Graph Data Models and Applications

Oscar Romero
Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

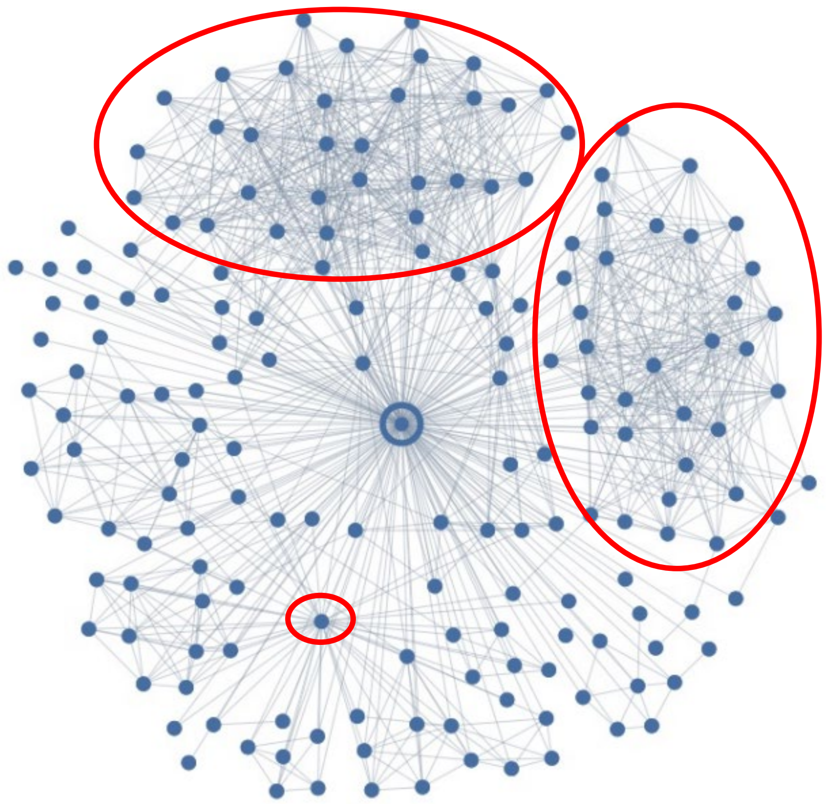
Graph Data Models and Data Analytics

- From a data management point of view:
  - Their extremely flexible
  - Schemaless by definition
  - Facilitate data governance
  - Facilitate ad-hoc transformations

- From a data analytics point of view:
  - Allow to exploit the data structure topology
  - Sits somewhere in between descriptive and probabilistic data analysis

Showcasing Graphs

- Crossing data from social networks it is possible to identify a graph like the one that follows:
  - In the centre there is a specific person *P*
  - The rest are *P* connections and connections among them
- Using sociology techniques…
  - We can identify *P social foci*:
    - Dense clusters of friends corresponding to long periods of interaction
    - Typically, college friends, coworkers, relatives, etc.
  - The *significant other* can be identified by a high *dispersion* rate
    - Highly connected with *P* connections,
    - But with a high dispersion degree wrt *P* social foci
- **Hypothesis**: when the node with higher dispersion degree Identified is not the partner, this couple is likely to split up in a period of 60 days
- L. Backstrom, J. Kleinberg. Romantic Partnerships and the Dispersion of Social Ties: A Network Analysis of Relationship Status on Facebook https://arxiv.org/pdf/1310.6753v1.pdf



GRAPH APPLICATIONS

Extracted from:
http://www.vldb.org/pvldb/vol11/p420-sahu.pdf

Entities Represented

- Humans: e.g., employees, customers, and their interactions
- Non-Human Entities: e.g., products, transactions, or web pages
  - Products: e.g., products, orders, and transactions
  - Business and Financial Data: e.g., business assets, funds, or bitcoin transfers
  - Web Data
  - Geographic Maps: e.g., roads, bicycle sharing stations, or scenic spots
  - Digital Data: e.g., files and folders or videos and captions
  - Infrastructure Networks: e.g., oil wells and pipes or wireless sensor networks
  - Knowledge and Textual Data: e.g., keywords, lexicon terms, words, and definitions.
- RDF or Semantic Web
- Scientific: e.g., chemical molecules or biological proteins

