The Data Model

COMP3008

Lauren Ansell

Introduction

Today's topics:

- Introduce the data model.
- Introduce the MERGE, WITH and COLLECT functions

Session learning outcomes - by the end of today's lecture, you will be able to:

- Explain what a data model is.
- Queries in Neo4j

Why is Data Modeling Important?

A comprehensive and optimised data model helps create a simplified, logical database.

It also equips all systems with a ‘single source of truth’.

Types of Data Modeling

There are three primary data model types:

1. relational,
2. dimensional,
3. entity-relationship (E-R).

The model type defines the logical structure - how the data is stored, logically - and therefore how it is stored, organised, and retrieved.

Levels of Data Abstraction

There are many types of data models, with different types of possible layouts.

The data processing community identifies three kinds of modeling to represent levels of thought as the models are developed.

- Conceptual data model
- Logical data model
- Physical data model

Data Modeling Examples

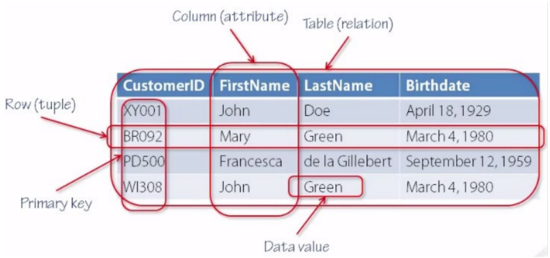
- ER (Entity-Relationship) Model
- Hierarchical Model
- Network Model
- Relational Model
- Object-Oriented Database Model
- Object-Relational Model

Relation Databases

A relational database is a database whose organisation is based on the relational model of data, as proposed by E. F. Codd in 1970.

The relational model organises data into one or more tables (or "relations") of columns and rows, with a unique key identifying each row.

Rows are also called records.



CustomerID	FirstName	LastName	Birthdate
XY001	John	Doe	April 18, 1929
BR092	Mary	Green	March 4, 1980
PD500	Francesca	de la Gillebert	September 12, 1959
WI308	John	Green	March 4, 1980

Non-relational Databases

Non-relational databases are sometimes referred to as “NoSQL”.

The main difference between these is how they store their information.

A non-relational database stores data in a non-tabular form, and tends to be more flexible than the traditional, SQL-based, relational database structures.

It does not follow the relational model provided by traditional relational database management systems.

NoSQL Features - Recap

Non-relational

Cluster friendly

Open source

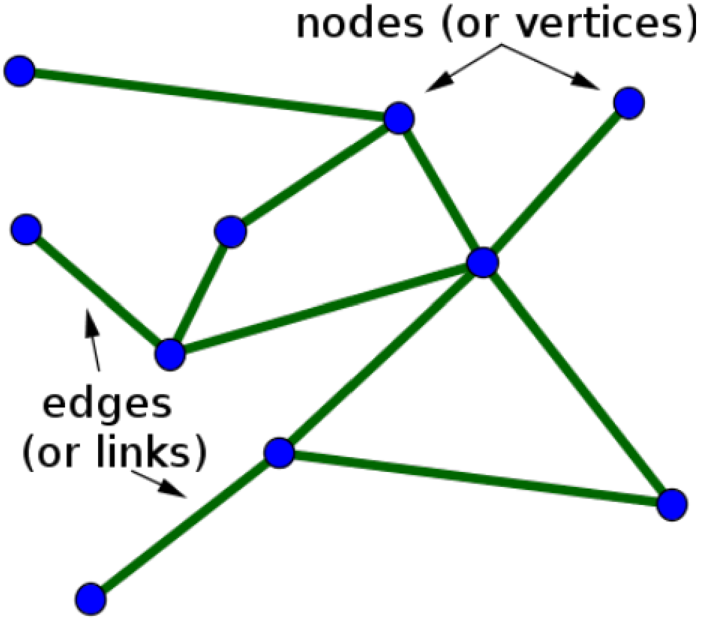
21st Century Web

Graph Databases

Graph databases are based on graph theory, and use nodes and edges to represent and store data.

