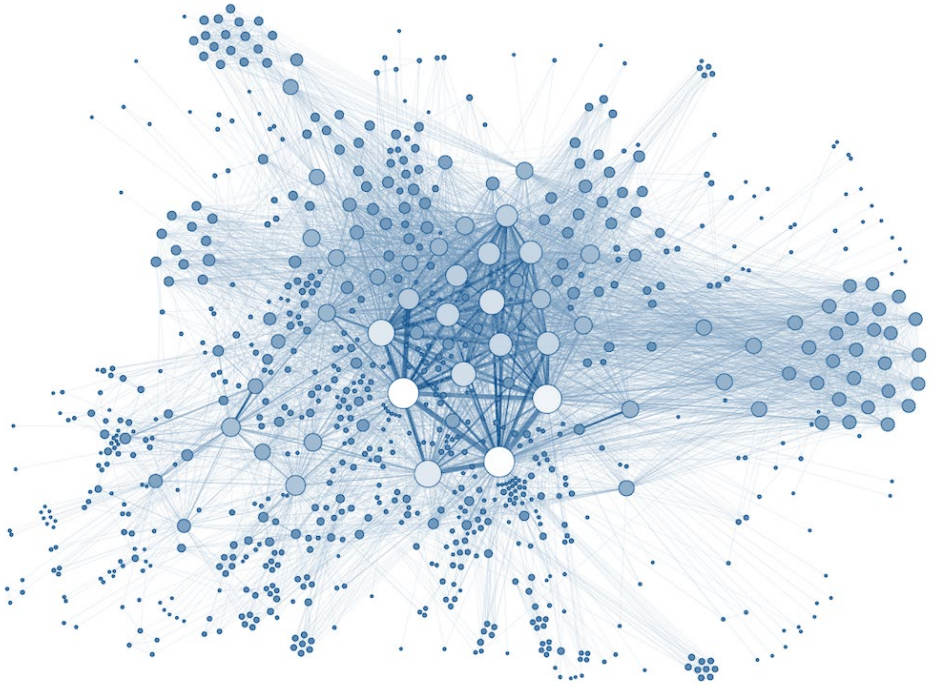
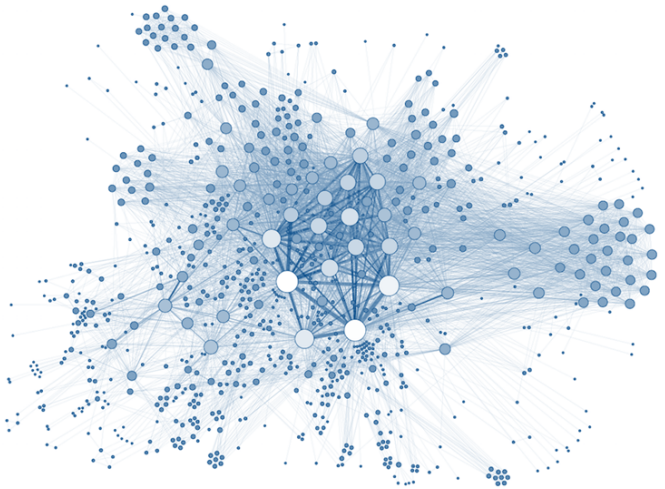
Graph Database



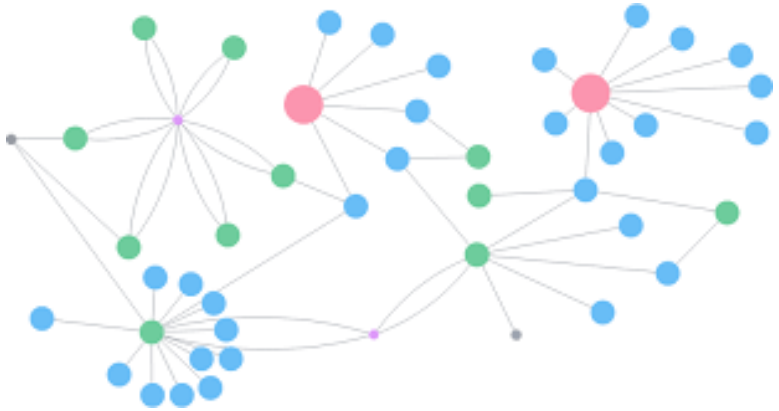
Content

- Why?
- Graph
- LPG vs RDF



Content

- Why?
- Graph
- LPG vs RDF



Why?

What is a Graph Database?

A graph database is an online database management system with Create, Read, Update and Delete (CRUD) operations working on a graph data model.

