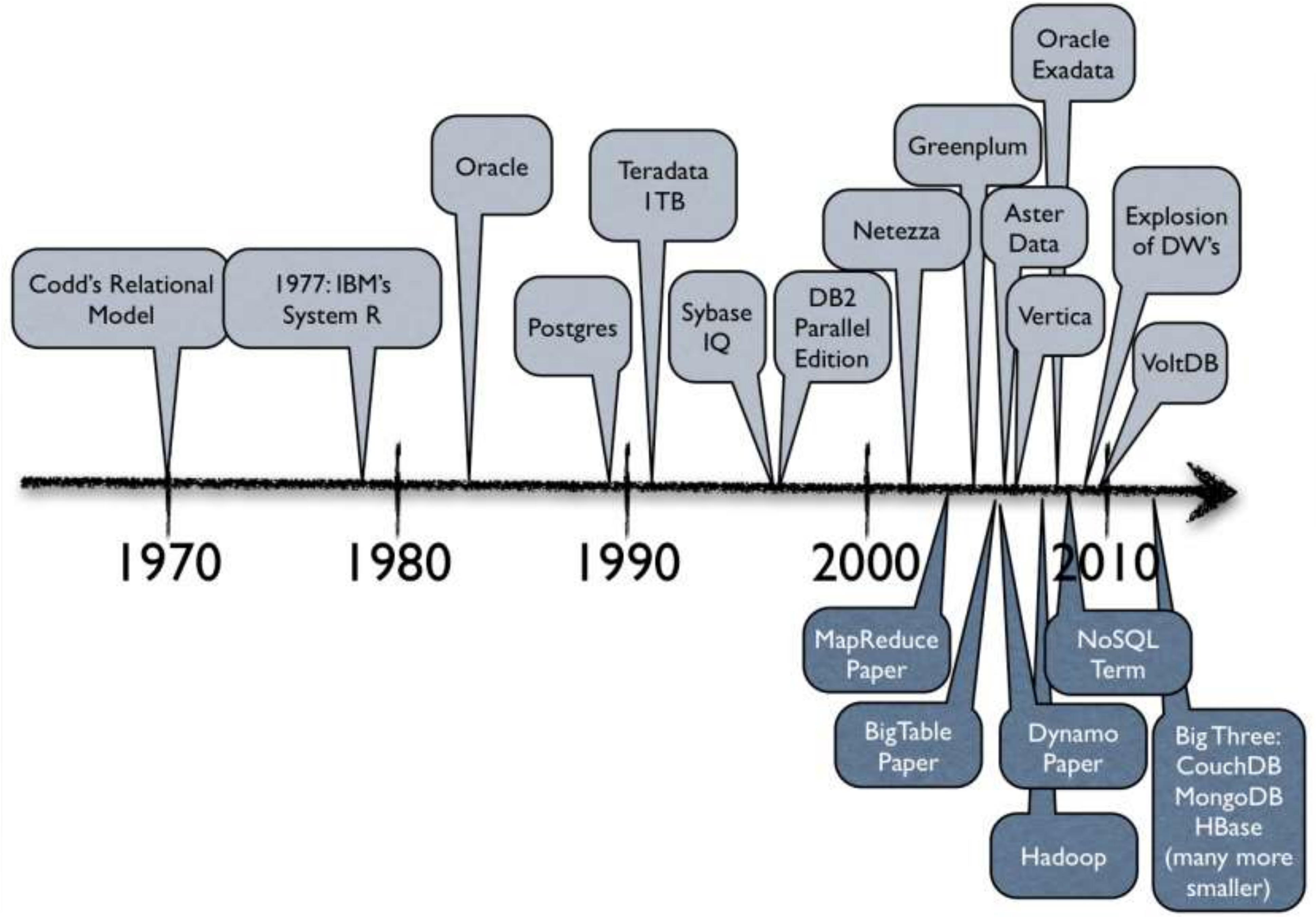
Graph Databases: Present & Future

Dr. Theodoros Chondrogiannis

Postdoctoral Researcher
Database and Information Systems Group
Department of Computer and Information Sciences
University of Kosntanz

16th Summer School of Applied Informatics
7.9.2019, Brno, Czech Republic

History of DBMS



Ashwani Kumar, NoSQL Databases

Relational databases

- ER modeling
- Relational schema
- Organize data in tables

Employee		
Name	Age	Salary
Alice	29	45000
Martin	26	38000
John	28	36000
Mario	35	58000

Department		
Director	Name	Building
Mario	IT	K
Alice	Finance	F

- Use indices to speed-up access

Relational databases - Pros

- Flexible by design
- Familiar BCNF structure (strong mathematical background)
- Transactions & ACID
- Very “mature” & well tested (mostly)
- Easy adoption/integration

