph format**, where nodes represent entities and edges represent relationships between them.

1.Storage Engine:

NATIVE GRAPH DATABASE:



1.Storage Engine:

NATIVE GRAPH DATABASE:

- In addition to their ability to manage complex relationships, native graph databases also offer high performance and scalability. Because graph databases are optimized for graph traversal and pattern matching, they can quickly and efficiently query large graphs with billions of nodes and edges.
- They also offer **high availability and fault tolerance** through features like replication and clustering.

2.Query Engine:

- The query engine is responsible for processing queries against the graph data stored in the storage engine.
- It supports a declarative query language called **Cypher,** which allows developers to express complex graph queries in a concise and intuitive way.

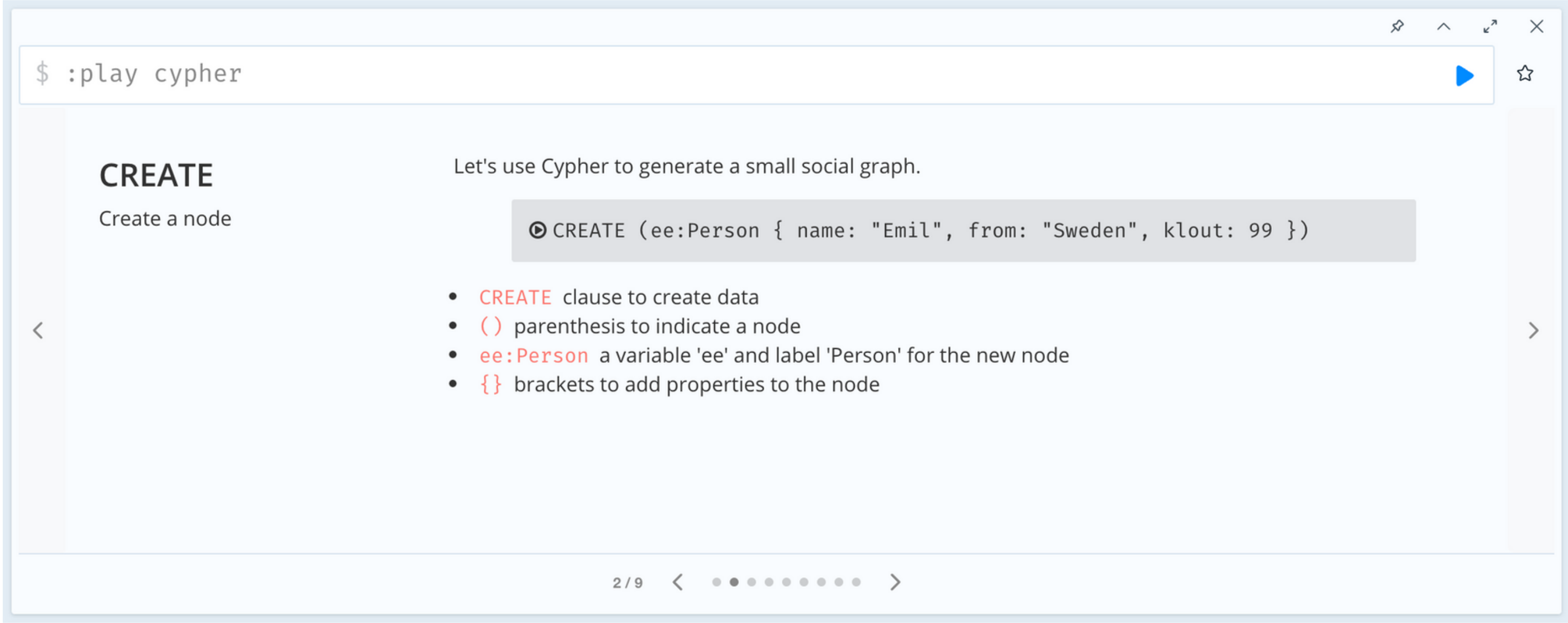
2.Query Engine:

CYPHER LANGUAGE:

- Cypher is Neo4j’s graph query language that lets you retrieve data from the graph. It is like SQL for graphs and was inspired by SQL, so it lets you focus on what data you want out of the graph (not how to go get it).

2.Query Engine:

CYPHER LANGUAGE:



3.API Layer:

- The API layer provides a set of APIs that developers can use to interact with Neo4j programmatically.
- It includes drivers for popular programming languages such as Java, Python, and JavaScript, as well as RESTful APIs for web-based applications. The API layer also includes tools for managing and monitoring the Neo4j instance, such as the Neo4j Browser and the Neo4j Management Console.

