BITS Pilani
Pilani | Dubai | Goa | Hyderabad

DSECL ZG 522: Big Data Systems

Session 5-3: Graph Databases

Janardhanan PS

janardhanan.ps@wilp.bits-pilani.ac.in

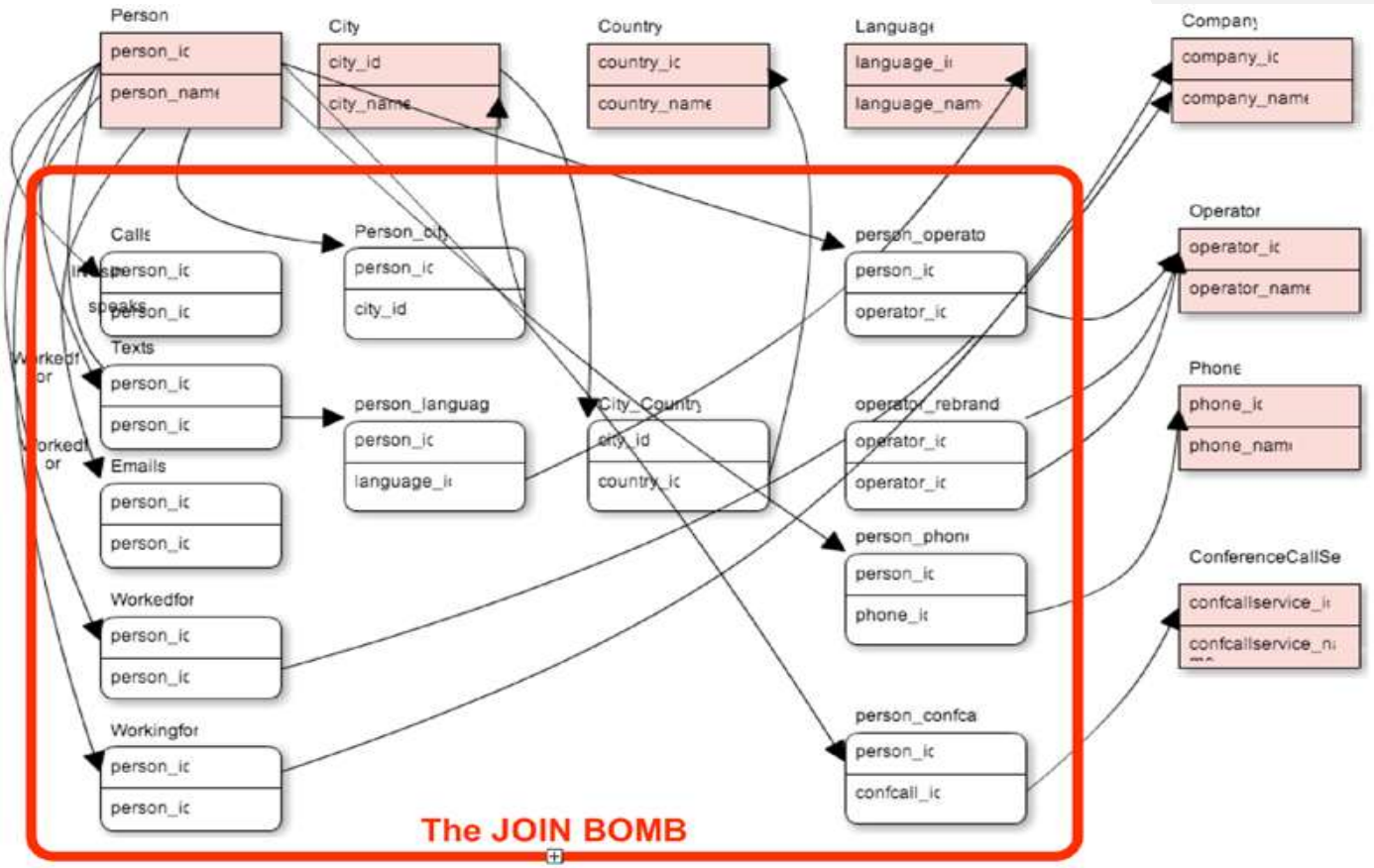
5 Signs You Need a Graph Database

Is your traditional database not up to the task of storing and deciphering your complex data?

Here are five of the telltale signs:

1. Hard-to-navigate interconnected data
2. Lack of real-time insights
3. Inflexible data structures
4. Inefficient querying
5. Inability to decode a web of highly connected data

Relational Database is Anti-Relational



What is Graph Database

- A database built around graph structure – Nodes & Edges
- Each Node has a set of incoming and outgoing edges
- Relationships / Edges connect nodes
- Each Node / Edge is uniquely identified
- Each Node/Edge has a collection of properties
- Properties store named data values
- Each edge has a label that defines the relationship between its ending nodes
- Can store any kind of data
- No fixed number of schema or properties
- Nodes are indexed for fast initial lookup
- Strong mathematics **(graph theory)** behind

