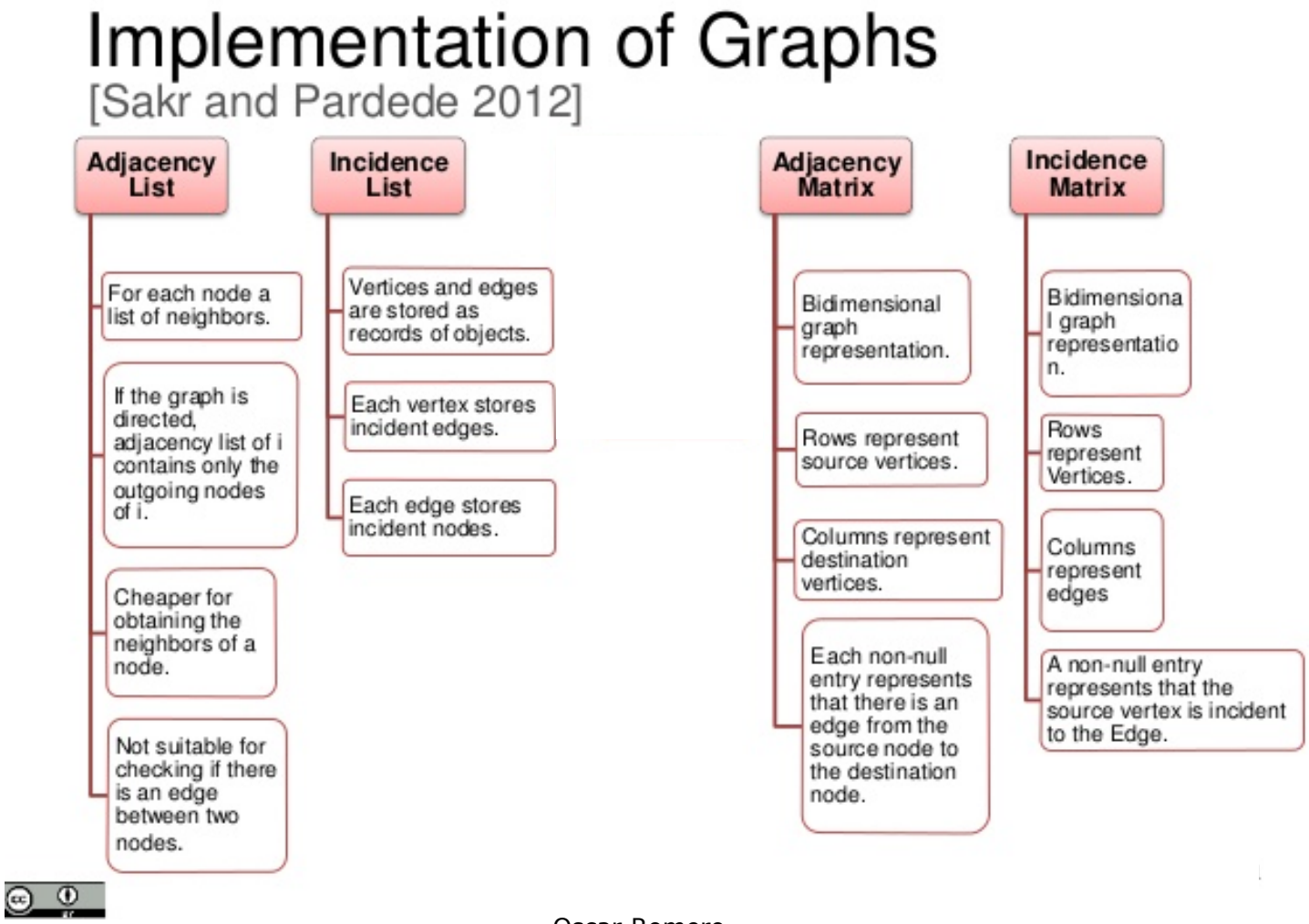
GRAPH DATABASES

Graph Databases

- A graph database:
  - Provides means to **store** and **persist** graph data
    - Each having potentially different physical graph data models
  - Provide means to **process** graph data
    - Note this does not diminish the possibility to plug a storage system with an external processing system