A graph database is essentially a collection of nodes and edges.

Each node represents a record in the database, and each edge represents a connection or relationship between two records.

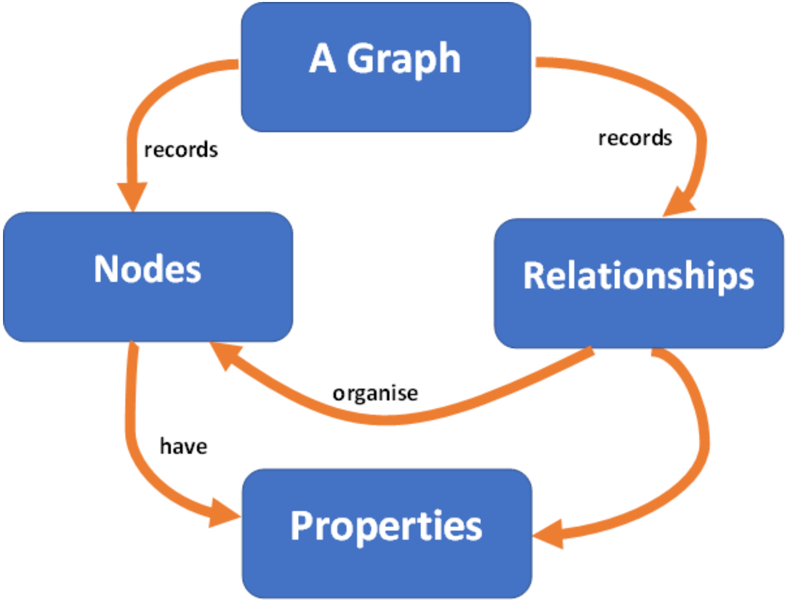

nodes (or vertices)
edges
(or links)

Terminology

Nodes are roughly the equivalent of the **record** or **row** in a relational database.

Edges, also known as **relationships**, are the arrows that connect nodes to other nodes; they represent the relationship between them.

Properties are pertinent information that relate to nodes and relationships.

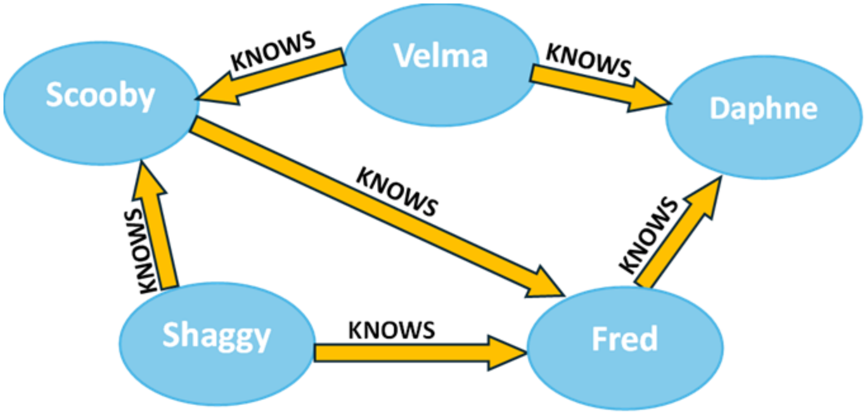

A Graph
records
records
Nodes
Relationships
organise
have
Properties

Relationships

Relationships always have a **type**: for example, KNOWS.