3.API Layer:

The Neo4j API layer includes several different APIs, each with its own specific purpose and functionality. Some of the most commonly used APIs in Neo4j include:

- **Cypher API**: This is a query language for Neo4j, used to perform operations such as creating, updating, and querying the database.
- **Bolt API:** This is a binary protocol used to connect to and interact with Neo4j over a network.
- **Java API:** This is a set of Java libraries that can be used to interact with Neo4j from Java-based applications.
- **REST API:** This is a web-based API that allows developers to interact with Neo4j using HTTP requests.

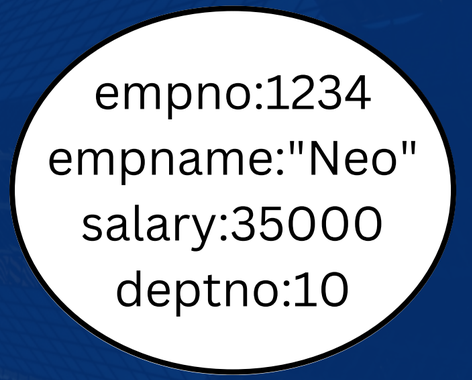
HOW NEO4J WORKS

How Information stored in Neo4j

Information is organized as nodes, relationships, and properties.

Nodes

- can have labels
- can have properties



Employee Node

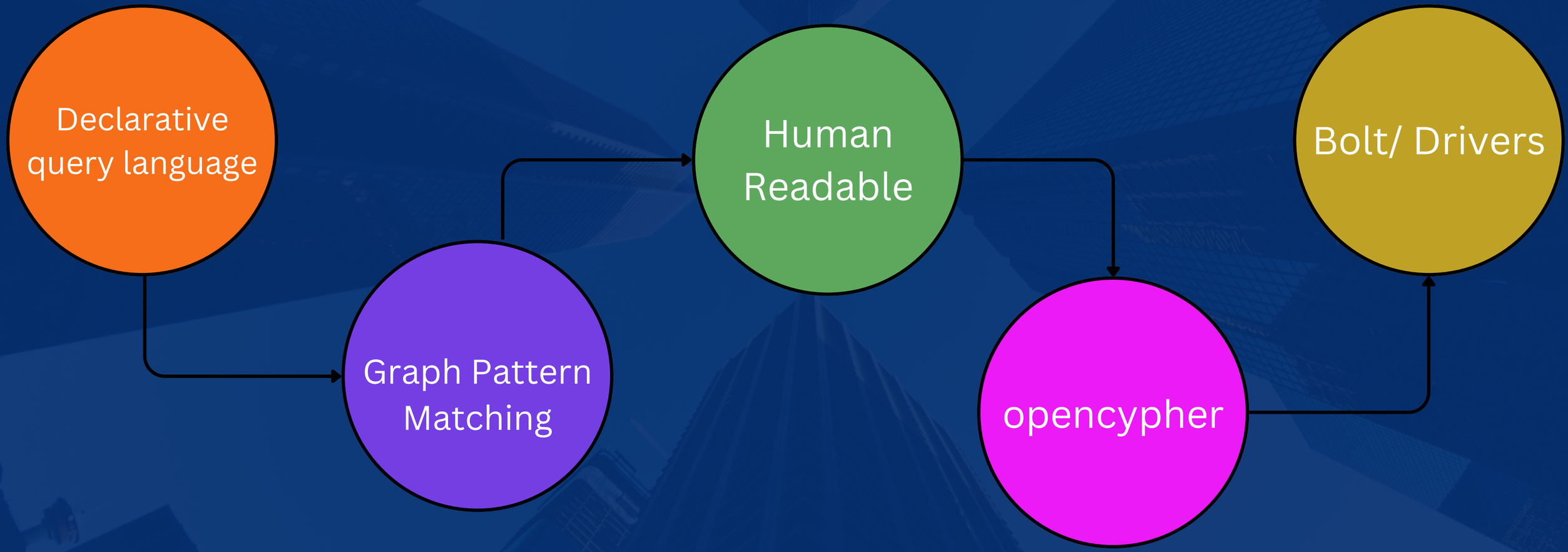
Here, Node Name = "Employee" and it contains a set of properties as key-value pairs.

Relationships

- relate nodes by type and direction
- can have properties

Here, Emp and Dept are two different nodes. "WORKS_FOR" is a relationship between Emp and Dept nodes
Each relationship contains one start node and one end node.
Here, "Emp" is a start node, and "Dept" is an end node.

What is Cypher?



Declarative query language

Declarative query language is means we specify the data that we are interested in not how to fetch that data from the database

Graph Pattern Matching

It gives us the expressiveness in cipher so nodes are defined within parenthesis with brackets where we define a graph pattern that we are looking for in the graph.

Human Readable

Cipher is intended to be human readable this helps us to allow on our domain focus on our data not necessarily the database operations

opencypher

Cypher is of course open source is through the open cypher project, cypher is open standard for graph quering cypher project aims to provide information for implementing cipher in different systems so cypher is not specific to neo4j

This is not specific to cypher but for neo4j uses binary protocol called bolt to speak cypher to the database, it allows for compressing data

Bolt/ Drivers

CRUD

Cypher is made for performing CRUD operations on graphs.