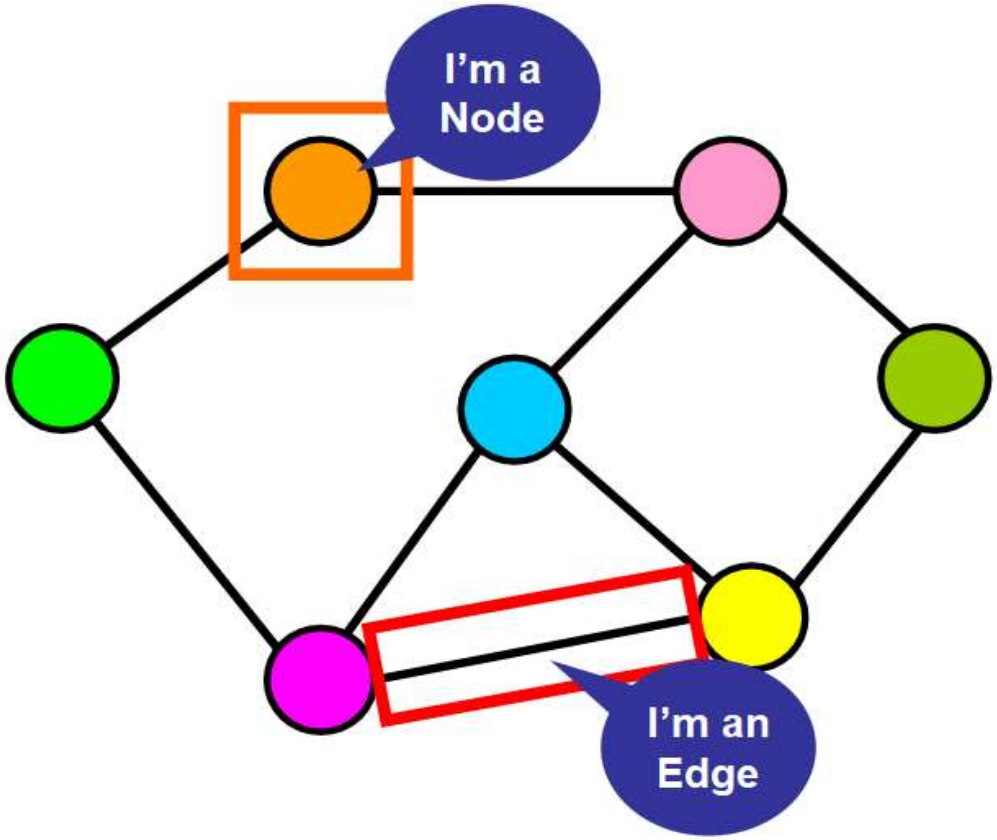
Graph Theory

Leonhard Euler

invented Graph Theory in **1736**,

275 years before Edgar Codd formulated the relational model

Graphs



Graphs - Mathematically

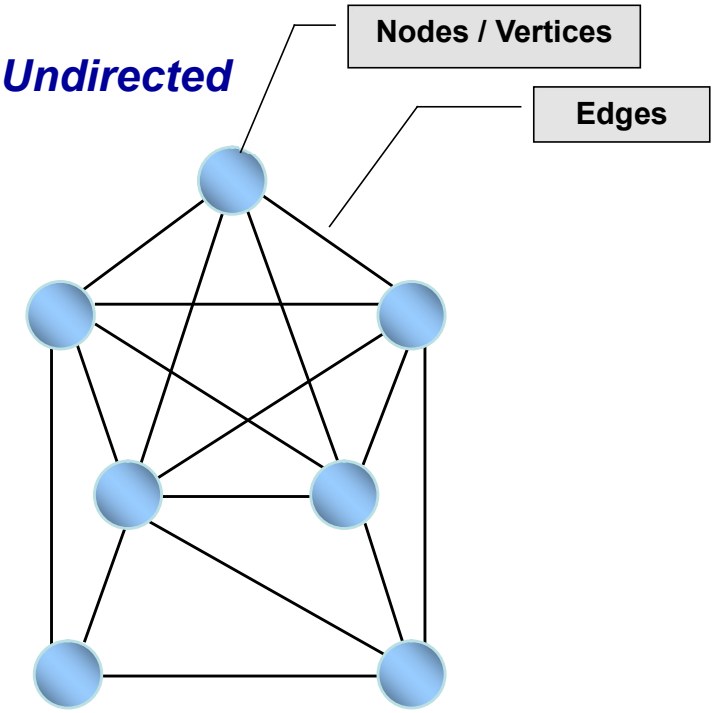
$$G = (V, E)$$

$$V = \{v_1, v_2, ..., v_n\}$$

$$E = \{e_k = (v_i, v_j) \mid v_i, v_j \in V, k = 1, ..., m\}$$

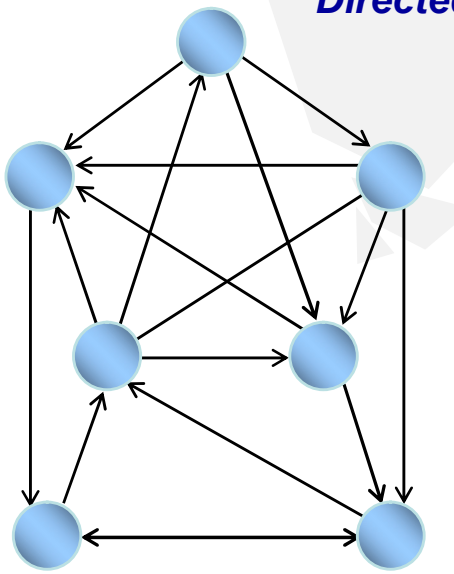
Graph with 7 nodes and 16 edges

Undirected



$$(v_i, v_j) = (v_j, v_i)$$

Directed

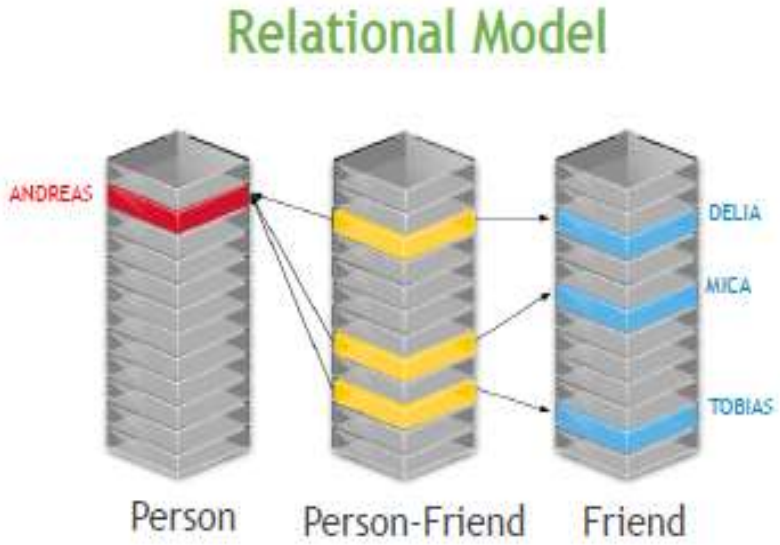


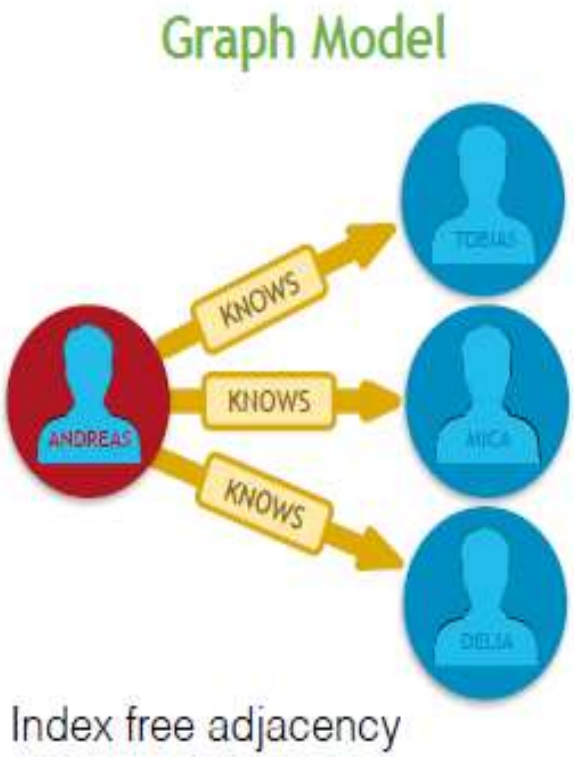