Graph Computations

Computation	Total	R	P	A
Finding Connected Components	55	18	37	12
Neighborhood Queries (e.g., finding 2-degree neighbors of a vertex)	51	19	32	3
Finding Short / Shortest Paths	43	18	25	17
Subgraph Matching (e.g., finding all diamond patterns, SPARQL)	33	14	19	21
Ranking & Centrality Scores (e.g., PageRank, Betweenness Centrality)	32	17	15	22
Aggregations (e.g., counting the number of triangles)	30	10	20	7
Reachability Queries (e.g.,checking if u is reachable from v)	27	7	20	3
Graph Partitioning	25	13	12	5
Node-similarity (e.g., SimRank)	18	7	11	3
Finding Frequent or Densest Subgraphs	11	7	4	2
Computing Minimum Spanning Tree	9	5	4	2
Graph Coloring	7	3	4	3
Diameter Estimation	5	2	3	2

Legend
R: Researchers
P: Practitioners
A: Academic publications

Machine Learning on Graphs

- Through Node / Edge embeddings

(a) Machine learning computations.

Computation	Total	R	P	A
Clustering	42	22	20	15
Classification	28	10	18	2
Regression (Linear / Logistic)	11	5	6	2
Graphical Model Inference	10	5	5	2
Collaborative Filtering	9	4	5	2
Stochastic Gradient Descent	4	2	2	3
Alternating Least Squares	0	0	0	2

(b) Problems solved by machine learning algorithms.

Computation	Total	R	P	A
Community Detection	31	15	16	5
Recommendation System	26	10	16	2
Link Prediction	25	10	15	2
Influence Maximization	14	5	9	2

Traversal	Total	R	P
Breadth-first-search or variant	19	5	14
Depth-first-search or variant	12	4	8
Both	22	8	14
Neither	20	11	9