- Unlike other databases, relationships take first priority in graph databases. This means applications don't have to infer data connections using things like foreign keys or out-of-band processing, such as MapReduce.
- The data model for a graph database is also significantly simpler and more expressive than those of relational or other NoSQL databases.
- Graph databases are built for use with transactional (OLTP) systems and are engineered with transactional integrity and operational availability in mind.

Why?

Concept

Nodes and **Edges**

- Each node represents an entity (a person, place, thing, category or other piece of data)
- Each edge represents how two nodes are associated.

This general-purpose structure allows to model all kinds of scenarios – from a system of roads, to a network of devices, to a population's medical history or **anything else defined by relationships**.

Graph Storage

Some graph databases use native graph storage that is specifically designed to store and manage graphs, while others use relational or document-oriented databases instead. Non-native storage is often much more latent.

Graph Processing Engine

Native graph processing (a.k.a. "index-free adjacency") is the most efficient means of processing graph data since connected nodes physically "point" to each other in the database. Non-native graph processing uses other means to process CRUD operations.

Why?

Concept

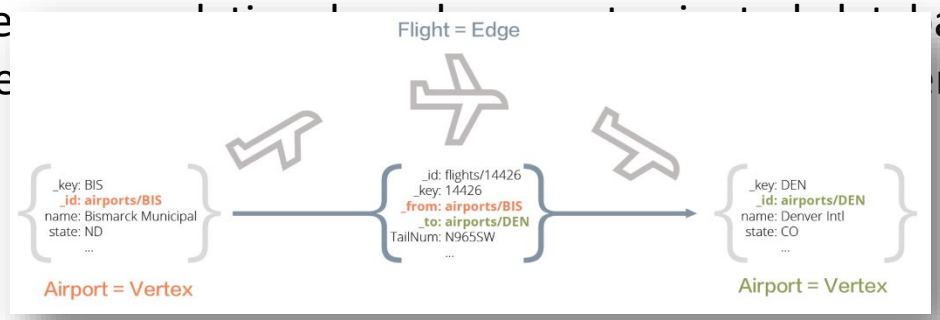
Nodes and **Edges**

- Each node represents an entity (a person, place, thing, category or other piece of data)
- Each edge represents how two nodes are associated.

This general-purpose structure allows to model all kinds of scenarios – from a system of roads, to a network of devices, to a population's medical history or **anything else defined by relationships**.

Graph Storage

Some graph databases use native graph storage that is specifically designed to store and manage graphs, while othe ases inste nt.



Graph Processing Engine

Native graph processing (a.k.a. "index-free adjacency") is the most efficient means of processing graph data since connected nodes physically "point" to each other in the database. Non-native graph processing uses other means to process CRUD operations.

Why?

Advantages

Today's CIOs and CTOs don't just need to manage larger volumes of data – they need to generate insight from their existing data. In this case, the **relationships between data points matter more than the individual points** themselves.

Performance

For intensive data relationship handling, graph databases improve performance by several orders of magnitude. With traditional databases, relationship queries will come to a grinding halt as the number and depth of relationships increase. In contrast, graph database performance stays constant even as your data grows year over year.

Flexibility

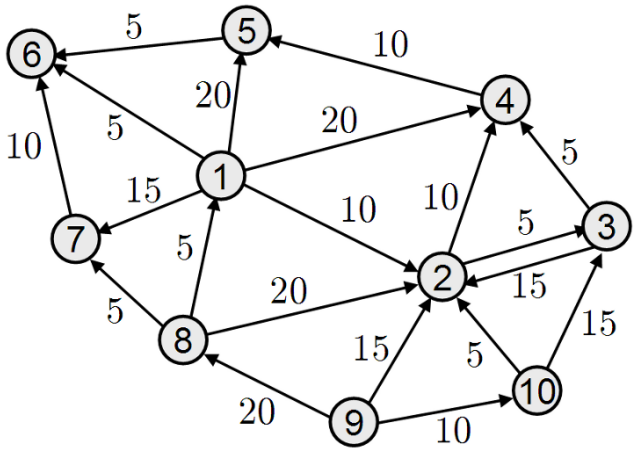
With graph databases, IT and data architect teams move at the speed of business because the structure and schema of a graph model flexes as applications and industries change. Rather than exhaustively modeling a domain ahead of time, data teams can add to the existing graph structure without endangering current functionality.

Agility

Developing with graph databases aligns perfectly with today's agile, test-driven development practices, allowing your graph database to evolve in step with the rest of the application and any changing business requirements. Modern graph databases are equipped for frictionless development and graceful systems maintenance.