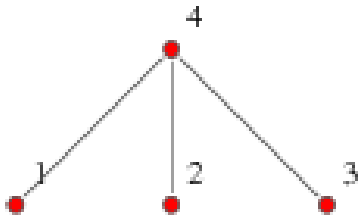
- Graph frameworks typically refer to processing frameworks. Thus, like the Relational Algebra, MapReduce or Spark, provide means to extract data from databases **BUT DO NOT STORE GRAPHS**
  - Pregel
  - Giraph
  - Turi / GraphLab
  - Etc.

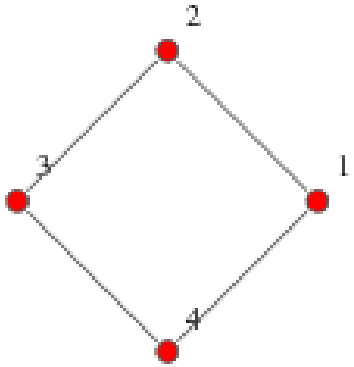
Implementation of Graphs

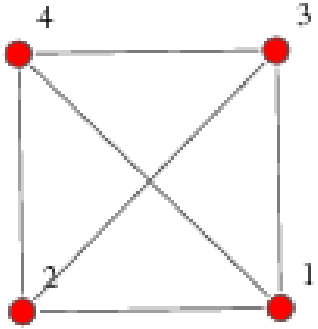


Graph Databases Architectures

- Adjacency matrix (baseline)



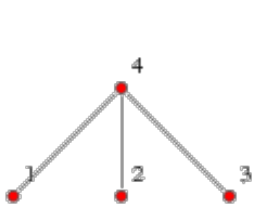


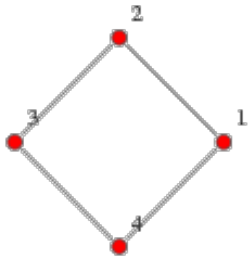


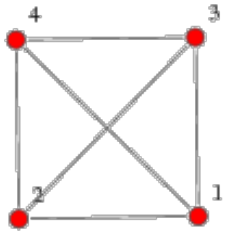
$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Activity

- *Objective: Understand the different structures needed to implement graphs following the main graph implementation strategies*
  - *Implement the following graphs as an adjacency list **AND** as an incidence list*







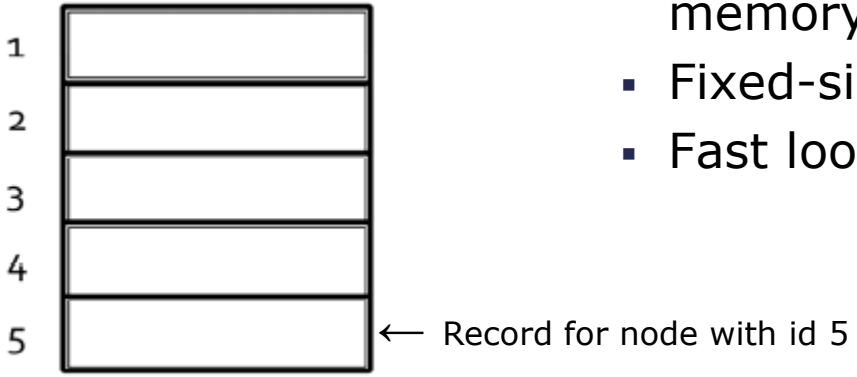
$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Examples

