UWE Bristol | University of the West of England

Presented by

Dr. Trupti Padiya

School of Computing and Creative Technologies

Advanced Databases UFCFU3-15-3

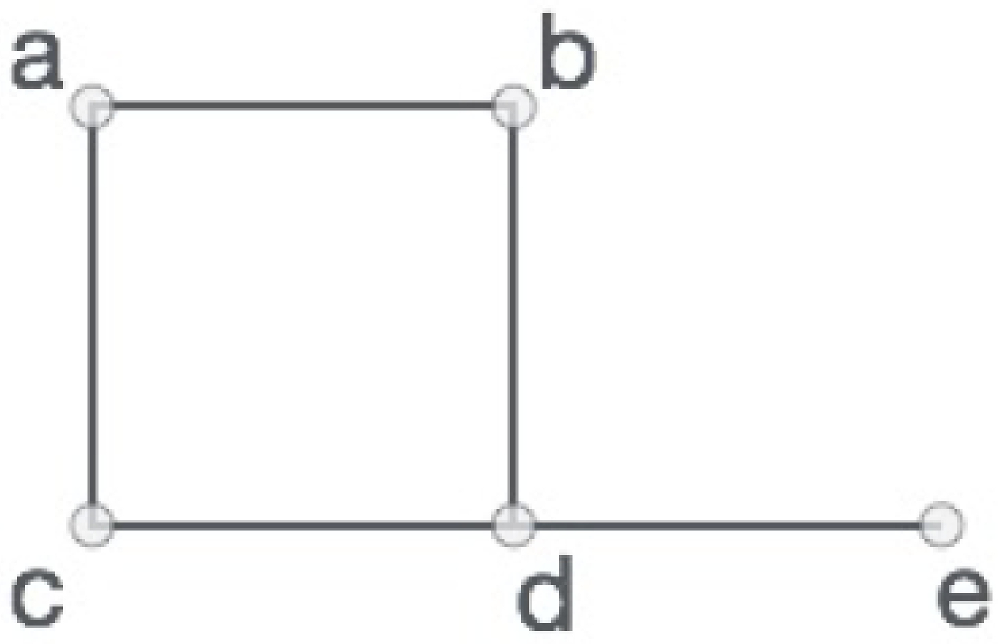
Introduction to Graph Databases

Slides with modification adapted from Dr. Elias Pimenidis

What is a graph?

- Formally, a graph is just a collection of vertices and edges—or, in less intimidating language, a set of nodes and the relationships that connect them.
- Graphs represent entities as nodes and the ways in which those entities relate to the world as relationships.
- This general-purpose, expressive structure allows us to model all kinds of scenarios, from the construction of a space rocket, to a system of roads, and from the supply chain, to medical history for populations, and beyond.

A very simple graph



Graph Databases

A graph database uses highly inter-linked data structures built from nodes, relationships, and properties. In turn, these graph structures support sophisticated, semantically rich queries at scale.