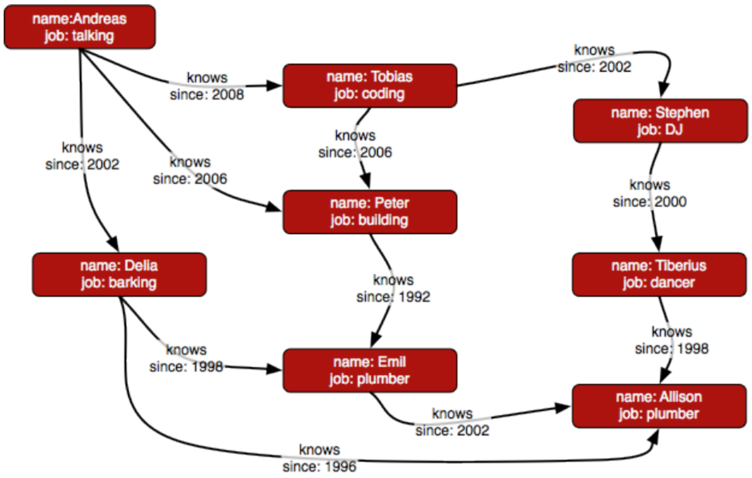
Relationships always have direction.

A relationship always have a **start node**, and an **end node**.


Velma
KNOWS
Scooby
KNOWS
Daphne
KNOWS
KNOWS
KNOWS
Shaggy
KNOWS
Fred

Properties

Each node or relationship has a collection of key-value pairs, or "**properties**", describing it.



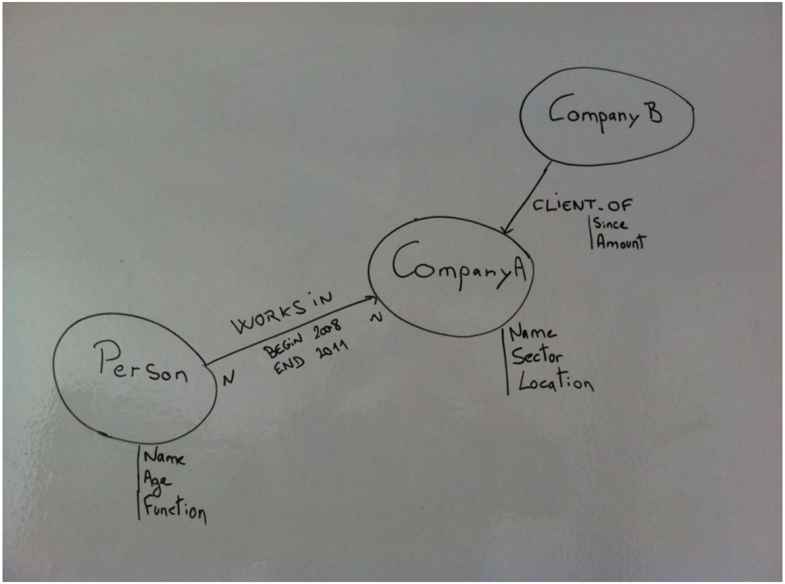
Data Model

A data model is a tool, typically a diagram, to describe the structure of a database. The data model is a diagram showing the nodes contained in a graph database and their relationships.

Nodes and relationships are tagged with labels representing their roles.

The diagram also shows the properties (or attributes) of the nodes and their relationships.