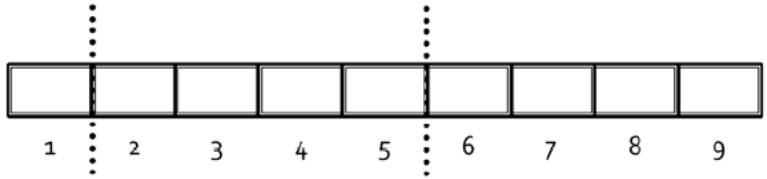
IMPLEMENTATION OF GRAPH DATABASES

Incidence Lists – Neo4J

Linked Lists – Neo4J



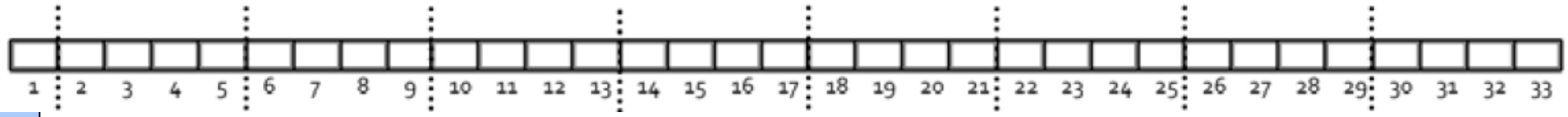
- One physical file to store all nodes (in-memory)
- Fixed-size record: 9 bytes in length
- Fast look-up: O(1)



- Each record is as follows:
- 1 byte (metadata; e.g., in-use?)
- 2-5 bytes: id first relationship
- 6-9 bytes: id first property

Incidence Lists – Neo4J

- Two files: relationship and property files.
  - Both contain records of fixed size
  - Cache with *Least Frequently Used* policy
- Relationship file (similarly for properties)
  - Metadata, id starting node, id end node, id type, ids of the previous and following relationship of the starting node and ending node, id first property

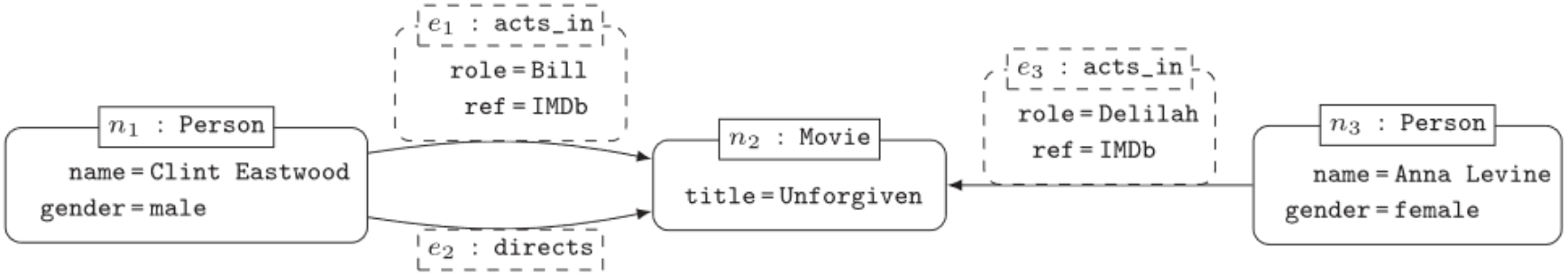


Incidence Lists – Neo4J

- Two files: relationship and property files.
  - Both contain records of fixed size
  - Cache with *Least Frequently Used* policy
- Property file
  - Metadata (incl. a bit determining whether it belongs to an edge or node), id node / edge, ids of the previous and following properties of the node / edge, id property name, id property value
  - Fixed size of 21 bytes

Activity

- *Objective: Understand how to implement a linked list to implement graphs*
- Consider the graph below. What would be the resulting data structures if you create such graph in Neo4J?



TYPES OF GRAPH DATABASES

Types of Graph Databases

- Some graph databases / processing frameworks are based on strong assumptions that are not explicit
- Operational graphs:
  - Map to the concept of a CRUD database
    - Nodes, edges can be deleted, updated, inserted and read
    - Example: Neo4j, Titan*, OrientDB, etc.
- Analytical graphs
  - They are *snapshots* that cannot be modified by the final user
    - Equivalent to the data warehouse but for graphs
    - Example: Sparksee, any graph framework, etc.