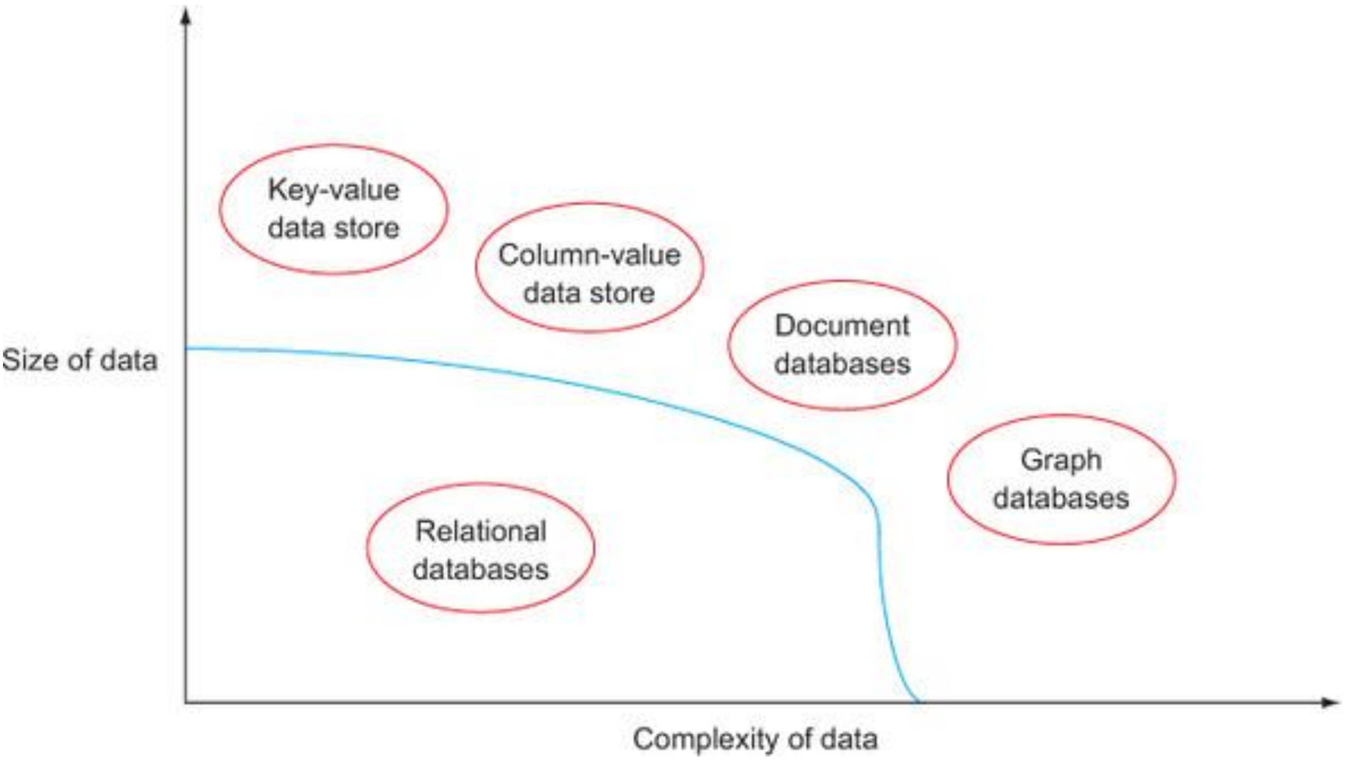
What are Graph Databases suitable for

The key to understanding when to use a graph database is the value of links.

If your data is connected, whether it supports an online mobile app or an offline machine learning framework, then a graph is going to be a good choice.

Conversely if your data is bulk storage, blob storage, time series, or logs, then a graph may not be the best choice because there aren’t many links between the data to exploit.

Graphs in the Database World



When relational databases can no longer cope with the complexity of a data set because of its connectedness, but not its size, graph databases may be your best option.

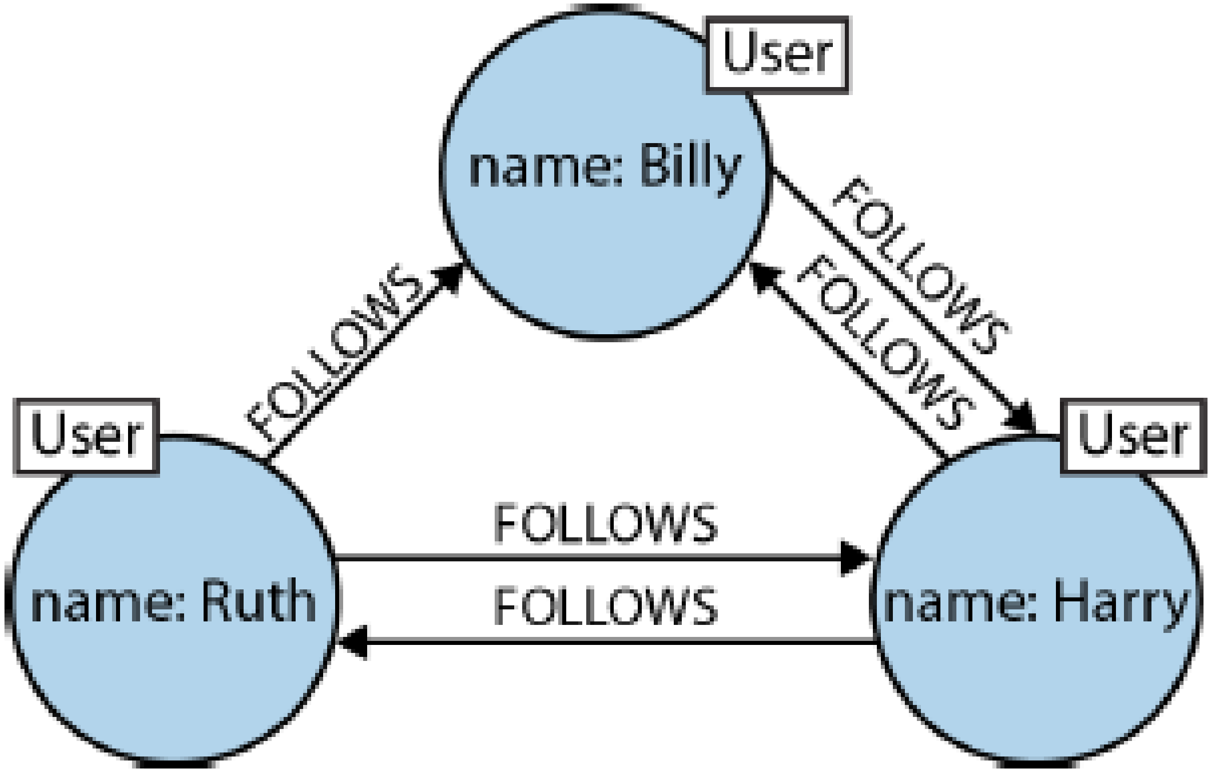
Graph Databases are Relationship focused