$$(v_i, v_j) \neq (v_j, v_i)$$

Graph computing

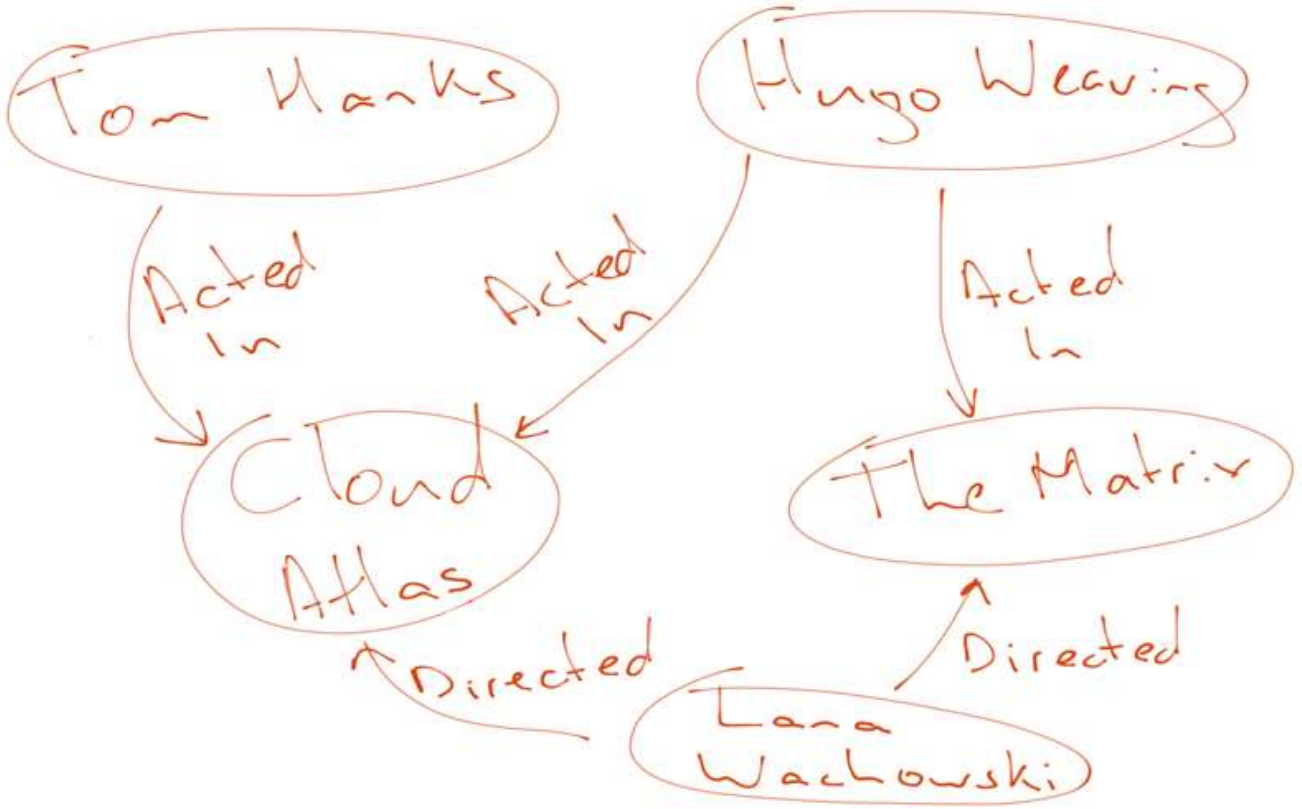
- When to use a graph DB ?
  - A relationship-heavy data set with large set of data items
  - Queries are like graph traversals but need to keep query performance almost constant as database grows
  - A diverse set of queries may be asked from the data and static indices on data will not work
- What are property graphs:
  - Data is represented as vertices and edges with properties
  - Properties are Name Value pairs
  - Edges are relationships between vertices