An Example of Graph Database

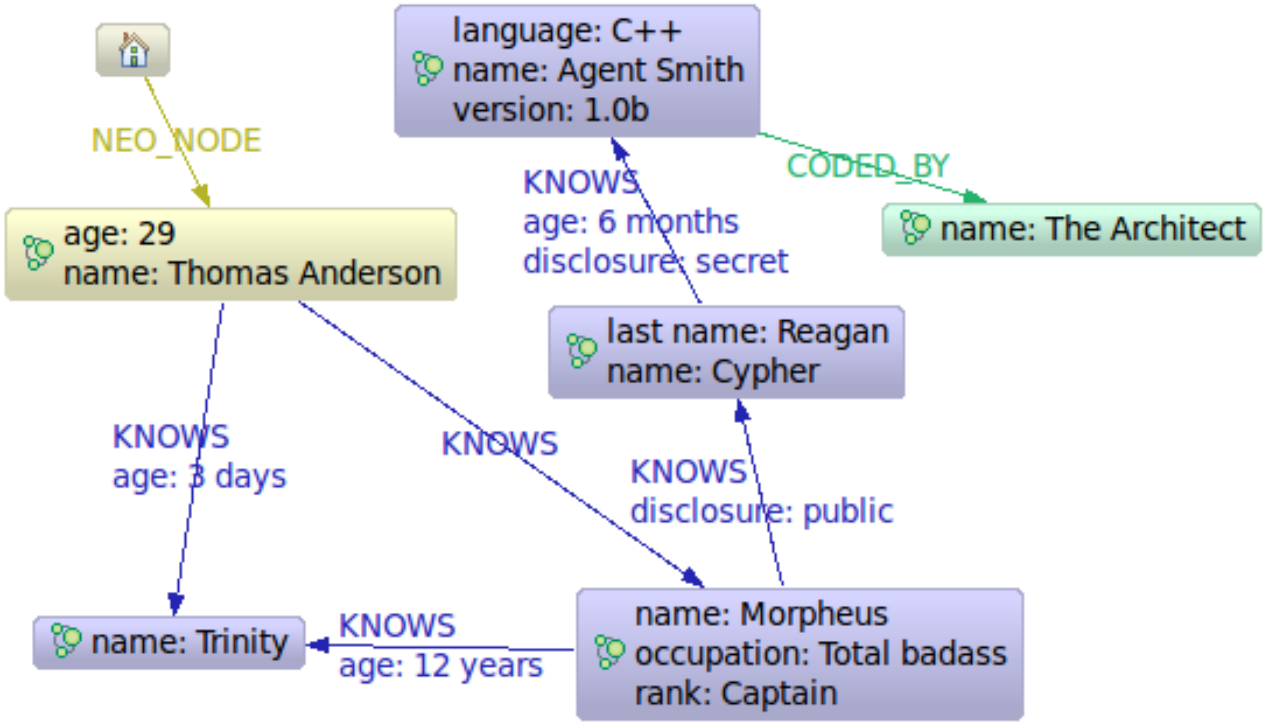
EXERCISE ON NEO4J

Neo4J Data Model

- It is a graph!
  - Use nodes to represent entities
  - Use relationships to represent the semantic connection between entities
  - Use node properties to represent entity attributes plus any necessary entity metadata such as timestamps, version numbers, etc.
  - Use relationship properties to represent connection attributes plus any necessary relationship metadata, such as timestamps, version numbers, etc.
- Unique constraints (~PK) can be asserted
  - `CREATE CONSTRAINT ON (book:Book) ASSERT book.isbn IS UNIQUE`

Neo4J: Data Model

- Example:

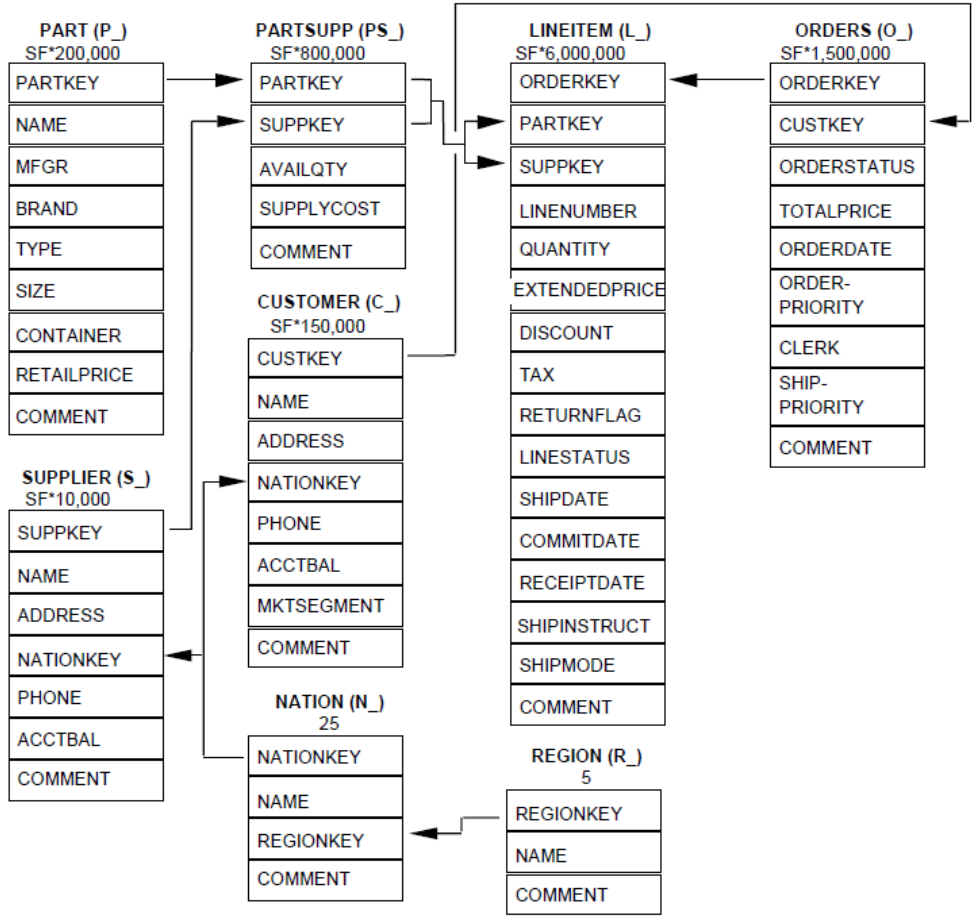


Source: Neo4J Java Tutorial

Activity: Modeling in Neo4J

- *Objective: Learn how to model graphs*
- *Tasks:*
  1. *(15') Model the TPC-H database as a graph*
     1. *Consider Neo4j internal data structures and model the graph **so that the given query is optimally executed (i.e., less hops and less I/O) on top of it***
     2. *Once the graph is created, write the SQL query as a Cypher query*

```
SELECT l_orderkey,
sum(l_extendedprice*(1-
l_discount)) as revenue,
o_orderdate, o_shippriority
FROM customer, orders, lineitem
WHERE c_mktsegment = '[SEGMENT]'
AND c_custkey = o_custkey AND
l_orderkey = o_orderkey AND
o_orderdate < '[DATE]' AND
l_shipdate > '[DATE]'
GROUP BY l_orderkey,
o_orderdate, o_shippriority
ORDER BY revenue desc,
o_orderdate;
```



Activity: Modeling in Neo4J

- *Objective: Learn how to model graphs*
- *Tasks:*
  1. *(15') Model the TPC-H database as a graph*
     1. *Consider Neo4j internal data structures and model the graph **so that the given query is optimally executed (i.e., less hops and less I/O) on top of it***
     2. *Once the graph is created, write the SQL query as a Cypher query*

Now, for the resulting graph and the Cypher query, draw the Neo4J internal structures and execute the access plan

- What type of query will it execute?
- Traverse the Neo4J internal structures to simulate the execution of the access plan

Summary

- Unfortunately, there is no standard to implement graph databases
- They all follow the same principles, but the way to implement it really affects graph processing
- When choosing a graph database, carefully think of:
  - Operational vs. Analytical graph database
  - Internal data structures
  - Impact of the internal data structures on the required graph processing for your project