- Contrary to what the name of relational databases indicates, not much is relational about them except that the foreign keys and primary keys are what relate tables.
- In contrast, **relationships in graph databases are first-class citizens**. Through this aspect, they lend themselves well to modelling and querying **connected data**.

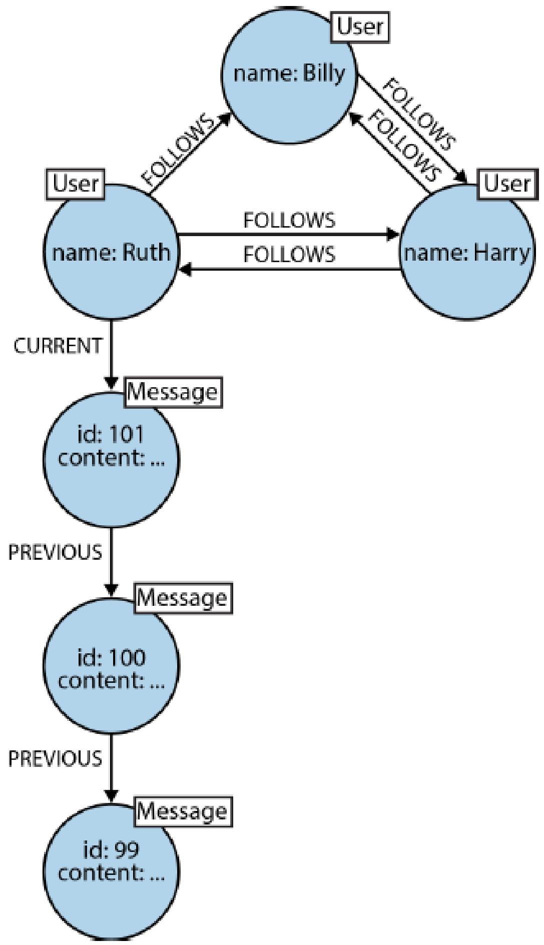
Graph DB Use cases

- Social network queries
  - E.g. Facebook stores the entire metadata in a social graph
- Network security
  - Find sequence of steps that lead to intrusion
- Fraud detection
  - Find fraud rings
- Knowledge bases
  - Answer questions, language models

A small social graph



A social graph with publishing messages



There are many kinds of graphs

- Directed or undirected?
- Labelled or unlabelled edges/nodes?
- What kinds of labels? Datatypes?
- Parallel edges (multi-graphs)? With same label?
- One graph or several graphs per database?

- Two types of graph database models dominate the market today: Resource Description Framework (RDF) and Property Graph

Resource Description Framework (RDF)

- RDF is a W3C standard for representing linked data on the Web
  - Directed labelled graph; nodes are identified by their labels
  - Labels are URIs or datatype literals
  - Multiple parallel edges only when using different edge labels
  - Supports multiple graphs in one database
  - W3C standard; implementations for many programming languages
  - Datatype support based on W3C XML Schema datatypes
  - Graphs can be exchanged in many standard syntax formats

Property Graph

- Property Graph is a popular data model of many graph databases
  - Directed labelled multi-graph; labels do not identify nodes
  - “Labels” can be lists of attribute-value pairs
  - Multiple parallel edges with the exact same labels are possible
  - No native multi-graph support (could be simulated with additional attributes)
  - No standard definition of technical details; most common implementation: Tinkerpop/Blueprints API (Java)
  - Datatype support varies by implementation
  - No standard syntax for exchanging data