Relational Vs Graph Models

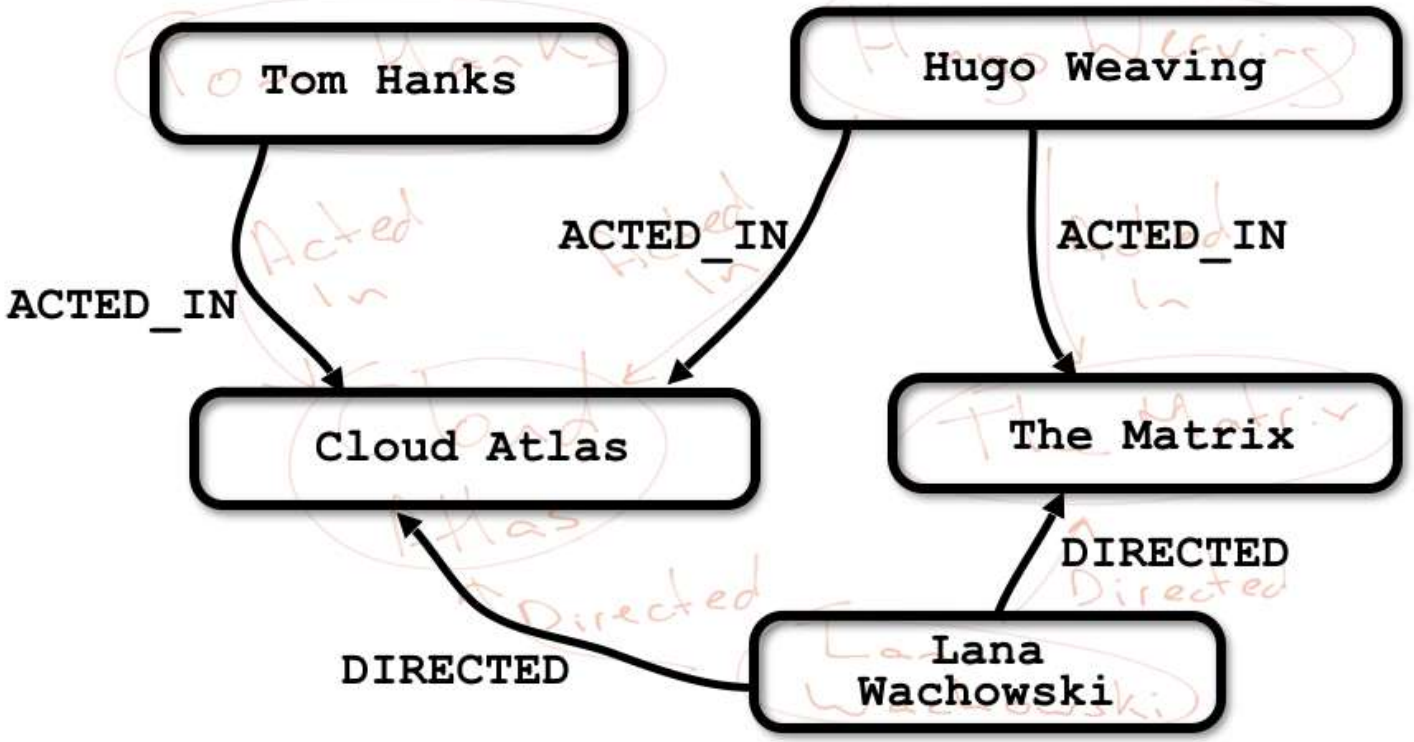




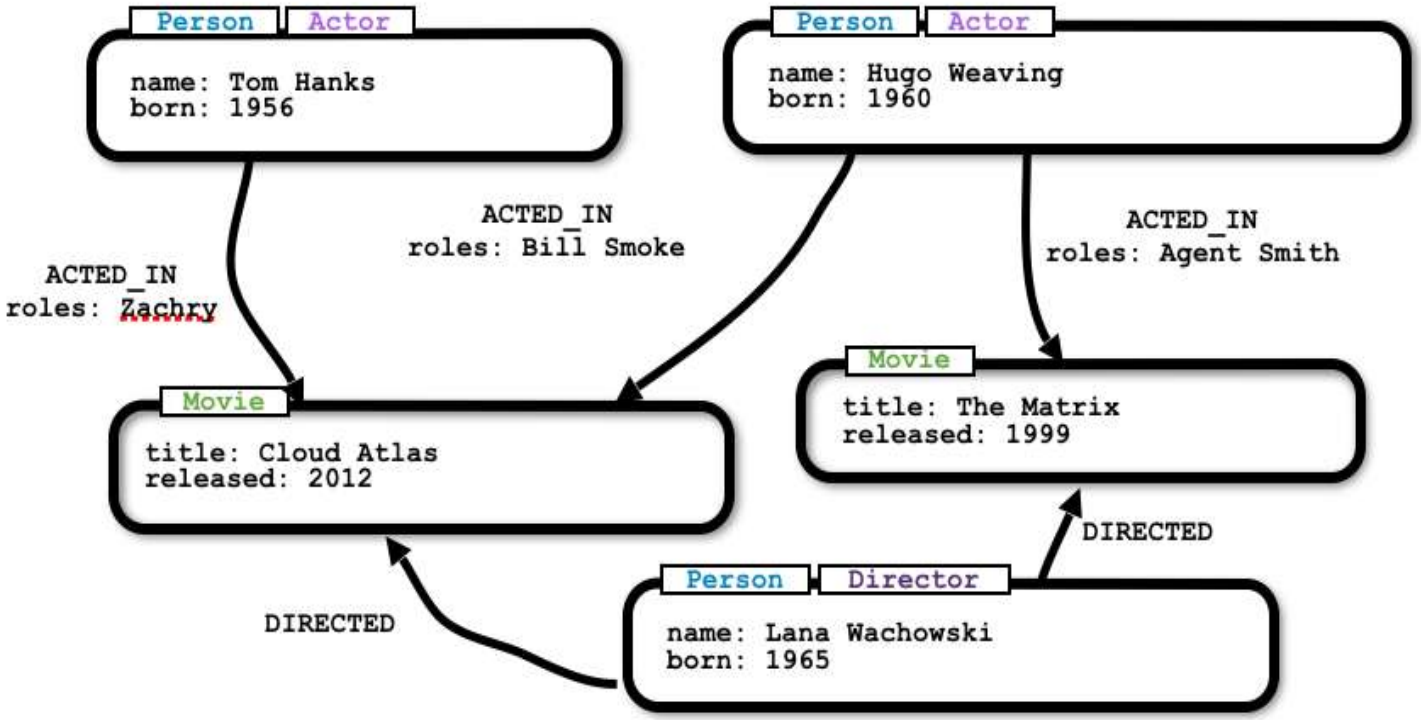
Graph Data Modelling – Whiteboard friendly model



Translating to Nodes & Relationships



Add labels and properties



Nodes

- Stores data in a Graph, with records called Nodes
- The simplest graph has just a single node with some named values called Properties
- Data is stored as Properties
- Properties are simple name/value pairs
- Schema-free, nodes can have a mix of common and unique properties
- Nodes are usually indicated by Nouns
- Nodes are the name for data records in a graph **(row in RDBMS Table)**

Labels

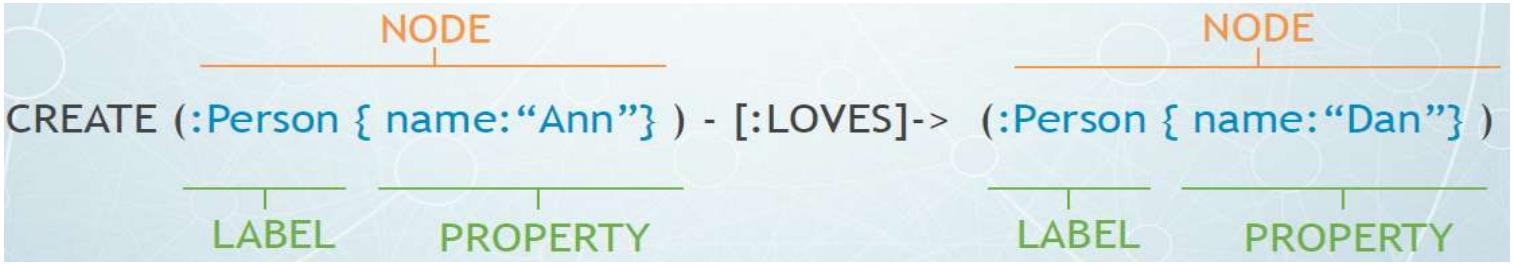
- Labels are tags that associate a set of nodes.
- Nodes can be grouped together by applying a Label to each member.
- In social graph, we'll label each node that represents a Person.
- Apply the label "Person" to the node we created for Emil
- A node can have zero or more labels
- Labels do not have any properties
- **Label Name maps to Table Name in RDBMS**

More data , More nodes

- Like any database, storing data in graph database can be as simple as adding more records
- Graph databases (Neo4j) can store billions of nodes
- Similar nodes can have different properties
- Node Properties can be
- Strings
- Arrays of Strings
- Numbers
- Boolean values
- Byte[]
- Date
- Node properties are usually adjectives

Relationships

- The real power of Graph database is in storing connected data
- To associate any two nodes, add a Relationship which describes how the records are related.
- Relationships always have direction
- Relationships always have an identifier
- Relationships form patterns of data
- Relationships are usually represented by Verbs



Relationship properties

- Relationships can also have properties , they are also data records
- Relationship properties:
- Store information shared by two nodes
- They are usually adverbs
- Examples:
  - Emil has known Johan since 2001
  - Emil rates Ian 5 (out of 5)

Property Graph

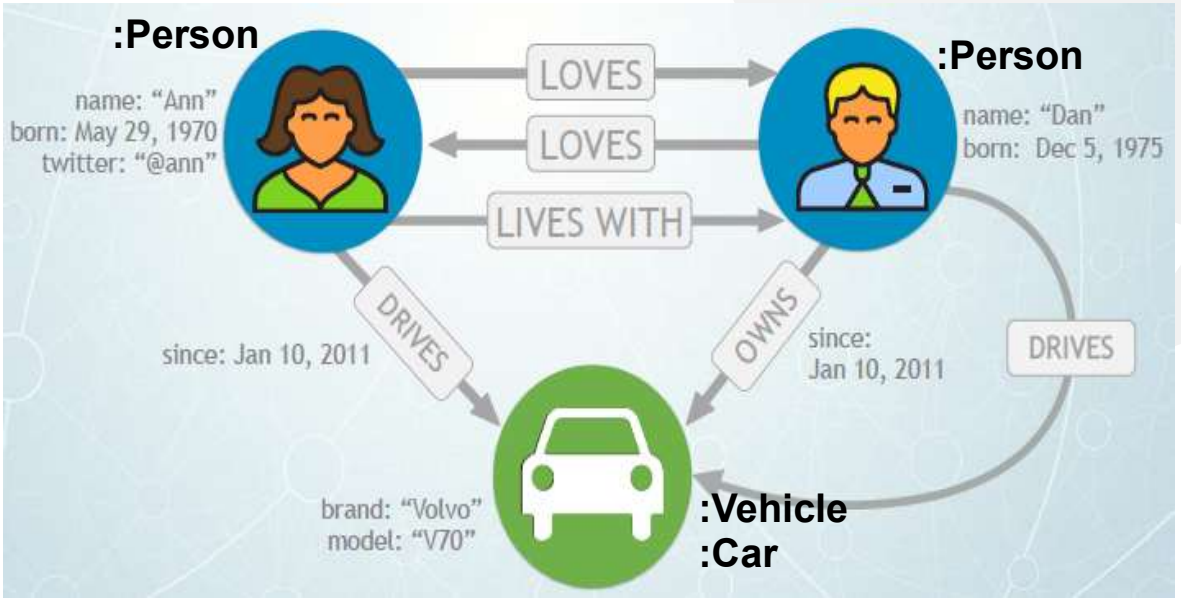
- Data stored in the form of properties of Nodes and Edges:
- Properties are Name Value pairs