Graph Traversals Performed

POPULAR USE CASES

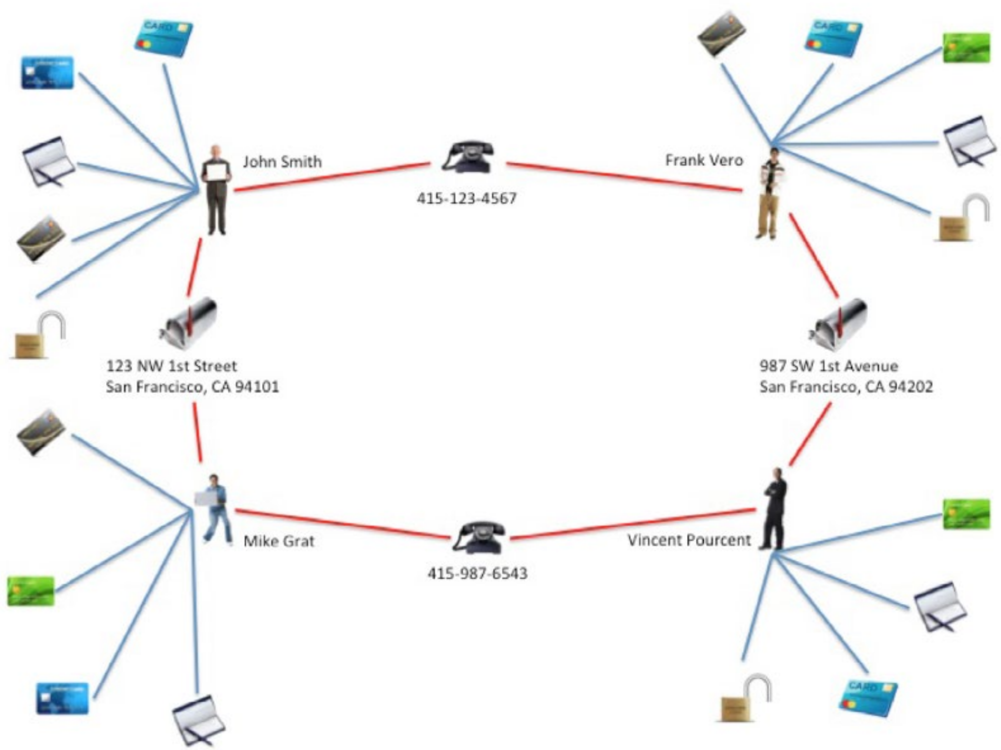
Fraud Detection: Banking

- Individuals / organisations asking for loans without any intention of paying them back
- Typical scenario
  1. A group of two or more people organize into a fraud ring
  2. The ring shares a subset of legitimate contact information, for example phone numbers and addresses, combining them to create a number of synthetic identities
  3. Ring members open accounts using these synthetic identities
  4. New accounts are added to the original ones: unsecured credit lines, credit cards, overdraft protection, personal loans, etc.
  5. The accounts are used normally, with regular purchases and timely payments
  6. Banks increase the revolving credit lines over time, due to the observed responsible credit behavior
  7. One day the ring “busts out”, coordinating their activity, maxing out all of their credit lines, and disappearing
  8. Sometimes fraudsters will go a step further and bring all of their balances to zero using fake checks immediately before the prior step, doubling the damage
  9. Collections processes ensue, but agents are never able to reach the fraudster
  10. The uncollectible debt is written off

White paper from Neo4J:
https://neo4j.com/use-cases/

Example

- 1. John Smith lives at 123 NW 1st Street, San Francisco, CA 94101 (his real address) and gets a prepaid phone at 415-123-4567
- 2. Vincent Pourcet lives at 987 SW 1st Ave, San Francisco, CA 94102 (his real address) and gets a prepaid phone at 415-987-65

