THE UNIVERSITY OF WESTERN AUSTRALIA

Data Warehousing

Lecture 8 – Graph DBs and Graph Data Modelling

CITS3401
CITS5504

Dr. Sirui Li

Computer Science and Software Engineering

School of Maths, Physics and Computing

Acknowledgement: The lecture slides are adopted from online sources.

Lecture Outline

THE UNIVERSITY OF WESTERN AUSTRALIA

- **What is a Graph**
- Graph Databases vs. Relational
- Graph Data Modelling
  - From Scratch
  - From an existing relational DB

What is a graph?

THE UNIVERSITY OF WESTERN AUSTRALIA

- A graph is a collection of **vertices** and **edges,** also known as **nodes** and **relationships.**
- We can model all sorts of scenarios using graphs – social networks, natural language, scientific papers, etc.

Examples of graphs in use today

THE UNIVERSITY OF WESTERN AUSTRALIA

Facebook's social network represents friendship among a set of users. It helps users connect with friends or find new friends based on interests, location, or mutual connections.

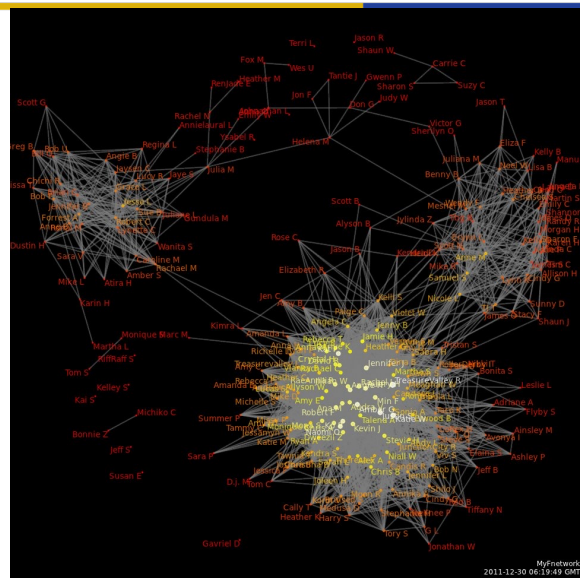
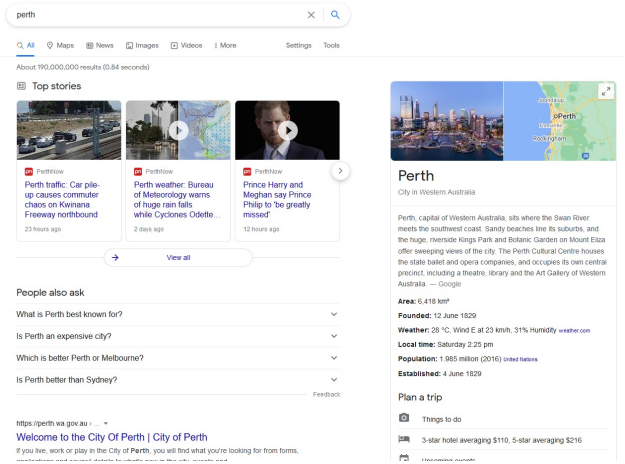


Image from https://en.wikipedia.org/wiki/Social_network_analysis

Examples of graphs in use today

Google's search functionality is made possible via a knowledge graph.



Labels and Relationships

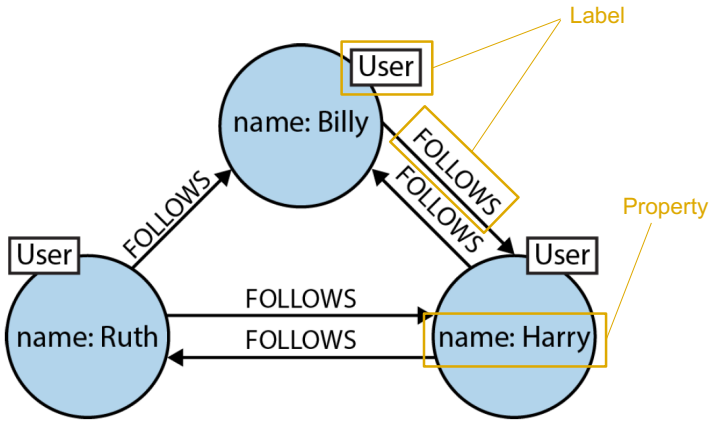
- Nodes can have **multiple labels**.
- We can use labels to group nodes – this way, we can ask the database to find all nodes labelled "User", or perhaps "User" and also "Admin".