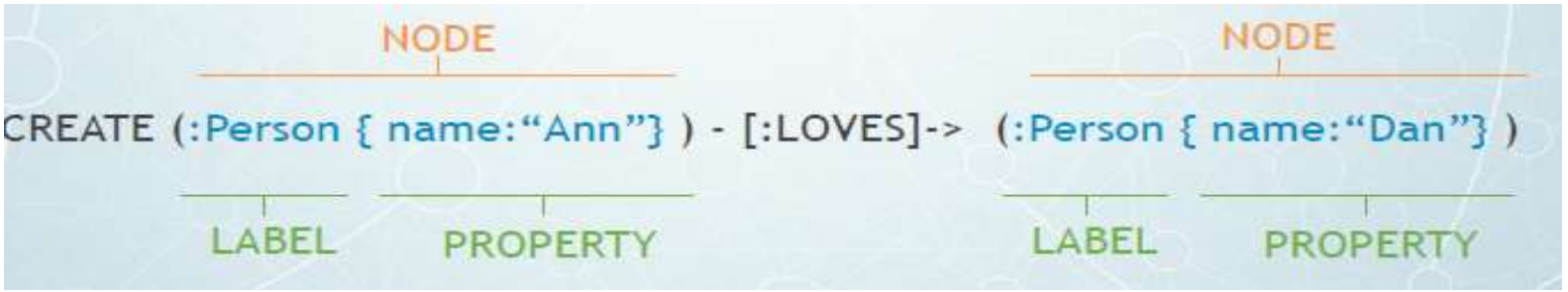
Nodes:

- The objects in the graph
- Can have name-value properties
- Can be labeled

Relationships:

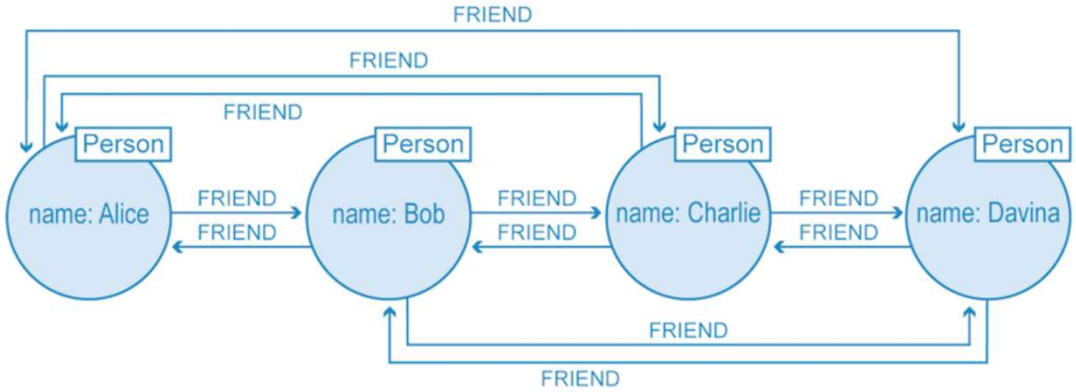
- Relate nodes by type and direction
- Can have name-value properties



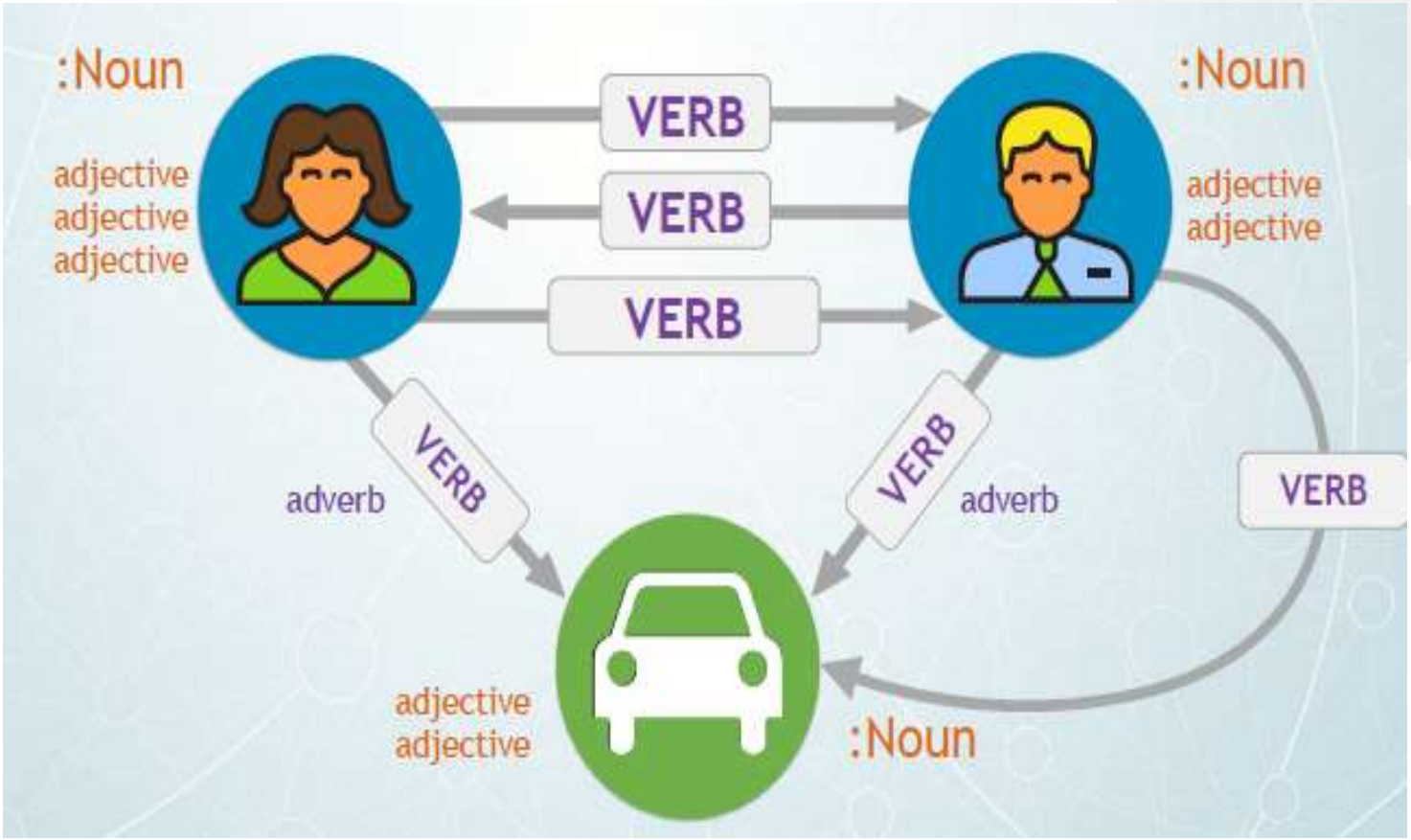


When to use Graph DB

- A relationship-heavy data set with large set of data items
- Queries are like graph traversals but need to keep query performance almost constant as database grows
- A variety of dynamically formed queries may be asked on the data and static indices on data will not help