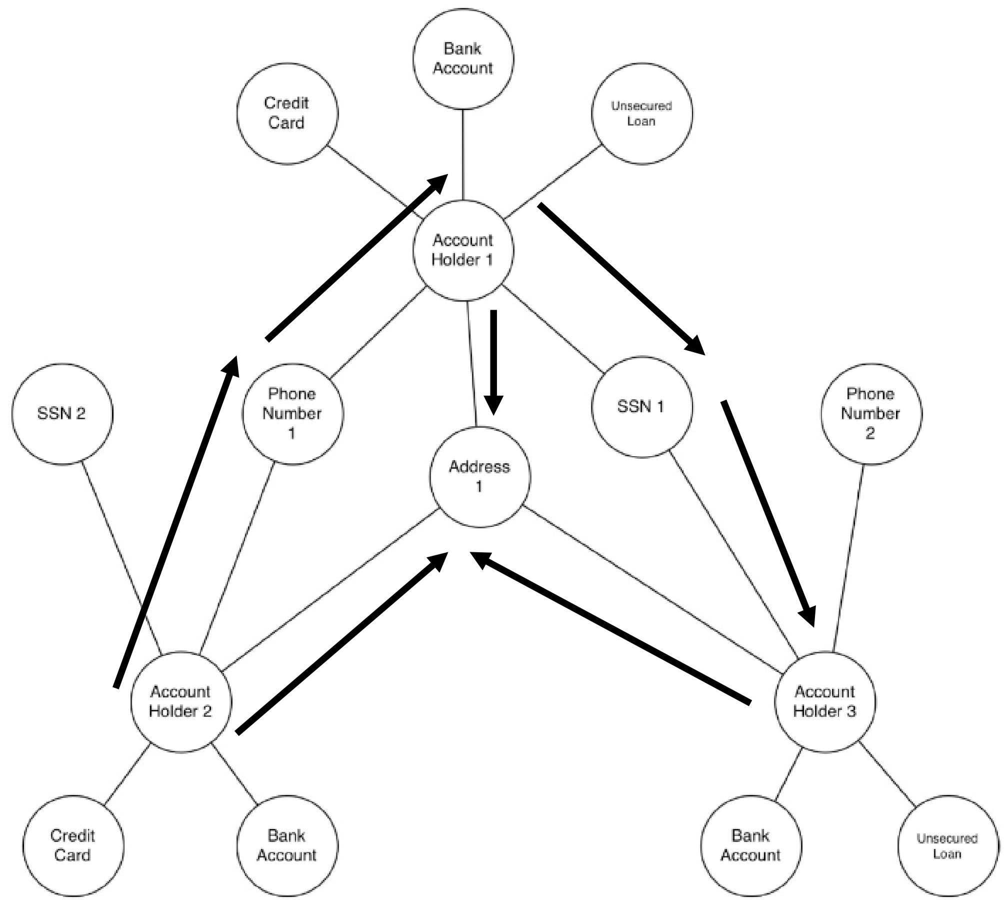


Entity Link Analysis

- In fraud detection, such fraud ring analysis is called *entity link analysis*
- A ring of n people (n≥2) sharing m elements of data (such as name, date of birth, phone number, address, SSN, etc.) can create up to n^m synthetic identities, where each synthetic identity is represented as a node and it is linked to $m \times (n-1)$ other nodes, for a total of $(n^m \times m \times (n-1)) / 2$ relationships
- Relational databases cannot compute such amount of combinations (as they are translated as joins and self-joins operations)
- In graph databases, it boils down to a graph pattern query

Graph Data Model

- Pattern matching:



Recommendations

- Graph databases have democratised recommendations
- The graph database naturally represents:
  - (customers, products, categories) and the relationships between them. E.g.,
    - Who bought what,
    - who “likes” whom,
    - which purchase happened first
- Note both explicit and implicit data can be asserted in the graph database

Recommendations

- Rationale
  - With labels one can categorise products and people
  - By analysing the relationships between products and people one can infer the interests of a person
  - We can weight a product or category likelihood for a given person or person category
- We are not using expensive mining algorithms but deterministic queries!

Natural Language Processing

- Sentences as basic construct

  (:my) – (:cat) – (:eats) – (:fish)

  - But a node can also represent a paragraph or even just a word
- Natural language processing
  - With NLP techniques, a label is attached
    - Verb, noun, adjective, etc.
  - Context can be computed for each word by means of Jaccard or similar indexes
  - Weight words relevance for keyword extraction (e.g., PageRank)

GRAPH DATA MODELS

Graph Data Model in a Nutshell

- Occurrence-oriented
  - It is a schemaless data model
    - There is no explicit schema
    - Data (and its relationships) may quickly vary
  - Objects and relationships as first-class citizens
    - *An object o relates (through a relationship r) to another object o'*
      - *Such relationship is often known as a triple (o r o')*
    - Both objects and relationships may contain properties
  - Built on top of the graph theory
    - Euler (18th century)
    - More natural and intuitive than the relational model to deal with relationships

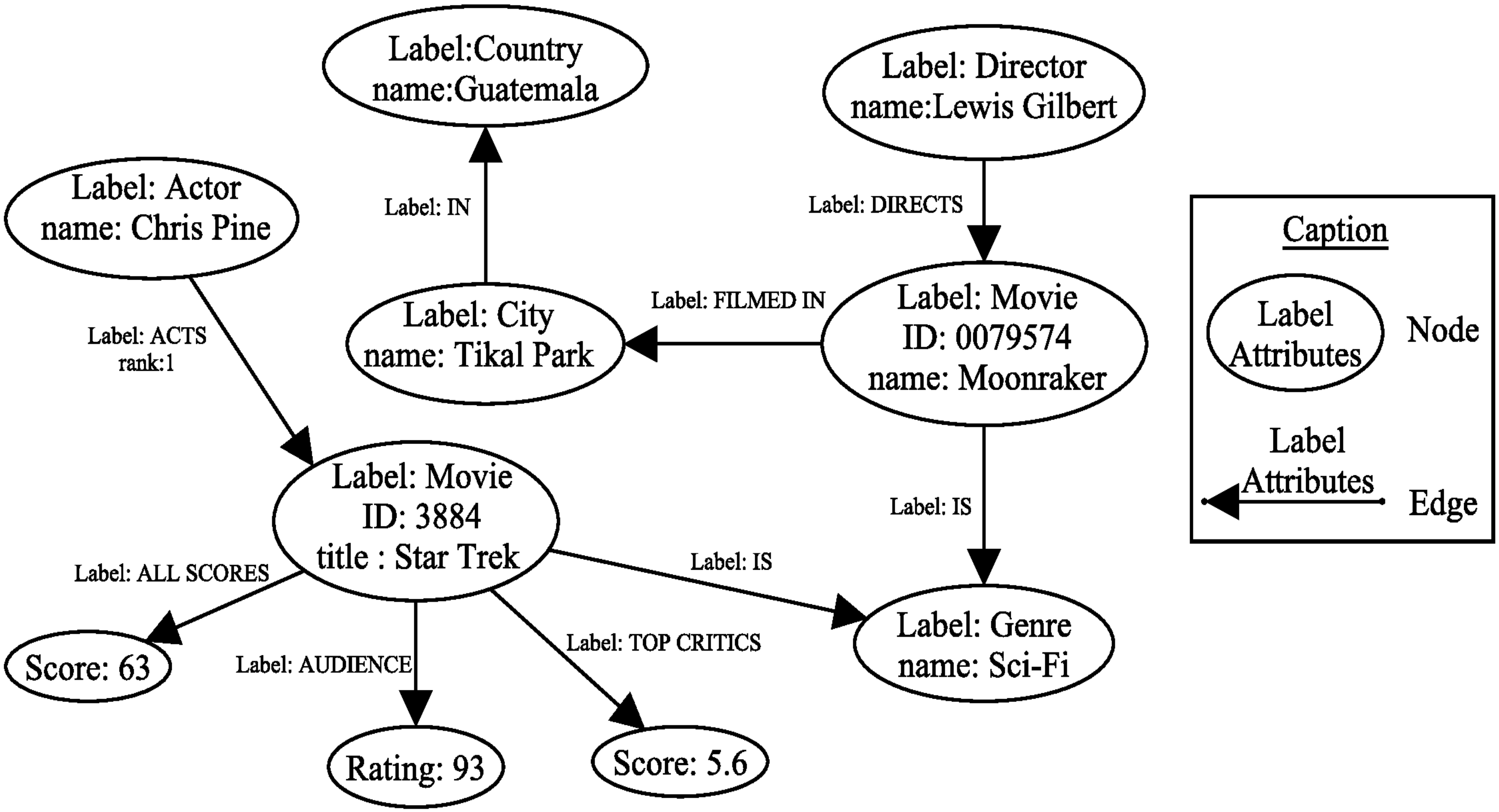
Notation (I)