Relational databases - Cons

- Large and unstructured data
- Lots of random I/Os and often write-heavy
- Not built for distributed applications
- Single point of failure
- Speed (performance), i.e, not fast enough for specialized applications
- Scale up, not out

Relational databases - Cons

- Scale up: grow capacity by replacing old machines with more powerful ones
    - Traditional approach
    - Expensive, as specialised machines cost a lot
- Scale out: incremental grow capacity by adding more COTS (Components Off The Shelf)
    - Phase in a few faster machines and replace old ones over a long period of time

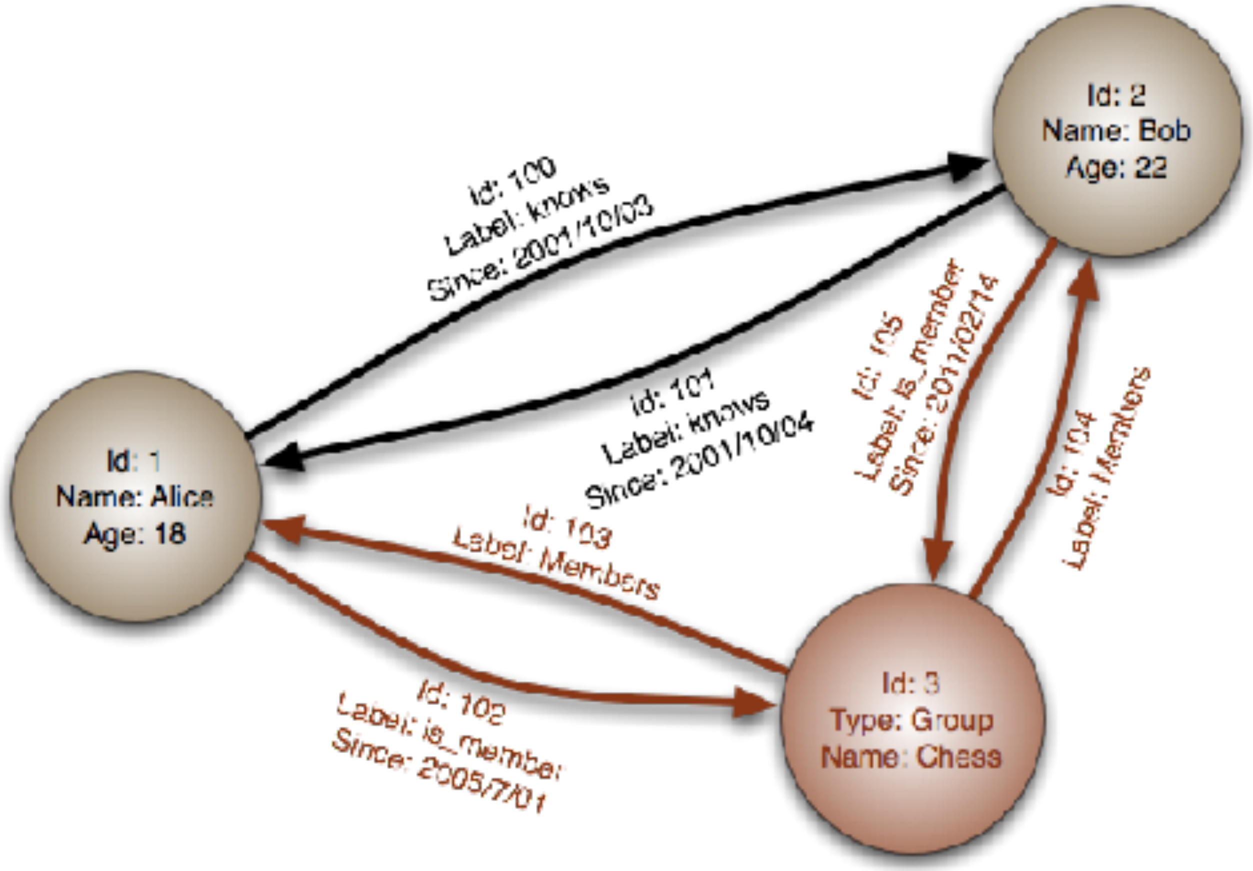
NoSQL Databases

NoSQL = “Not Only SQL”