Mapping Properties To English Language context



20

Graph Data Base Space

Amazon Neptune
Fast, reliable graph database built for the cloud

neo4j

Cayley

ArchiGraph

OrientDB

HyperGraphDB

ANZOGRAPH® DB

FlockDB

Titan

GraphBase

IBM

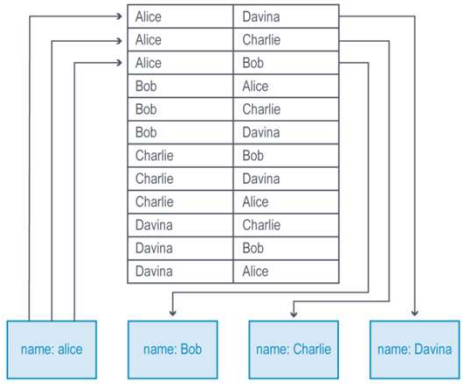
IBM Graph

blazegraph™
by SYSTAP

Native vs Non-Native Graph storage

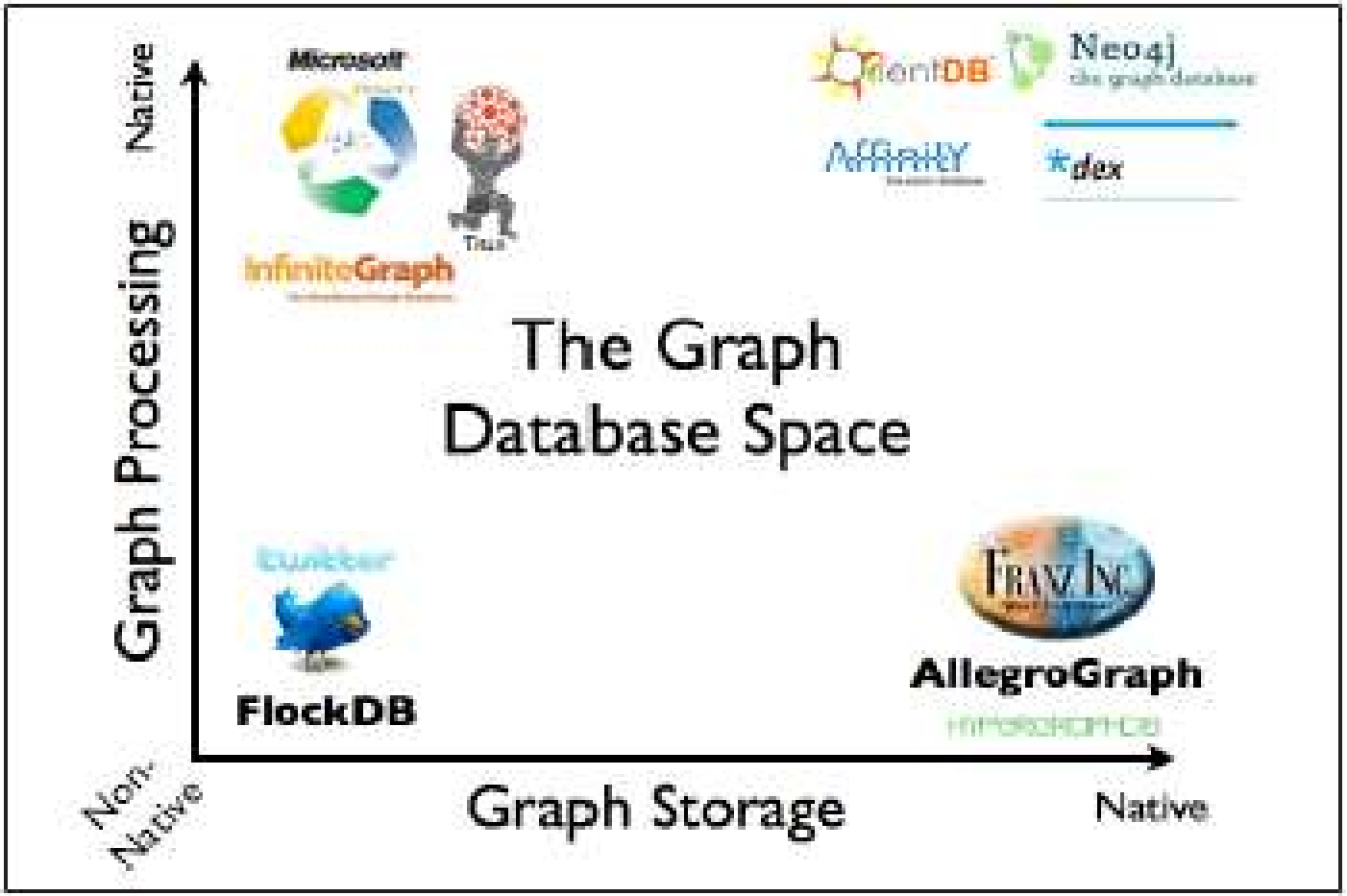
- Non-native graph computing platforms can use external DBs for data storage
  - e.g. TinkerPop is an in-memory DB + computing framework that can store in ElasticSearch, Cassandra etc.
- Native platform support built-in storage
  - e.g. Neo4j
- Native approach is much faster because adjacent nodes and edges are stored closer for faster traversal
  - In a non-native approach, extensive indexing has to be used
- Native approach scales as nodes get added

One-hop index



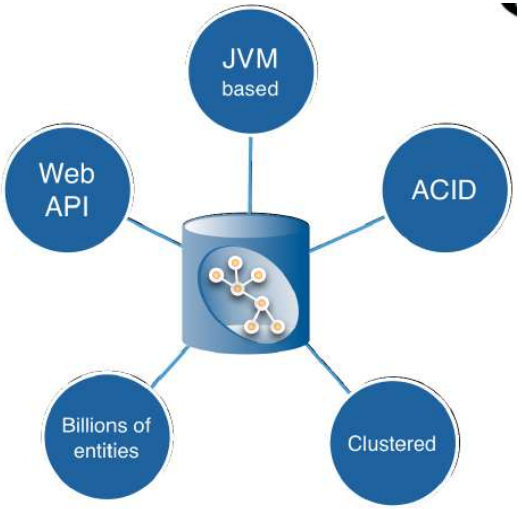
https://neo4j.com/blog/native-vs-non-native-graph-technology/

Graphs for Storage and Processing



What is Neo4J

- It's is a Graph Database supporting full ACID Transactions
- Embeddable in applications and server deployable
- Java based, Open sourced
- Schema free, bottom-up data model design
- Neo4j is stable
- In 24/7 operation since 2003
- Neo4j is under active development
- High performance graph operations
- Supports the **Cypher** query language
- Traverses 1,000,000+ relationships/sec on commodity hardware
- No. of nodes and relationships decide Volume of data



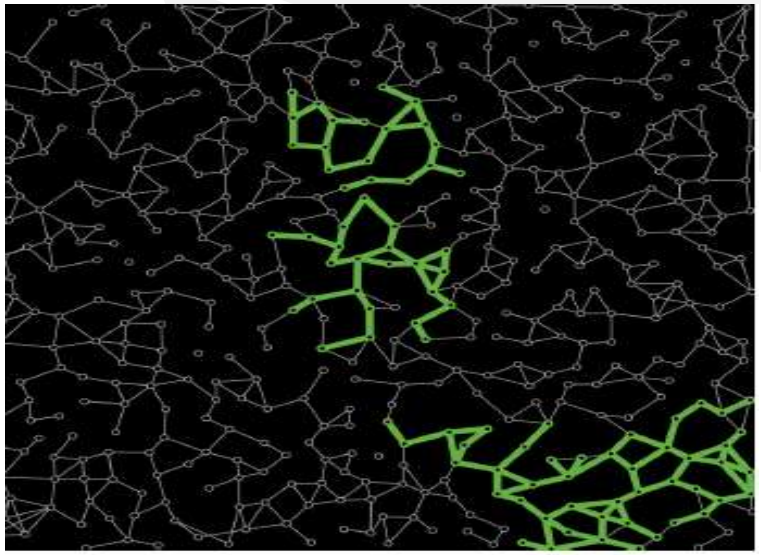
One-hop index

neo4j – Key product features

- Native Graph Storage - Ensures data consistency and performance
- Native Graph Processing - Millions of hops per second, in real time
- Whiteboard Friendly Data Modelling - Model data as it naturally occurs
- High Data Integrity - Fully ACID transactions
- Powerful Expressive Query Language - Requires 10x to 100x less code than SQL
- Scalability and High Availability -Vertical & Horizontal scaling optimized for graphs
- Built-in ETL - Seamless import from other databases
- Integration - Drivers and APIs for all popular languages