- A **graph** *G* is a set of nodes and edges: *G (N, E)*
- *N* - **Nodes** (or vertices): n_1, n_2, … N_m
- *E* - **Edges** are represented as pairs of nodes: (n_1, n_2)
  - An edge is said to be **incident** to n_1 and n_2
  - Also, n_1 and n_2 are said to be **adjacent**
  - An edge is drawn as a line between n_1 *and* n_2
  - **Directed edges** entail direction: *from* n_1 *to* n_2
  - An edge is said to be **multiple** if there is another edge exactly relating the same nodes
  - An **hyperedge** is an edge inciding in more than 2 nodes.
- **Multigraph**: If it contains at least one multiple edge.
- **Simple graph**: If it does not contain multiple edges.
- **Hypergraph**: A graph allowing hyperedges.

Notation (II)

- **Size** (of a graph): #edges
- **Degree** (of a node): #(incident edges)
  - The degree of a node denotes the node adjacency
  - The neighbourhood of a node are all its adjacent nodes
- **Out-degree** (of a node): #(edges leaving the node)
  - Sink node: A node with 0 out-degree
- **In-degree** (of a node): #(incoming edges reaching the node)
  - Source node: A node with 0 in-degree
- Cliques and trees are specific kinds of graphs
  - **Clique**: Every node is adjacent to every other node
  - **Tree**: A connected acyclic simple graph