ASCII-Art

The property graph model comprises nodes and relationships which together make patterns. Cypher is also based on patterns that employ ASCII-Art, hence providing a visual way for querying graphs:

- Nodes are represented using round brackets () .
- Relationships are represented using --> or <--.
- Relationship types are represented using square brackets [].

Cypher also allows specifying labels and properties conveniently.

Example

Node

Node

```
MATCH(:Person{name:“Dan”})-[:LOVES]→(:Person{name:“Ann”})
```

LABEL PROPERTY

LABEL PROPERTY

Clauses in Cypher Query Language

- Writing clauses
- Reading clauses
- Projecting clauses
- Reading sub-clauses
- Reading hints
- Reading/Writing clauses
- Set operations
- Importing data
- Schema clauses

Write clauses of Neo4j Cypher Query Language

Sr.No	Write Clause	Usage
1	CREATE	This clause is used to create nodes, relationships, and properties.
2	MERGE	This clause verifies whether the specified pattern exists in the graph. If not, it creates the pattern.
3	SET	This clause is used to update labels on nodes, properties on nodes and relationships.
4	DELETE	This clause is used to delete nodes and relationships or paths etc. from the graph.
5	REMOVE	This clause is used to remove properties and elements from nodes and relationships.
6	FOREACH	This class is used to update the data within a list.
7	CREATE UNIQUE	Using the clauses CREATE and MATCH, you can get a unique pattern by matching the existing pattern and creating the missing one.
8	Importing CSV files with Cypher	Using Load CSV you can import data from .csv files.

Creating Nodes

- Creating a Single Node

create (employee)

- Creating Multiple Nodes

create (e : employee), (d : department)

- Creating Nodes with labels

create (e : employee)

- Creating Nodes with multiple labels

create (e : employee: professor)

- Create Node with properties

create (e : employee{name:'batool',address:'karachi'})

- Create Multiple node with properties.

create (e : employee{name:'batool'}), (d: department{name:'marketing'})

Read clauses of Neo4j Cypher Query Language

Sr.No	Read Clauses	Usage
1	MATCH	This clause is used to search the data with a specified pattern.
2	OPTIONAL MATCH	This is the same as match, the only difference being it can use nulls in case of missing parts of the pattern.
3	WHERE	This clause id is used to add contents to the CQL queries.
4	START	This clause is used to find the starting points through the legacy indexes.
5	LOAD CSV	This clause is used to import data from CSV files.

Match Clause

Get all nodes

match (n) return n

Get all nodes with a label

match (e : employee) return e

Get name property of all nodes of a specific type

match (e : employee) return e.name

Get name property of all nodes of all labels

match (e : employee) return e.name

Get name peoperty of all nodes of all labels

match(e: employee),(d: department) return e.name, d.name

Get name property of nodes only belonging to two specific labels

match (e : employee{name:'batool'}) return e

Creating Relationships

We can create a relationship using the CREATE clause. We will specify relationship within the square braces “[ ]” depending on the direction of the relationship it is placed between hyphen “ - ” and arrow “ → ”

Example

First of all, create two nodes Pak and Hasan in the database:

CREATE (Hasan :player{name: "Hasan Ali", YOB: 1985, POB: "Pakistan"})

CREATE (Hasan: Country {name: "Pakistan"})

Now, create a relationship named BATSMAN_OF between these two nodes as –

CREATE (Hasan)-[r:BATSMAN_OF]->(Pak)

Finally, return both the nodes to see the created relationship.

RETURN Hasan, Pak

Merge Command

MERGE command is a combination of CREATE command and MATCH command.

MERGE (Waseem:player) RETURN Waseem

When you execute this query, Neo4j verifies whether there is any node with the label player. If not, it creates a node named “Waseem” and returns it.
If, there exists any node with the given label, Neo4j returns them all.

String Functions

Neo4J CQL has provided a set of String functions to use them in CQL Queries to get the required results.

1	UPPER It is used to change all letters into upper case letters.
2	LOWER It is used to change all letters into lower case letters.
3	SUBSTRING It is used to get substring of a given String.
4	**Replace** It is used to replace a substring with a given substring of a String.

Create Index

In Neo4j, index is a data structure which is used to improve the speed of data retrieval operations in a database.
An index can be created over a property on any node that has been given a label. Once an index is created, Neo4j will manage it and keep it up to date whenever the database is changed.
CREATE INDEX ON statement is used to create an index.

Every time you create an index Neo4j will create the index in the background. If your database is large, this may take some time. Only when Neo4j has finished creating the index, will it be brought online, and it can be used in queries.

Create Constraints

In Neo4j, a constraint is used to place restrictions over the data that can be entered against a node or a relationship.