- Relationships in a graph **naturally form paths**.
- This makes graph databases **extremely efficient** for querying, particularly when the data is heavily connected.

Property graph model

- The most common form of graph model is the **property graph model**, whereby:
  - The graph contains **nodes** and **relationships.**
  - A node may have zero or more **properties** (key-value pairs).
  - Nodes can be labelled with one or more **labels.**
  - Relationships can be **named** and **directed**, and always have a start and end node.
  - Relationships can also contain **properties**.

Example: Twitter



*No relationship properties are shown here. An example might be the date in which the person followed another person.

Lecture Outline

- What is a Graph
- **Graph Databases vs. Relational**
- Graph Data Modelling
  - From Scratch
  - From an existing relational DB

Graph vs Relational: The Irony of RDBMS

- RDBMS were originally designed to **codify** paper forms and tabular structures.
- Relational databases are excellent for storing **tabular data**, but are not ideal for storing data where relationships are involved, despite their name!

Relational Database Example

User

UserID	User	Email	Address
1	Alice	alice@example.org	1 Duck St, ...
2	Bob	bob@example.org	1 Duck St, ...
3	...	...	37 Rabbit Lane...
4	Zach	zach@example.org	49 Rabbit Lane...

Product

ProductID	Description	Handling
321	Strawberry ice cream	freezer
765	Potatoes	null
...	...	...
987	Dried spaghetti	null

Order

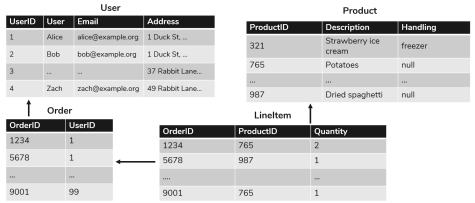
OrderID	UserID
1234	1
5678	1
...	...
9001	99

LineItem

OrderID	ProductID	Quantity
1234	765	2
5678	987	1
....		...
9001	765	1

Relational Database Example

- "Which items did a customer buy?"
- "Which customers bought this product?"
- "Which customers buying *this* product also bought *that* product?"



Relational Database Example

- Queries across multiple tables are
  - inefficient yet doable
  - prohibitively slow
- RDBMs schemas are **inflexible**, and can't keep up with dynamic and uncertain variables.
- Relational databases **aren't effective at handling data relationships**, especially when those relationships are added or adjusted on an ad hoc basis.

User

UserID	User	Email	Address
1	Alice	alice@example.org	1 Duck St, ..
2	Bob	bob@example.org	1 Duck St, ..
3	..	..	37 Rabbit Lane..
4	Zach	zach@example.org	49 Rabbit Lane..

Product

ProductID	Description	Handling
321	Strawberry ice cream	freezer
765	Potatoes	null
...	...	...
987	Dried spaghetti	null

Order

OrderID	UserID
1234	1
5678	1
...	...
9001	99

LineItem

OrderID	ProductID	Quantity
1234	765	2
5678	987	1
....		...
9001	765	1

"I suppose it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail".
- Abraham Maslow (The Psychology of Science, 1966)

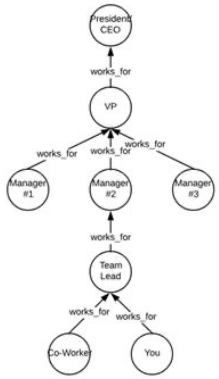
How the Graph Model differs from the Relational Model

- There are **no nulls**. Non-existing value entries (properties) are just not present.
- It describes the relationships in more detail.
  - For example, we know that an employee SOLD an order rather than having a foreign key relationship between the Orders and Employees tables.
  - We could also choose to add more **metadata** about that relationship, should we wish.
- Either model can be more normalized.
  - For example, addresses have been denormalized in several of the tables, but could have been in a separate table.
  - In a future version of our graph model, we might also choose to separate addresses from the Order (or Supplier or Employee) entities and create separate Address nodes.
- **No nulls!**