Example



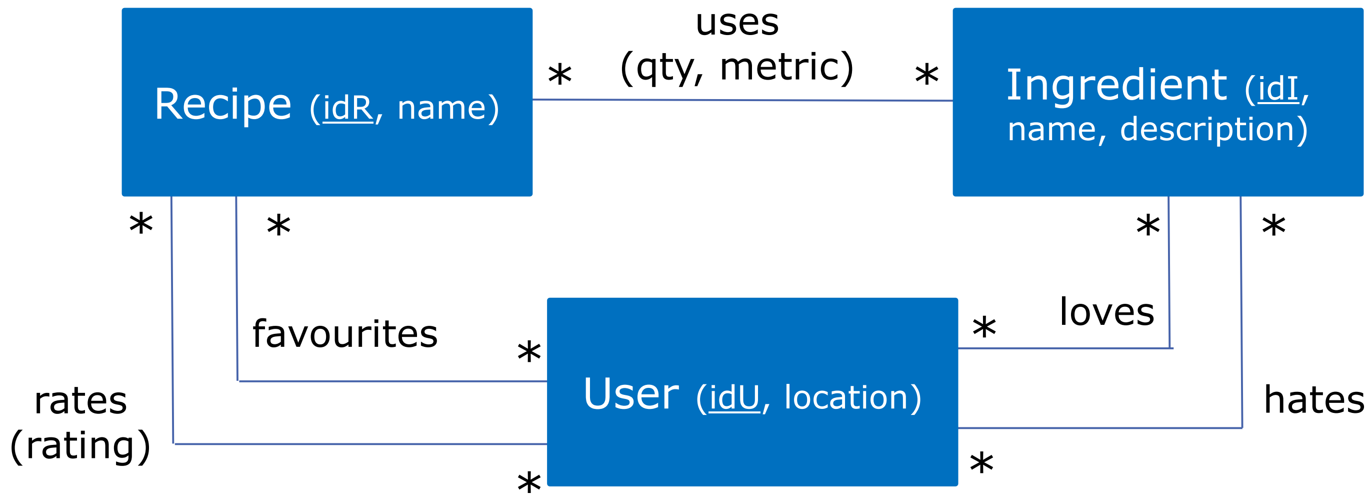
Label:Country
name:Guatemala
Label: Director
name:Lewis Gilbert
Label: Actor
name: Chris Pine
Label: IN
Label: DIRECTS
Label: ACTS
rank:1
Label: City
name: Tikal Park
Label: FILMED IN
Label: Movie
ID: 0079574
name: Moonraker
Caption
Label
Attributes
Node
Label
Attributes
Edge
Label: Movie
ID: 3884
title : Star Trek
Label: IS
Label: IS
Label: ALL SCORES
Label: AUDIENCE
Label: TOP CRITICS
Score: 63
Label: Genre
name: Sci-Fi
Rating: 93
Score: 5.6

Pros and Cons of Graphs

COMPARISON WITH OTHER DATA MODELS

Activity: Comparison with other Data Models

- (5') Refresh the main data models
  - (10') Consider the UML class diagram below:
    - Propose a relational **and** a graph schema as optimised as possible to deal with the following queries:
      - For a given recipe, give me all the users that favorited it
      - For a given recipe, the list of ingredients it contains with their quantity and metric
      - For a given ingredient, how many users do hate it
      - For a given ingredient, all the recipes each participate
      - For a given user, all the ingredients he loves
      - List all ingredients of a recipe rated above 3 for all users in BCN
    - What are the pros and cons of each data model?



Pros and Cons

Graphs

- They are occurrence-oriented
- **Occurrences** are **pointed by / point to** related occurrences
  - Query operators do not rely on schema
  - Naturally facilitate data linking
- The schema information is embedded together with data
  - The concept of stand-alone catalog does not exit
- Purely schemaless
  - Semantics are fixed by the edge / node labels
- Difficult to benefit from sequential access. Typically, it relies on random accesses
- By definition, it follows an Open-World assumption (i.e., assumes incomplete data)