There are two types of constraints in Neo4j:

Uniqueness Constraint: It specifies that the property must contain a unique value. (For example: no two nodes with an player label can share a value for the Goals property.)

Property Existence Constraint: It makes ensure that a property exists for all nodes with a specific label or for all relationships with a specific type.

Comparison with Other Databases

Relational Database

IRelational databases organize data into tables and enforce a strict schema. Data relationships are typically represented using foreign keys. In contrast, Neo4j is a graph database that stores data as nodes and relationships, allowing for more flexible data modeling and querying. Neo4j can handle complex queries involving relationships between nodes more efficiently than relational databases. Neo4j is also more scalable than relational databases, making it better suited for applications that require large-scale data processing

NoSQL Databases

INoSQL databases are designed to handle large amounts of unstructured data. While Neo4j is also capable of handling unstructured data, its primary strength lies in handling structured data that has complex relationships. Unlike many NoSQL databases, Neo4j supports ACID transactions, making it a better choice for applications that require strong data consistency.

Other Graph Databases

There are several other graph databases available, but Neo4j is unique in several ways. For one, Neo4j is more mature and has a larger community of users than most other graph databases. Additionally, Neo4j supports Cypher, a powerful query language specifically designed for graph databases. Finally, Neo4j has a range of enterprise features such as clustering, backup and restore, and role-based access control that make it more suitable for large-scale production deployments.

Neo4j and Data Science

Neo4j is a powerful tool for data science, particularly for applications that involve complex data relationships. Here are some ways in which Neo4j can be used in data science:

Data Modeling

Data modeling and exploration: Neo4j allows data scientists to create flexible data models that can capture complex relationships between data points. This makes it easier to explore the data and identify patterns that might be missed using traditional data modeling approaches.

Graph Algorithms

Neo4j has a built-in library of graph algorithms that can be used to perform advanced graph analysis on data. These algorithms can be used to identify key nodes in a graph, detect communities and clusters, and perform pathfinding and traversal operations.

Recommendations Engine

Neo4j is particularly well-suited for building recommendation engines, which involve identifying patterns in the data to make personalized recommendations to users. For example, a social network could use Neo4j to recommend new connections based on shared interests and connections.

Neo4j can be used to identify fraud patterns in large datasets. By modeling relationships between data points, it becomes easier to detect suspicious activity that might be missed using traditional data analysis methods.

Fraud Detection

Machine Learnin

Neo4j can be integrated with machine learning frameworks to build predictive models based on graph data. For example, machine learning algorithms can be used to predict which nodes are likely to be connected in the future, based on historical data.

Overall, Neo4j's strengths in data modeling, graph analysis, and recommendation engines make it a powerful tool for data scientists working with complex datasets. Its ability to integrate with machine learning frameworks also makes it a valuable addition to the data science toolkit.

Conclusion

Future Scope of Neo4j in Data science

The future of Neo4j in data science is bright, as the use of graph databases and graph-based analysis is becoming increasingly important in many areas of data science. Here are some potential future applications for Neo4j in data science:

Knowledge Graphs

Knowledge graphs are becoming increasingly important for representing complex relationships between entities in a variety of fields, from healthcare to finance to e-commerce. Neo4j is well-suited for building and querying knowledge graphs, making it a valuable tool for data scientists working in these areas.

Neo4j can be used in conjunction with natural language processing (NLP) techniques to analyze text data and identify relationships between entities. This can be useful in a variety of applications, from social media analysis to customer feedback analysis.

Natural language processing

Internet of Things

The Internet of Things (IoT) involves connecting a large number of devices and sensors to the internet, generating massive amounts of data. Neo4j can be used to model the relationships between these devices and sensors, making it easier to analyze and make sense of the data generated by IoT systems.

Personalization

Neo4j is well-suited for building personalized recommendation engines, which can be used to recommend products, services, and content to users based on their preferences and behavior. This can be useful in a variety of applications, from e-commerce to media and entertainment.

Graph Analytics

As the use of graph-based analysis becomes more widespread, there is likely to be an increased demand for tools that can perform advanced graph analytics. Neo4j's built-in library of graph algorithms and its ability to integrate with machine learning frameworks make it a valuable tool for performing these kinds of analyses.

Conclusion

Overall, Neo4j's strengths in data modeling, graph analysis, and recommendation engines make it a powerful tool for data scientists, and its ability to integrate with other technologies makes it well-suited for a variety of data science applications.

PROJECT DEMONSTRATION AND LAB SESSION

References:

- **Neo4j Graph Data Platform | Graph Database Management System**
- **https://www.javatpoint.com/features-of-neo4j**
- **https://www.tutorialspoint.com/neo4j/neo4j_features_advantages.htm**
- **https://www.tutorialspoint.com/neo4j/neo4j_quick_guide.htm**