Recursive Queries

Given a list of employees and managers in a company, how we would determine a person's reporting hierarchy?

```
g.V().
  repeat(
    out('works_for')
  ).path().next()
```




```
WITH RECURSIVE org AS (
     SELECT employee_id,
            manager_employee_id,
            employee_name,
            1 AS level
     FROM org_chart
  UNION
     SELECT m.employee_id,
            e.manager_employee_id,
            e.employee_name,
            m.level+1 AS level
     FROM org_chart AS e
       INNER JOIN org AS m ON
e.manager_employee_id = m.employee_id
  )

SELECT employee_id,
manager_employee_id, employee_name,
FROM org
ORDER BY level ASC;
```


Summary – Relational DB vs. Graph DB

- Because relational DBs were originally designed to codify tabular forms, they are not ideal for storing heavily related data.
- Graph DBs, on the other hand, excel when dealing with **highly connected data**.
- Graph DBs do **not need to store null values**, and we can add **new properties** or node types (labels) **on the fly**.
- They are also much better at handling **recursive queries** and **queries across multiple entities**.

Lecture Outline

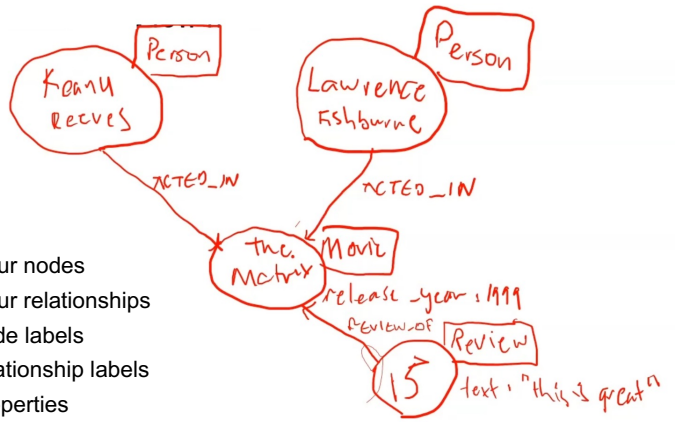
- What is a Graph
- Graph Databases vs. Relational
- **Graph Data Modelling**
  - **From Scratch**
  - From an existing relational DB

Intro to Graph Data Modelling

- **Data modelling** aims to bring specific facets of an unruly domain into a space where they can be **structured** and **manipulated**.
- Graph representations are one way to model data, and are excellent when modelling **highly-interconnected** data.
- Unlike relational database modelling, we do not need to deviate from **natural language representations** when modelling graphs.

Planning out a graph DB

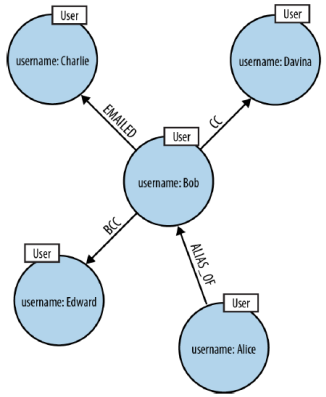
Graph data modelling is whiteboard-friendly. Let's plan a graph model for a few movies…



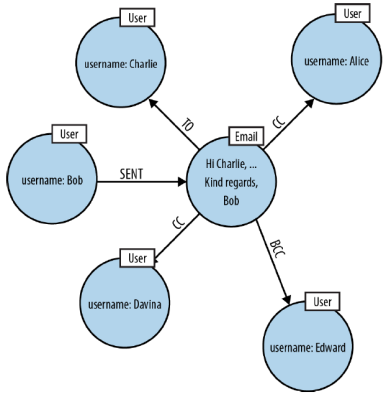
1. Plan your nodes
2. Plan your relationships
3. Add node labels
4. Add relationship labels
5. Add properties

Pitfalls to avoid

It's important to make sure that your graph model **facilitates the types of queries** that you are interested in.