Content

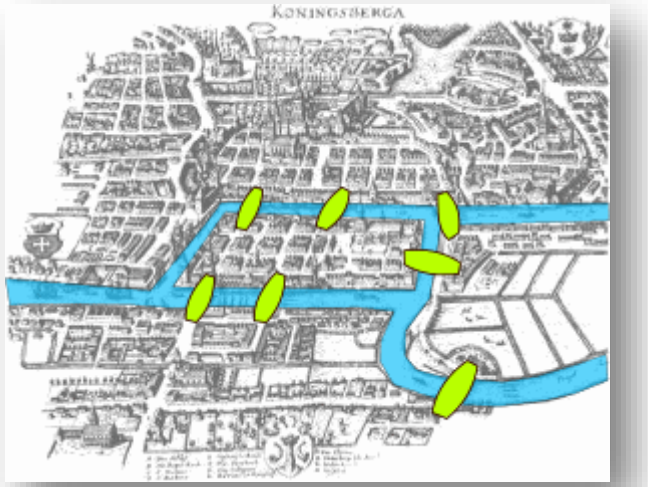
- Why?
- Graph
- LPG vs RDF



Graph

Graph-Theory

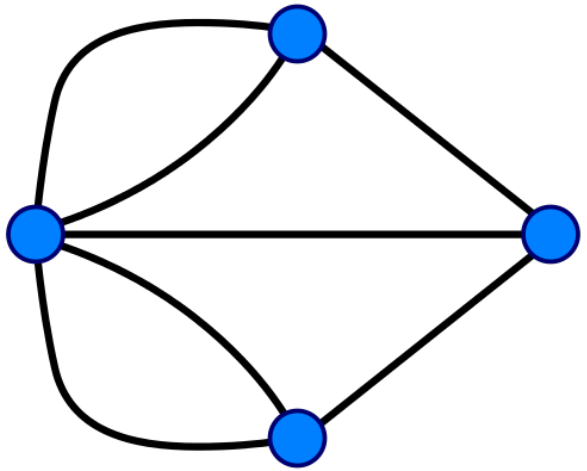
The paper written by Leonhard Euler on *the Seven Bridges of Königsberg* and published in 1736 is regarded as the first paper in the history of graph theory.



Graphs are mathematical structures used to **model pairwise relations** between objects.

A graph in this context is made up of **nodes** (vertices, points) which are connected by **edges** (arcs, lines).

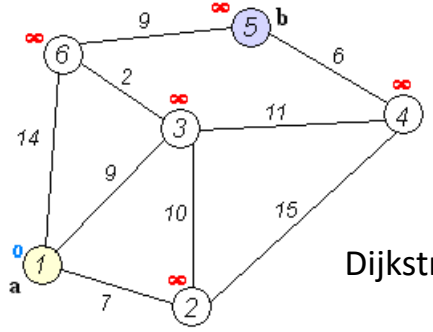
A graph may be **undirected**, meaning that there is no distinction between the two nodes associated with each edge, or its edges may be **directed** from one node to another



Graph

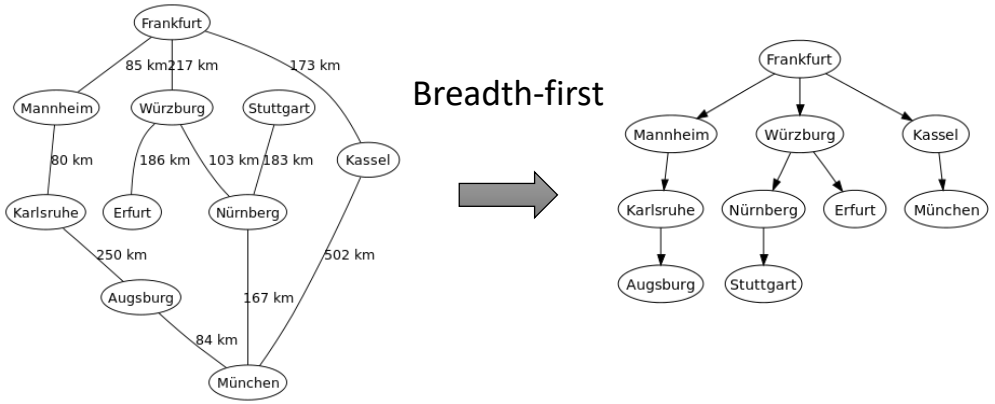
Paths

When using the graph model, many problems are solved by finding a set of possible paths or the shortest path between two single nodes or two node groups (subgraphs).