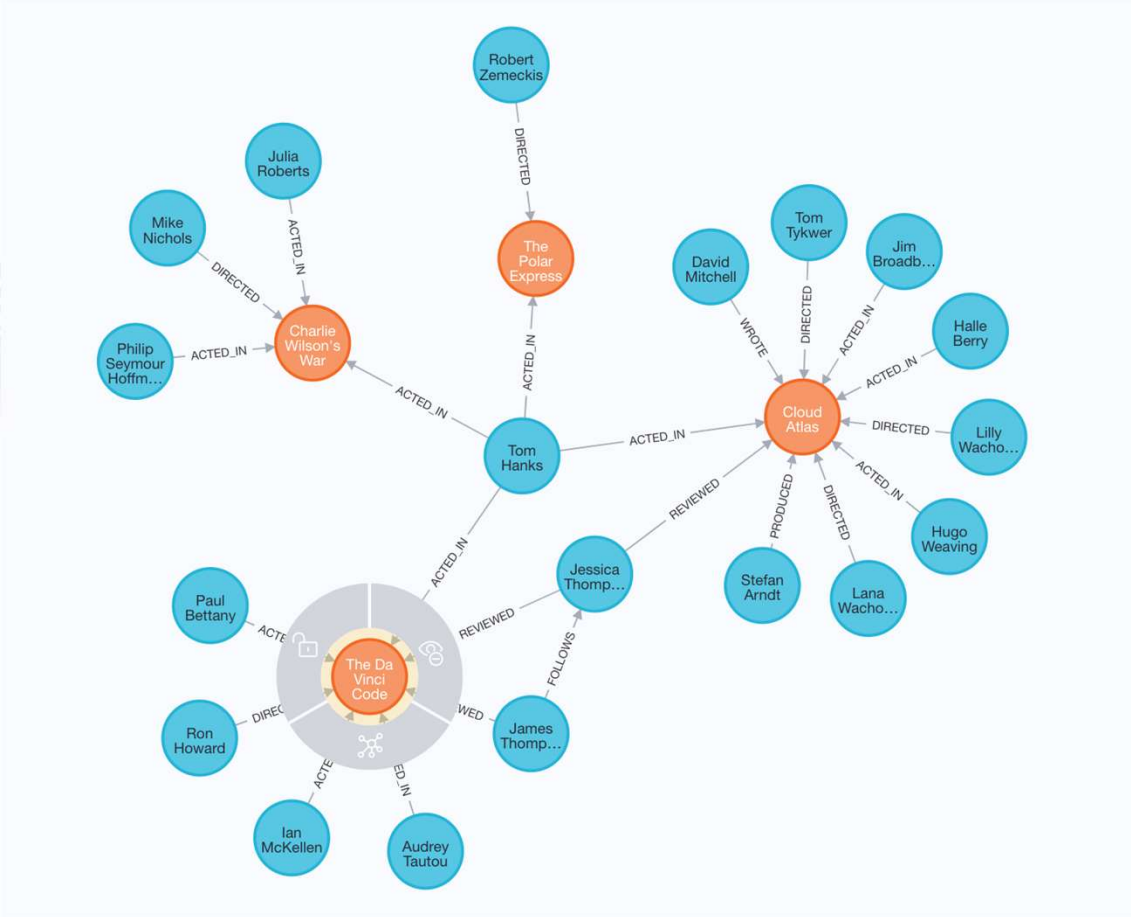
Querying a Neo4j Graph

- Nodes are indexed using Apache Lucene - Scalable high-performance indexing
- Nodes can be indexed by properties also
- Find one or more start nodes - **Anchor** nodes
- Explore surrounding graph starting from the anchor
- Millions of hopes per second using **index free adjacency**



Neo4j / Cypher

- Cypher is a Declarative language for graph query
- Example: match (:Person {name: 'Tom Hanks'})-[:ACTED_IN]->(m:Movie) where m.released > 2000 RETURN m limit 5



Neo4j / Cypher: More queries

- Find movies that Tom Hanks acted in and directed by Ron Howard released after 2000
  - Match (:Person {name: 'Tom Hanks'})-[:ACTED_IN]->(m:Movie),(:Person {name: 'Ron Howard'})-[:DIRECTED]->(m) where m.released > 2000 RETURN m limit 5
- Who were the other actors in the movie where Tom Hanks acted in and directed by Ron Howard released after 2000
  - Match (:Person {name: 'Tom Hanks'})-[:ACTED_IN]->(m:Movie),(:Person {name: 'Ron Howard'})-[:DIRECTED]->(m), (p:Person)-[:ACTED_IN]->(m) where m.released > 2000 RETURN p limit 5

SQL Vs Cypher

SQL Statement:

```
SELECT name FROM Person

LEFT JOIN Person_Department
  ON Person.Id = Person_Department.PersonId

LEFT JOIN Department
  ON Department.Id = Person_Department.DepartmentId

WHERE Department.name = "IT Department"
```

Cypher Statement:

```
MATCH (p:Person)<-[:EMPLOYEE]-(d:Department)

WHERE d.name = "IT Department"

RETURN p.name
```

Neo4j editions

Community Edition - is a full functioning version of Neo4j suitable for single instance deployments. It has full support for key Neo4j features, such as ACID compliance, Cypher and access via the binary protocol and HTTP APIs. It is ideal for smaller internal or do-it-yourself projects that do not require high levels of scaling or professional services and support.

Enterprise Edition - extends the functionality of the Community Edition to include key features for performance and scalability such as a clustering architecture for High Availability and online backup functionality. It is the right choice for production systems with availability requirements or needs for scaling up or out.

Subscription based licensing:

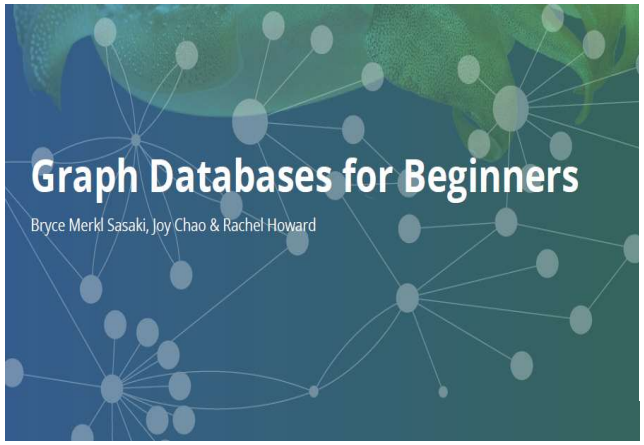
- USD 10,000 per core - license cost per year
- 3 Machines / 8 Cores each - 20,000 per year (Total 24 cores)