Representing Graphs

Graphs (of any type) are usually viewed as sets of edges

- RDF: triples of form subject-predicate-object
  - When managing multiple graphs, each triple is extended with a fourth component (graph ID) { quads
  - RDF databases are sometimes still called "triple stores", although most modern systems effectively store quads
- Property Graph: edge objects with attribute lists
  - represented by Java objects in Blueprints

Graphs can be stored in relational databases

- RDF: table Triple[Subject,Predicate,Object]
- Property Graph: tables Edge[SourceId,EdgeId,TargetId] and Attributes[Id,Attribute,Value]

Representing Data in Graphs

Property Graphs can represent RDF:

- use attributes to store RDF node and edge labels (URIs)
  - The subject of an RDF statement is either a uniform resource identifier (URI) or a blank node, both of which denote resources. Resources indicated by blank nodes are called anonymous resources.
- use key constraints to ensure that no two distinct nodes can have same label

RDF can represent Property Graphs:

- use additional nodes to represent Property Graph edges
- use RDF triples with special predicates to represent attributes

Either model can also represent hypergraphs/RDBs (exercise)

- all models can represent all data in principle
- supported query features and performance will vary

Querying Graphs

- Preferred query language depends on graph model
- RDF: W3C SPARQL query language
- Property Graph: no uniform approach to data access
  - many tools prefer API access over a query language
  - proprietary query languages, e.g., “Cypher” for Neo4j
- However, there are some common basics in almost all cases:
  - Conjunctive queries
  - Regular path queries

Resource Description Framework

World Wide Web Consortium specification

- Used for the Semantic Web
- Web pages define human-readable content
- **Goal: add machine-readable meta-data describing how pages relate**
- Format to reuse and share data across the Web

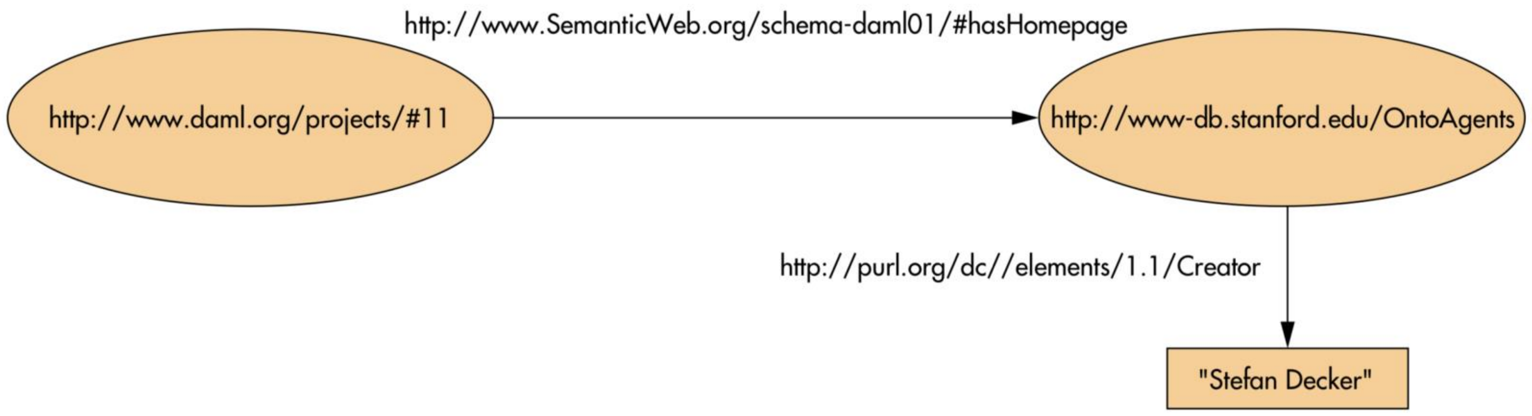
Examples

- Wikipedia, census, life sciences, DBPedia

Directed labelled multi-graph

RDF Format

- Graph is set of triplets = (Subject, Predicate, Object)
- • Subject and predicate are resources
- • Associated with Unique Resource Identifiers (URI)
- • Object can be resource or literal (string)


http://www.SemanticWeb.org/schema-daml01/#hasHomepage
http://www.daml.org/projects/#11
http://www-db.stanford.edu/OntoAgents
http://purl.org/dc//elements/1.1/Creator
"Stefan Decker"

Labelled Property Graphs

- The most widely used model for graph databases is the labelled property graph model. To experts, this shorthand is useful to distinguish between this model and other more mathematically inclined models, such as hypergraphs.
- But if you aren’t an expert, this description may need a little unpacking.
- The fundamental components of the labelled property graph model are nodes and relationships (you may also know these as vertices and edges) and constraints.
- In the labelled property graph model, we use naming conventions to distinguish elements at a glance.