Key-oriented Models

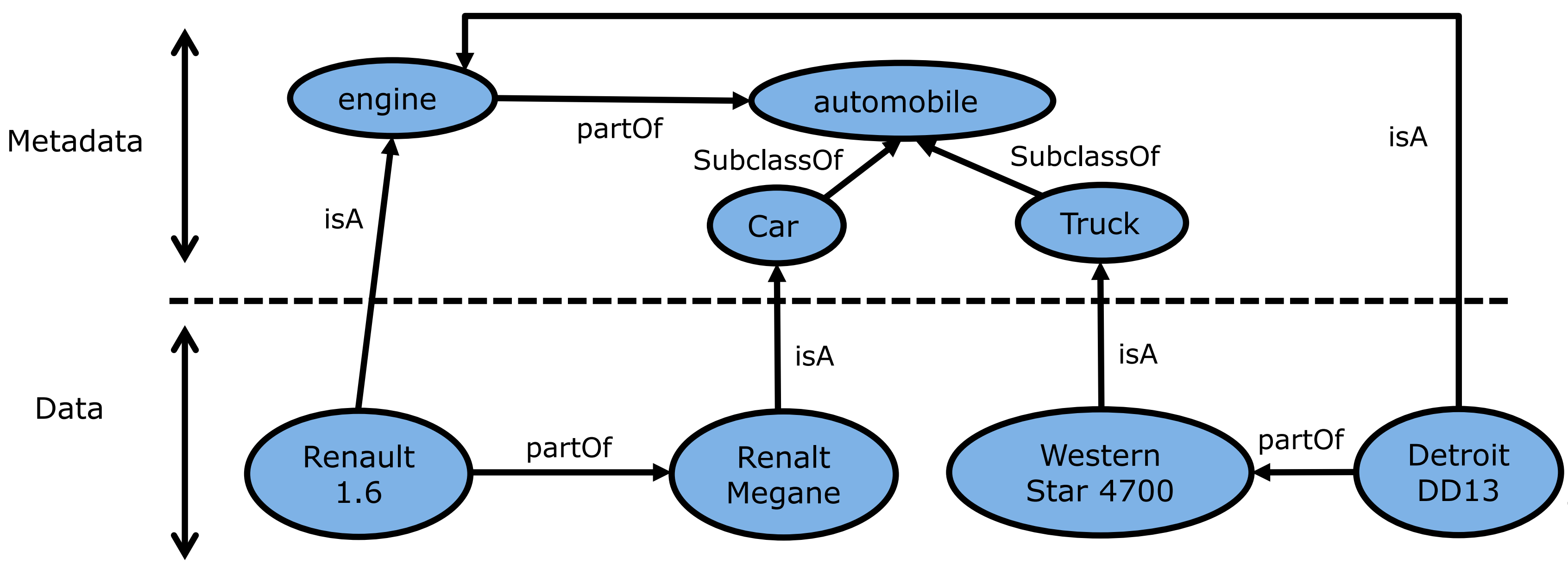
- The relational model is schema-oriented. Document-stores and key-values are schemaless databases but still rely on key-based structures
- Key-oriented models need to make a strong modeling call, which unbalances the logical / physical model
  - As consequence, the degree of (de)normalisation has a big impact in queries
- Can naturally benefit from sequencial reads
- Views are either virtual definitions or, if materialised, additional stand-alone constructs
- Poor relationship semantics: the relational model only deals with FK, document-stores / key-values do not support relationships
- Relational model, and most key-value / document-stores, follow a Closed-World assumption (i.e., complete data)

The extreme modeling of graphs

GRAPHS AS CANONICAL DATA MODELS FOR INTEGRATION

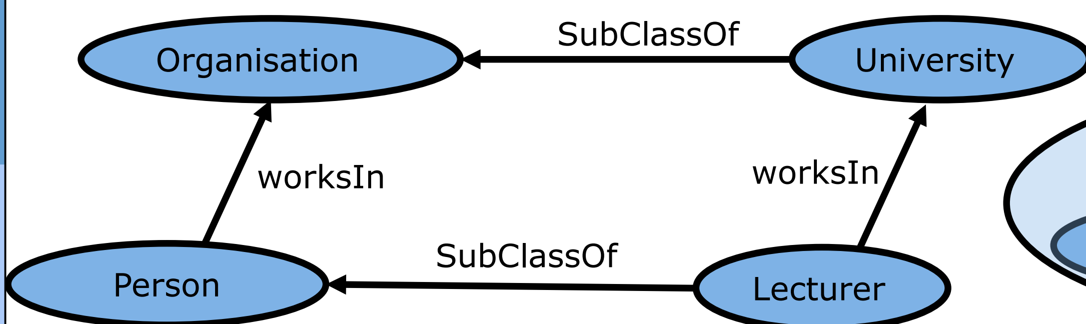
Graphs As Canonical Data Model (I)

- Expressiveness
  - Structural expressiveness
    - Relational data model: concepts / instances, referential integrity constraint
    - Graph data model: being a purely schemaless database, labels might embed any desirable semantics