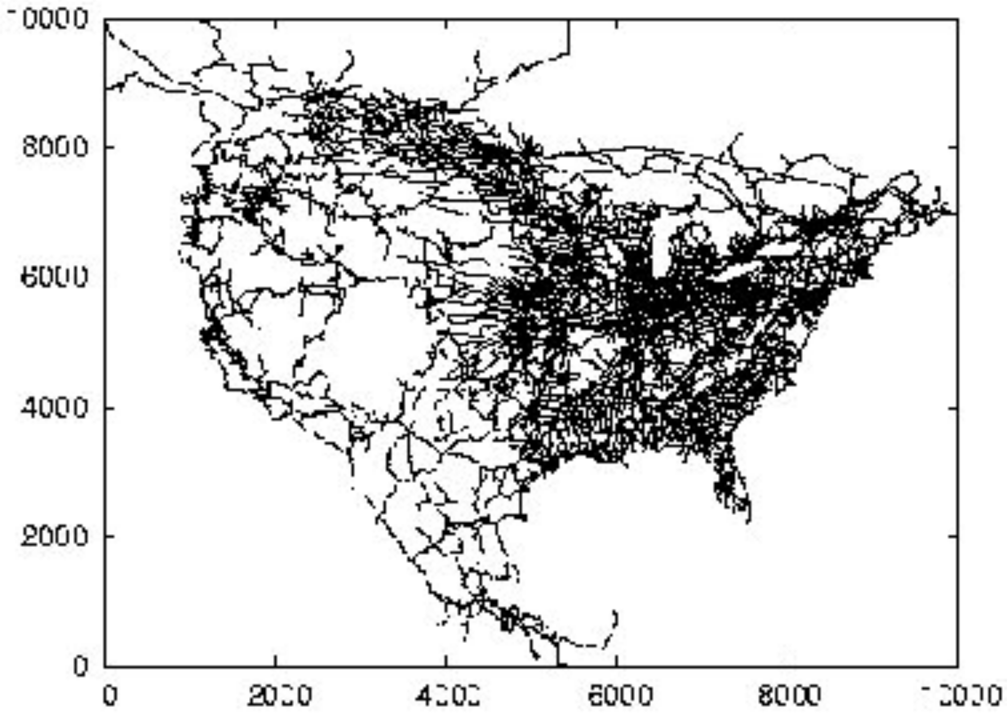
NoSQL Databases

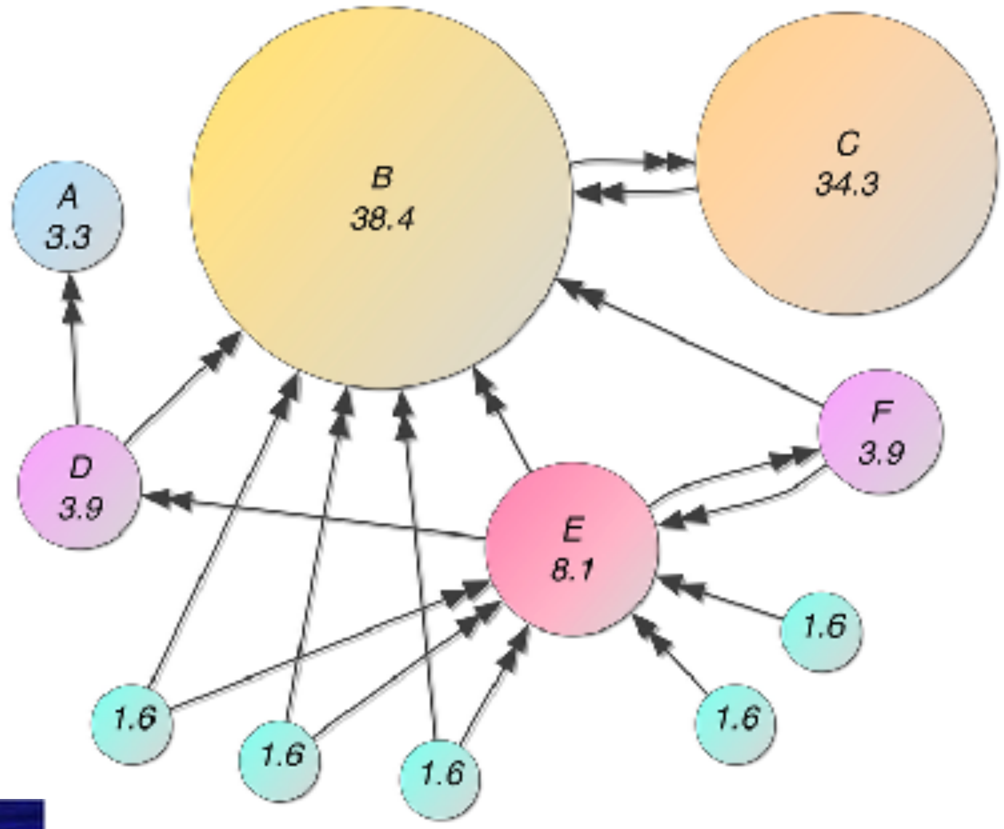
- Key/Value Stores
- Document Databases
- Column Oriented Databases
- **Graph Databases**
- More…

```
{
    "name": "Gina"
    "age": 24
    "occupation": "waitress"
}
```