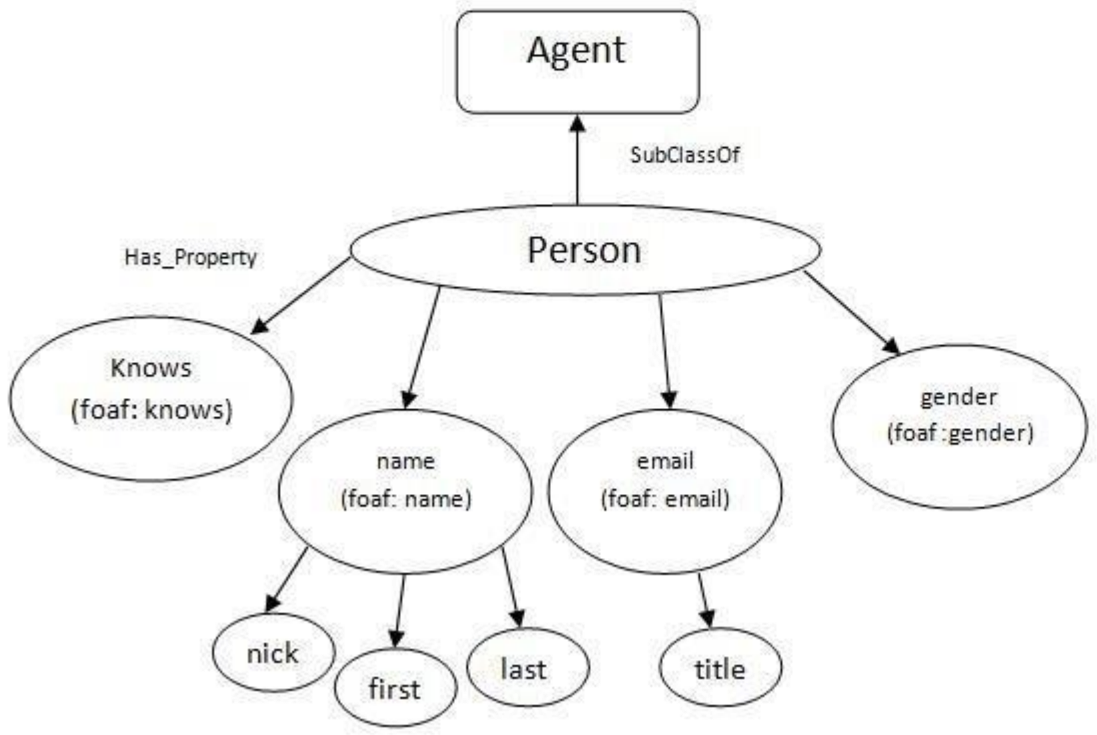
Semantic Relationships are hidden in Relational Databases

- Join tables add accidental complexity; they mix business data with foreign key metadata.
- Foreign key constraints add additional development and maintenance overhead just to make the database work.
- Sparse tables with nullable columns require special checking in code, despite the presence of a schema.
- Several expensive joins are needed just to discover what a customer bought.

NOSQL Databases Lack Relationships

- Most NOSQL databases store sets of disconnected documents/values/columns.
- One well-known strategy for adding relationships to such stores is to embed an aggregate’s identifier inside the field belonging to another aggregate—effectively introducing foreign keys.
- This requires joining aggregates at the application level, which quickly becomes prohibitively expensive.
- When we look at an aggregate store model, we imagine we can see relationships. Seeing a reference to order: 1234 in the record beginning user: Alice, we infer a connection between user: Alice and order: 1234. This gives us false hope that we can use keys and values to manage graphs.

Modelling a part of FOAF (Friend of a Friend)



Using Labels – Efficient Querying

- Using labels in this way, **we can group nodes**. We can ask the database, for example, to find all the nodes labelled Person. Labels also provide a hook for declaratively indexing nodes.
- Relationships in a graph naturally form paths. **Querying**—or traversing—the graph involves **following paths**.
- Because of the fundamentally path-oriented nature of the data model, the majority of path-based graph database operations are highly aligned with the way in which the data is laid out, **making them extremely efficient.**

Some Extra resources

- https://neo4j.com/docs/getting-started/graph-database/
- https://www.datacamp.com/blog/what-is-a-graph-database
- https://neo4j.com/blog/knowledge-graph/rdf-vs-property-graphs-knowledge-graphs/
- https://www.ontotext.com/knowledgehub/fundamentals/rdf-vs-property-graphs/