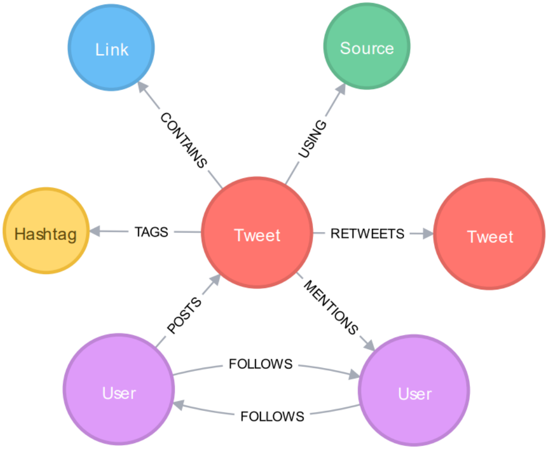
Data Model Example



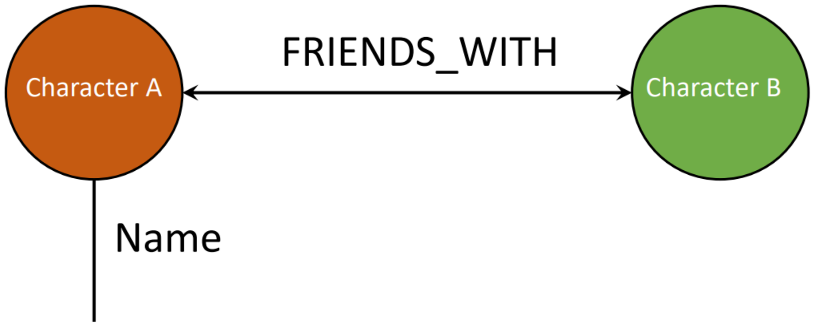
Data Model - Property Graph Model

A property graph model shows the connected entities - the nodes - and their relationships in a graph database.

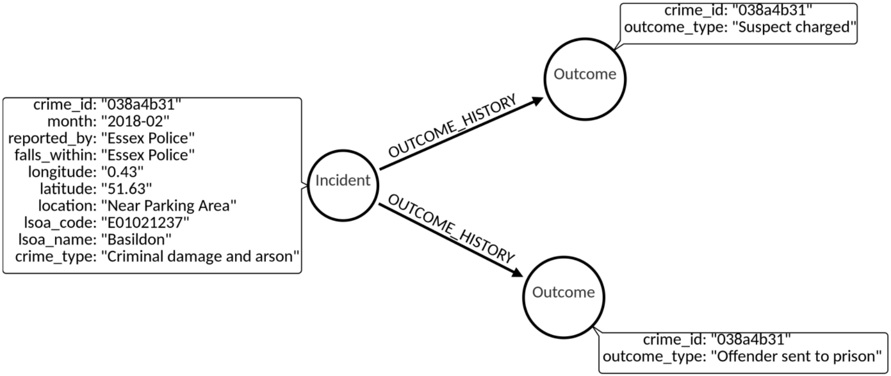
Nodes are tagged with labels representing their roles.