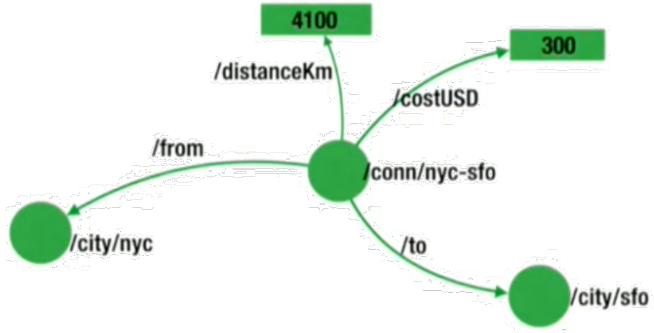
Dijkstra’s algorithm

- *Dijkstra's algorithm* | *Bellman–Ford algorithm*
  Algorithms to find the shortest path(s) between nodes
- *Depth-first search* | *Breadth-first search*
  Algorithms to travers/search a whole graph
- *Nearest Neighbour algorithm*
  Algorithm to find a path that visits all nodes in a graph



Content

- Why?
- Graph
- LPG vs RDF



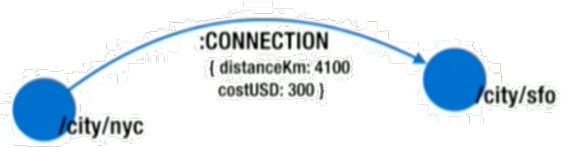
LPG vs RDF

Labeled Property Graph (LPG)

- developed for efficient storage
- for fast querying connected data
- similar to relational databases

- **Vertices**
  Nodes: ID + a set of key-value pairs

- **Edges**
  Relationships: ID + Type + set of key-value pairs

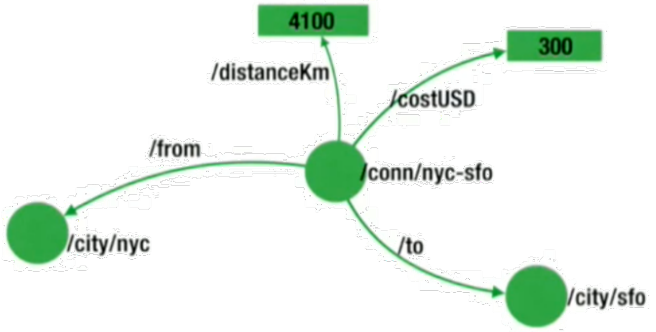


Resource Description Framework (RDF)

- W3C standard for data exchange in the Web
- exchange model that represents data as a graph
- semantic graph database → triple stores

- **Vertices**
  Resources: URIs
  Attribute Values: Literal Values

- **Edges**
  Relationships: URIs



LPG vs RDF

Example

Connection:

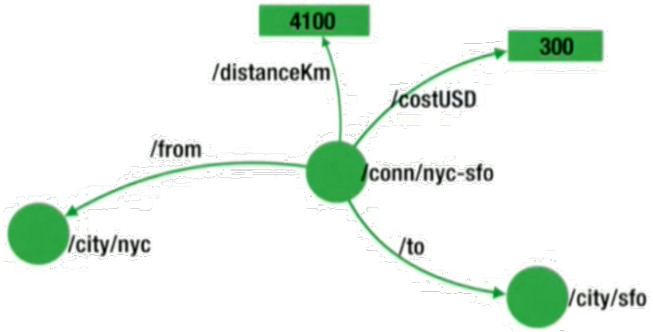
- From : New York City
- To : San Francisco
- Distance : 4100km
- Cost : 300 USD

Main difference

- LPG - Vertices and Edges have internal structure
- RDF - Vertices or Edges have NO internal structure

LPG

RDF

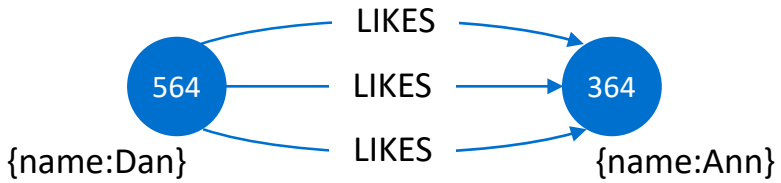


LPG vs RDF

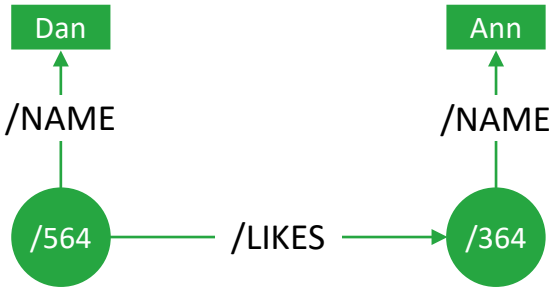
One other important difference…

➞ RDF does not uniquely identify instances of relationships of the same type

LPG



RDF



It is possible to have connections of the same type between the same pair of nodes.

It is not possible to have connections of the same type between the same pair of nodes because that would represent exactly the same triple, with no extra information