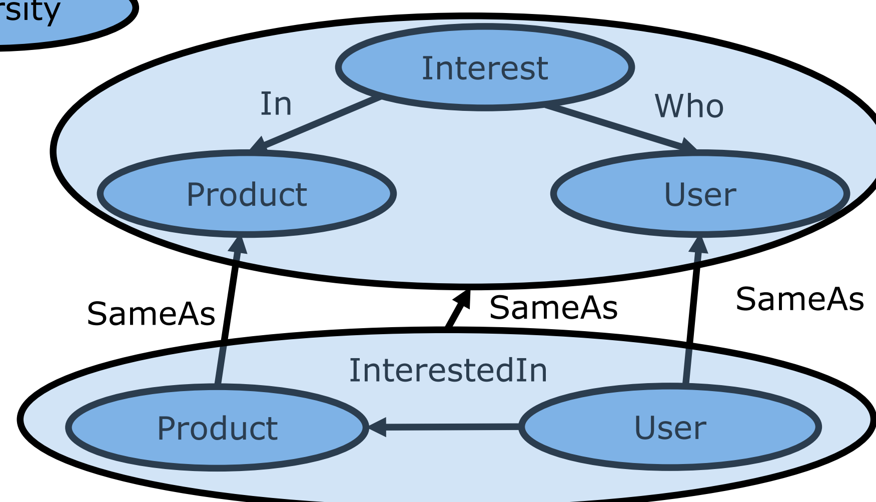
Graphs As Canonical Data Model (II)

- Semantic Relativeness
  - Relational Model: Monolithic
    - The model semantics are fixed. Table, columns and datatypes pre-defined at design time
    - Evolution not well-handled. Adding / deleting a column or changing a datatype may have a huge impact at the physical level
    - A powerful algebra available: the relational algebra
  - Graph Model: Flexible
    - New concepts / semantics can be added at any moment without drastically impacting the current data structures
    - Its flexibility allows to deal with evolution as first-class citizen
    - Powerful algebras available: For example, GraphQL is reducible to the relational algebra. In addition, other relevant operations not naturally expressable on top of the relational model (e.g., pattern matching)



How to do this in relational?

But graphs are even more flexible than that



Activity: The Graph Data Model

- Objective: Understand the graph data model
- Tasks:
  1. (10’) With a teammate think of the following:
     I. Assume graphs as canonical data model
     II. First, model as graphs each source (separately):
        I. Model schema and at least one instance for each

- User
- Tweet
- Date
- Location

- Product
- Product features

- User
- Product
- Landing time
- #visits

     III. Now, relate elements from each graphs with new edges generating a unique connected graph
        I. Look for similar or identical concepts
        II. Think of interesting relationships you could exploit later
     IV. Now, carefully check the resulting graphs. Can you identify what is data and metadata?

Properties of the Graph Data Model

- Its semantic relativeness allow to represent any other data model
  - Highly expressive data structure
    - Nodes and edges enough to represent any modeling construct
      - Two basic structures
    - Allows to deal with semantic conflicts
      - Arbitrary semantics embedded in the edges
      - N-ary relationships can be represented by hypergraphs
  - Rich algebra
    - Mappable to the relational algebra plus topology-oriented operations to manipulate graph structures
- Arbitrary semantic annotations
  - Its structural and behavioural expressiveness allow a wide range of annotations
    - Distinguish classes / instances
    - Express rich relationships
    - Arbitrary constraints

Graph Data Models

- There is not a single graph data model
- Two main families of graphs
  - **<u>Property Graphs</u>**
    - Born in the database field
    - Not predefined semantics
    - Follow a Closed-World assumption
    - Generate data silos
    - Algebraic operations based on graph structures
  - **<u>Knowledge Graphs</u>**
    - Born in the knowledge representation field
    - Assume the Open-World assumption
    - Facilitate data sharing and linking
    - Two main families
      - RDF and RDF(S)
        - Born in the semantic web field
        - Vocabulary-based pre-defined semantics
        - Combine algebraic operations with simple reasoning operations
      - Description Logics (DL)-based
        - Representation of (subsets of) first-order logic
        - Pre-defined semantics based on logics
        - Reasoning operations founded in their logics nature

Summary

- Graphs are the perfect canonical data model given their:
  - Semantic expressiveness,
  - Semantic relativeness
- As result, data and metadata (semantic annotations on data) are stored together
  - Machine-readable metadata opens the door to automatic transformations
  - Covering the right metadata artefacts, graphs help to automate the whole data integration lifecycle
- Main graph families
  - Property graphs
  - Knowledge graphs