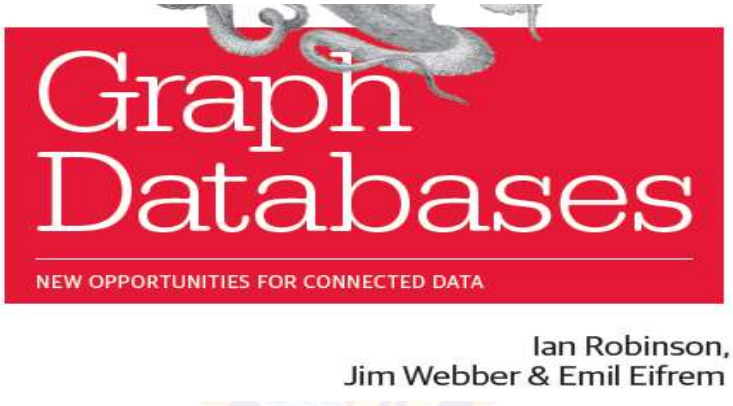
Neo4j on Cloud

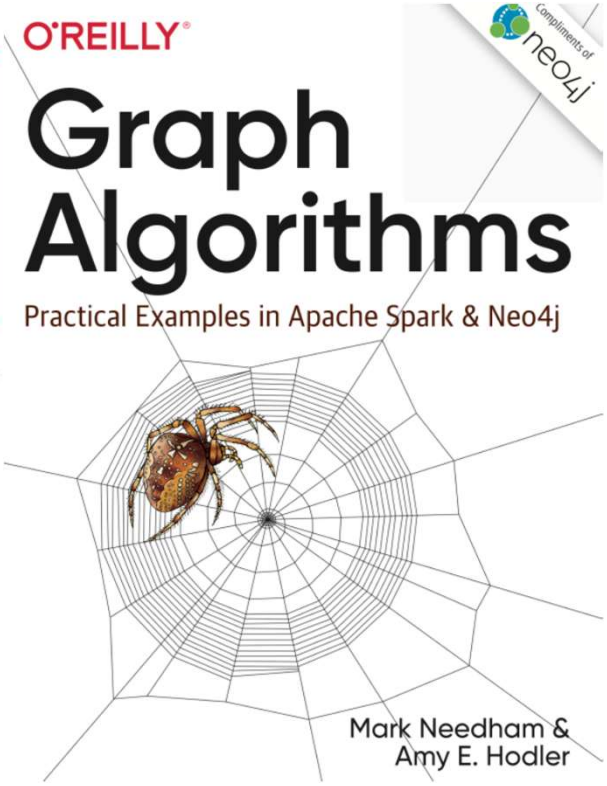
- Neo4j Enterprise Edition on AWS
- Neo4j - The Easiest Way to Graph on MS Azure
- NEO4J AURA on Google Cloud

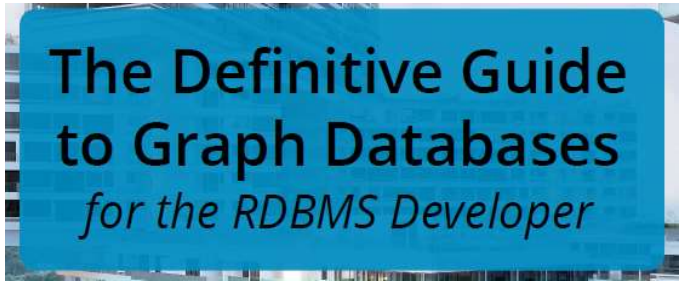
Webcasts /Books on Graph Database

https://www.infoq.com/presentations/neo4j-graph-theory
https://www.infoq.com/presentations/graph-db-tools









Neo4j Training / Certification

Neo4j Training

neo4j Graphacademy - https://graphacademy.neo4j.com/

Free, Self-Paced, Hands-on Online Training | Free Neo4j Courses from GraphAcademy

Spark GraphX

GraphX is Apache Spark's API for graphs and graph-parallel computation.

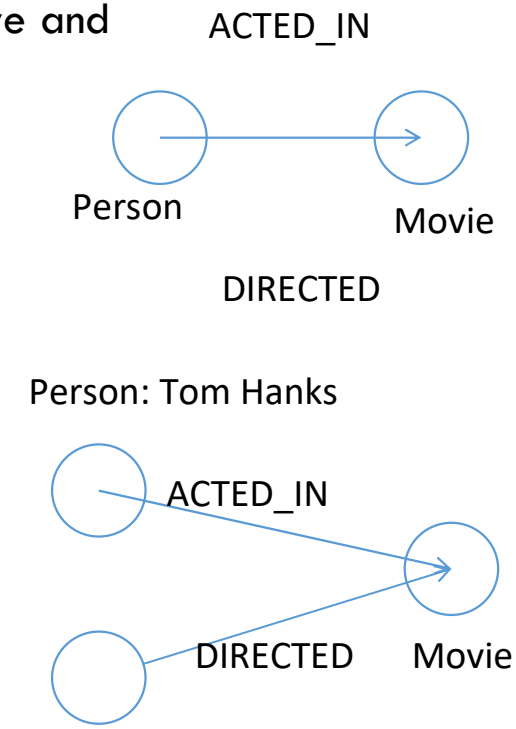
https://spark.apache.org/graphx/

Features:

- Flexibility
  - ✓ GraphX unifies ETL, exploratory analysis, and iterative graph computation within a single system. You can view the same data as both graphs and collections, transform and join graphs with RDDs efficiently, and write custom iterative graph algorithms
- Speed
  - ✓ Comparable performance to the fastest specialized graph processing systems.
  - ✓ GraphX competes on performance with the fastest graph systems while retaining Spark's flexibility, fault tolerance, and ease of use.
- Algorithms
  - ✓ Choose from a growing library of graph algorithms.
  - ✓ In addition to a highly flexible API, GraphX comes with a variety of graph algorithms, many of which were contributed by our users

Apache Tinkerpop / Gremlin

- TinkerPop is a computing platform that connects to GraphDBs that actually store the nodes and edges. Built-in TinkerGraph stores in-memory data only.
- Gremlin is the query language (with traversal machine) that supports Declarative and Imperative flavours
- Sample queries
  - ✓ movies where Tom Hanks has acted
    - g.V().hasLabel('person').has('name','Tom Hanks').outE('ACTED_IN').hasLabel('movies').values('name')
  - ✓ movies where Tom Hanks has acted and directed by Ron Howard
    - g.V().hasLabel('person').has('name','Tom Hanks').outE('ACTED_IN').inE('DIRECTED').has('name','Ron Howard').outE('DIRECTED').values('name')



Importing data from RDBMS to Graph database

When deriving a graph model from a relational model, you should keep the following general guidelines in mind:

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

https://neo4j.com/docs/getting-started/appendix/tutorials/guide-import-relational-and-etl/

Refer: The Definitive Guide to Graph Databases for the RDBMS Developer

Handson with GraphDB

Download and install Neo4j Desktop

- ✓ Basic Cypher commands
- ✓ NoSQL features of Graph Database
- ✓ Setting Relationships
- ✓ Cypher queries
- ✓ Path finding
- ✓ RDBMS to Graph Database conversion (Exercise)

https://guides.neo4j.com/northwind/index.html

For creating indexes, follow new cypher syntax

```
CREATE  INDEX  CustomerIndex FOR (n:Customer) ON (n.customerID)
CREATE  INDEX OrderIndex FOR (n:Order) ON (n.orderID)
```

Summary

Graph DBs and computing models are very suitable when data sets are relationship heavy - can be modelled as large number of nodes and edges and queries are similar to graph traversal

- ✓ Complex relation centric queries are possible
- ✓ Graph traversal costs can be kept stable with data growth

Next Session:
Hadoop architecture and filesystem