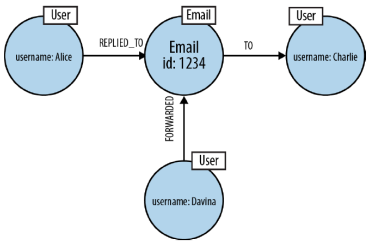
Bad



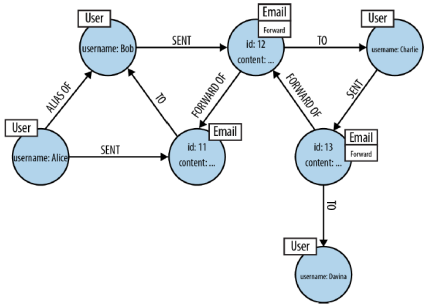
Good

Pitfalls to avoid

It's also important to **think ahead** and ensure that your graph can evolve with new types of nodes/relationships.

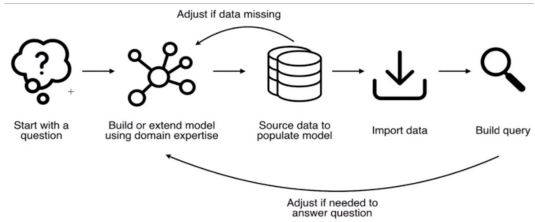


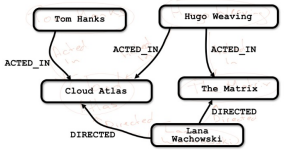
Bad

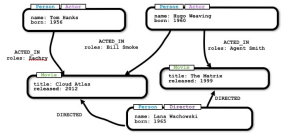


Good

Graph Data Modelling & Implementation







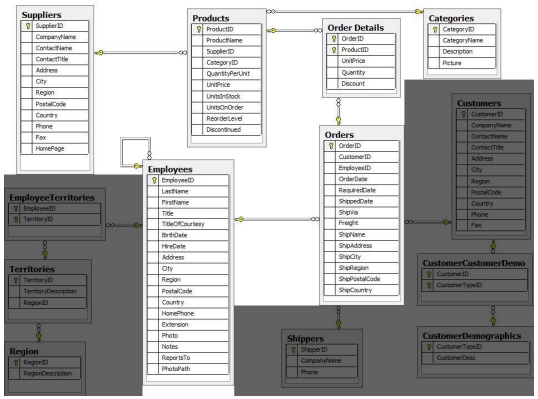
Arrows: Neo4j Graph Modelling Tool

https://neo4j.com/developer/guide-data-modeling/

Lecture Outline

- What is a Graph
- Graph Databases vs. Relational
- **Graph Data Modelling**
  - From Scratch
  - **From an existing relational DB**

The Popular Northwind Dataset for SQL



Northwind - a product sale system

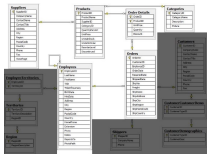
- customers,
- products,
- customer orders,
- warehouse stock,
- shipping,
- suppliers,
- employees, and
- sales territories.

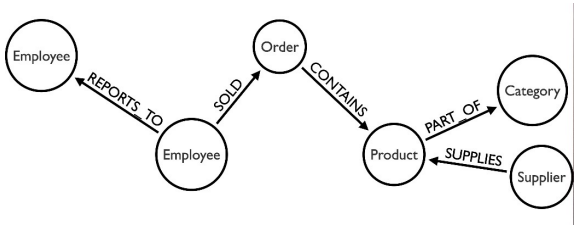
A *row* is a *node*
A *table name* is a *label name*
A *join or foreign key* is a *relationship*

https://neo4j.com/developer/guide-importing-data-and-etl/

Rows to Nodes, Table names to labels

Each row on the **Orders** table becomes a node in with *Order* as the label.
Each row on the **Products** table becomes a node with *Product* as the label.
Each row on the **Suppliers** table becomes a node with *Supplier* as the label.
Each row on the **Categories** table becomes a node with *Category* as the label.
Each row on the **Employees** table becomes a node with *Employee* as the label.