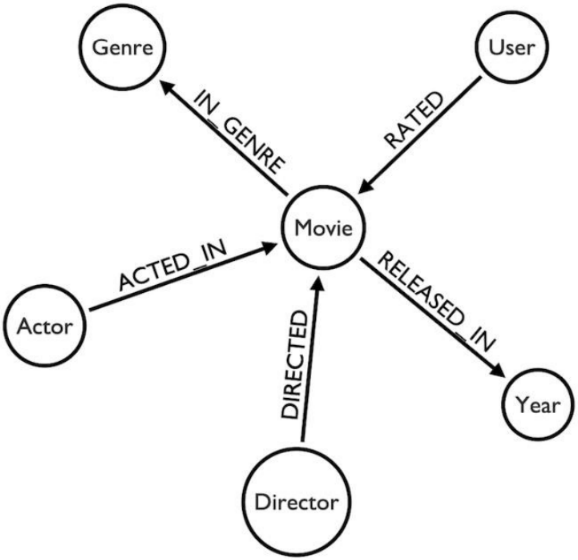
Data Model



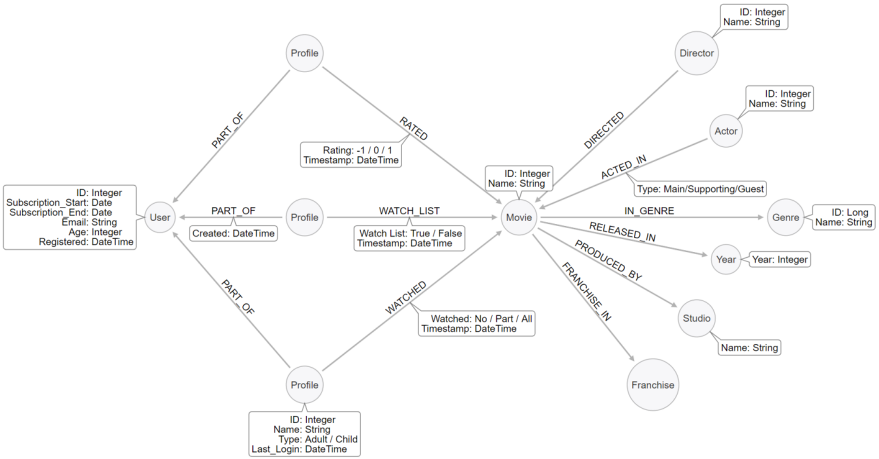
Crime Data


crime_id: "038a4b31"
month: "2018-02"
reported_by: "Essex Police"
falls_within: "Essex Police"
longitude: "0.43"
latitude: "51.63"
location: "Near Parking Area"
lsoa_code: "E01021237"
lsoa_name: "Basildon"
crime_type: "Criminal damage and arson"
Incident
OUTCOME_HISTORY
OUTCOME_HISTORY
Outcome
crime_id: "038a4b31"
outcome_type: "Suspect charged"
Outcome
crime_id: "038a4b31"
outcome_type: "Offender sent to prison"

Movie Database



Netflix


Profile
PART_OF
RATED
Rating: -1 / 0 / 1
Timestamp: DateTime
ID: Integer
Subscription_Start: Date
Subscription_End: Date
Email: String
Age: Integer
Registered: DateTime
User
PART_OF
Created: DateTime
Profile
WATCH_LIST
Watch List: True / False
Timestamp: DateTime
ID: Integer
Name: String
Movie
PART_OF
WATCHED
Watched: No / Part / All
Timestamp: DateTime
Profile
ID: Integer
Name: String
Type: Adult / Child
Last_Login: DateTime
ID: Integer
Name: String
Director
DIRECTED
ID: Integer
Name: String
Actor
ACTED_IN
Type: Main/Supporting/Guest
IN_GENRE
Genre
ID: Long
Name: String
RELEASED_IN
Year
Year: Integer
PRODUCED_BY
Studio
Name: String
FRANCHISE_IN
Franchise

MERGE

MERGE ensures that a pattern exists in the graph.

Either the pattern already exists, or it must be created.

MERGE can be used to match or create a relationship.

MERGE can also be used to create simple queries.

MERGE Example

```
MATCH (r:Person {name:  'Romeo'})
MATCH (j:Person {name:  'Juliet'})
MERGE (r)-[:IN_LOVE]->(j)
```

MERGE finds or creates a relationship between the nodes.

WITH

WITH separates query parts explicitly.

This allows us to declare which variables we want to carry over to the next part of the query.

It is important to note that WITH affects variables in scope.

Any variables not included in the WITH clause are not carried over to the rest of the query.

The wildcard * can be used to include all variables that are currently in scope.

Example

```
MATCH (george {name: ’George’})<–(otherPerson)
WITH otherPerson, toUpper(otherPerson.name) AS
upperCaseName
WHERE upperCaseName STARTS WITH ’C’
RETURN otherPerson.name
```

Example with wildcard

```
MATCH (person)-[r]->(otherPerson)
WITH *, type(r) AS connectionType
RETURN person.name, otherPerson.name, connectionType
```

COLLECT()

The function COLLECT() returns a single aggregated list containing the values returned by an expression.

```
MATCH (person:Person)
WHERE 'Ozzy' IN COLLECT { MATCH
(person)-[:HAS_DOG]->(dog:Dog) RETURN dog.name }
RETURN person.name AS name
```

Summary

Data model

- The data model is a diagram showing the entities - or nodes - contained in a graph database and their relationships.
- Nodes are tagged with labels representing their roles.
- The diagram also shows the properties (or attributes) of the nodes and their relationships.

Neo4j functions

- MERGE ensures that a pattern exists in the graph.
- WITH allows us to have more flexibility in our queries by allowing us to declare which variables to carry over.
- COLLECT() brings the results of an expression together into a single aggregated list.