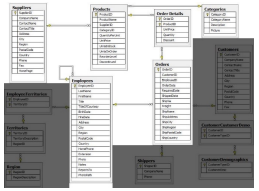


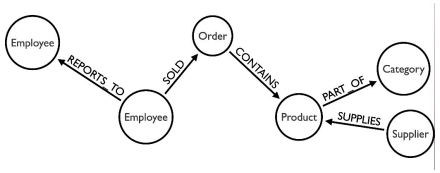


Schema-level graph

Rows to Nodes, Table names to labels

Join between **Suppliers** and **Products** becomes a relationship named SUPPLIES (where supplier supplies product).
Join between **Products** and **Categories** becomes a relationship named PART_OF (where product is part of a category).
Join between **Employees** and **Orders** becomes a relationship named SOLD (where employee sold an order).
Join between **Employees** and ***itself*** (unary relationship) becomes a relationship named REPORTS_TO (where employees have a manager).
Join with join table (Order Details) between **Orders** and **Products** becomes a relationship named CONTAINS with properties of `unitPrice`, `quantity`, and `discount` (where order contains a product).

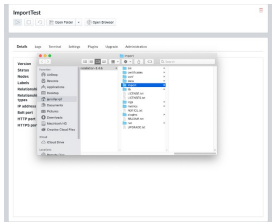




Importing the Data using Cypher

```
// Create orders
LOAD CSV WITH HEADERS FROM 'file:///orders.csv' AS row
MERGE (order:Order {orderID: row.OrderID})
  ON CREATE SET order.shipName = row.ShipName;

// Create products
LOAD CSV WITH HEADERS FROM 'file:///products.csv' AS row
MERGE (product:Product {productID: row.ProductID})
  ON CREATE SET product.productName = row.ProductName, product.unitPrice = toFloat(row.UnitPrice);

// Create suppliers
LOAD CSV WITH HEADERS FROM 'file:///suppliers.csv' AS row
MERGE (supplier:Supplier {supplierID: row.SupplierID})
  ON CREATE SET supplier.companyName = row.CompanyName;

// Create employees
LOAD CSV WITH HEADERS FROM 'file:///employees.csv' AS row
MERGE (e:Employee {employeeID:row.EmployeeID})
  ON CREATE SET e.firstName = row.FirstName, e.lastName = row.LastName, e.title = row.Title;

// Create categories
LOAD CSV WITH HEADERS FROM 'file:///categories.csv' AS row
MERGE (c:Category {categoryID: row.CategoryID})
  ON CREATE SET c.categoryName = row.CategoryName, c.description = row.Description;
```

With Neo4j Desktop, we can place CSV files in the local database import directory.
Use the file:/// prefix in our Cypher statements to locate the files.



https://neo4j.com/developer/desktop-csv-import/#csv-location

Summary

- We have shown how to model a graph database model using a whiteboard, and discussed some pitfalls when modelling a graph database.
- Next week we will cover Cypher in much more detail and show how to query the database.
- We will also look at the process of converting an existing CSV (tabular) dataset to a graph database.

Copyright Notice

THE UNIVERSITY OF WESTERN AUSTRALIA

Copyright Notice

Material used in this recording may have been reproduced and communicated to you by or on behalf of **The University of Western Australia** in accordance with section 113P of the *Copyright Act 1968*.

Unless stated otherwise, all teaching and learning materials provided to you by the University are protected under the Copyright Act and is for your personal use only. This material must not be shared or distributed without the permission of the University and the copyright owner/s.

THE UNIVERSITY OF WESTERN AUSTRALIA

THE UNIVERSITY OF WESTERN AUSTRALIA

Reading

THE UNIVERSITY OF WESTERN AUSTRALIA

1. What is knowledge graph: https://www.youtube.com/watch?v=y7sXDpffzQQ
2. Property graph: https://docs.oracle.com/en/database/oracle/property-graph/22.2/spgdg/what-are-property-graphs.html
3. Importing data from a relational database into Neo4j: https://neo4j.com/docs/getting-started/appendix/tutorials/guide-import-relational-and-etl/