Applications



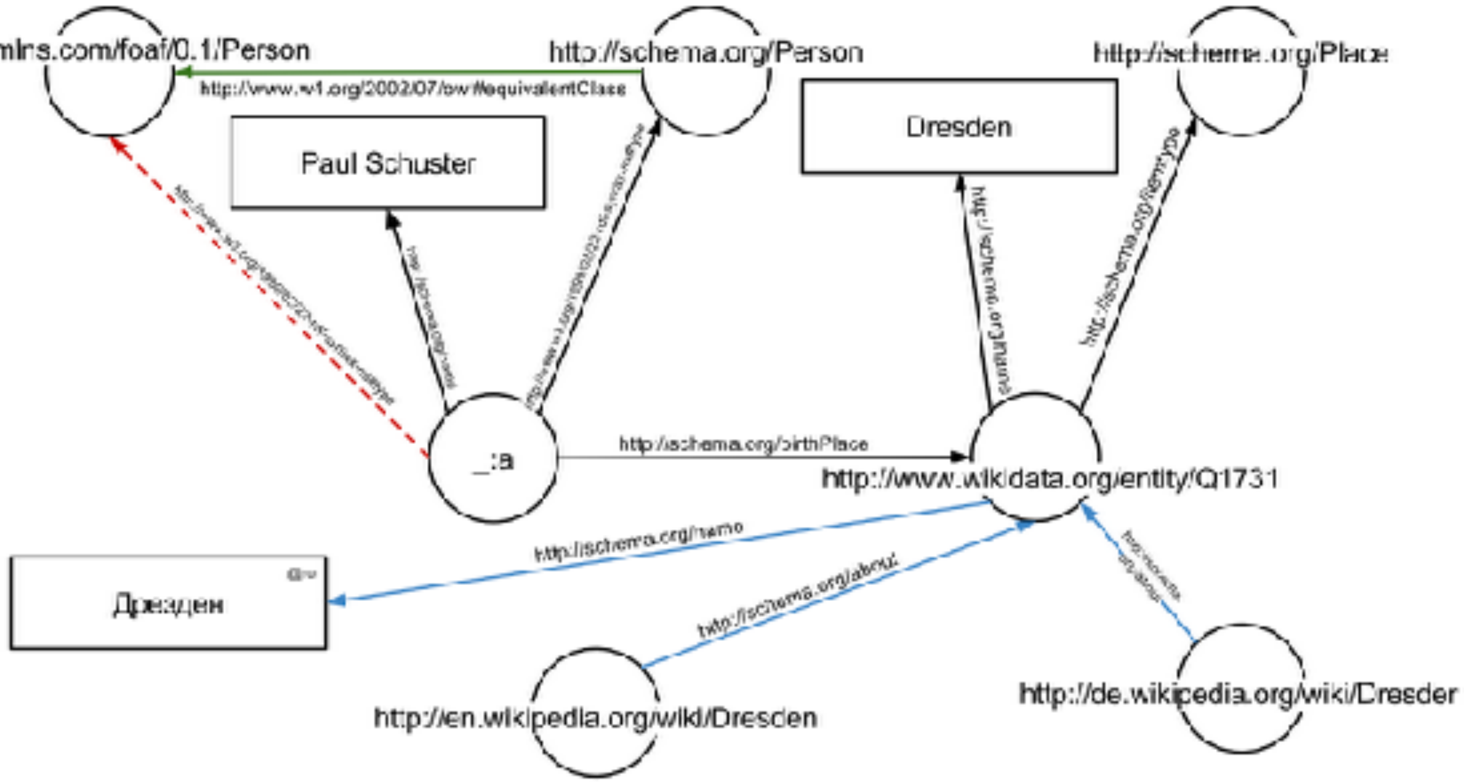


https://www.cs.utah.edu/~lifeifei/SpatialDataset.htm

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Not an Inherently New Idea

- RDF Triple Stores
  - RDF Schema
  - SPARQL


http://xmlns.com/foaf/0.1/Person
http://schema.org/Person
http://www.w1.org/2002/07/owl#equivalentClass
http://schema.org/Place
Paul Schuster
Dresden
_:a
http://schema.org/birthPlace
http://www.wikidata.org/entity/Q1731
http://schema.org/name
http://schema.org/name
Дрезден
http://schema.org/about
http://en.wikipedia.org/wiki/Dresden
http://de.wikipedia.org/wiki/Dresden

- XML Databases
  - XQuery
  - XPath

Graph Databases: The Present

Graph Databases

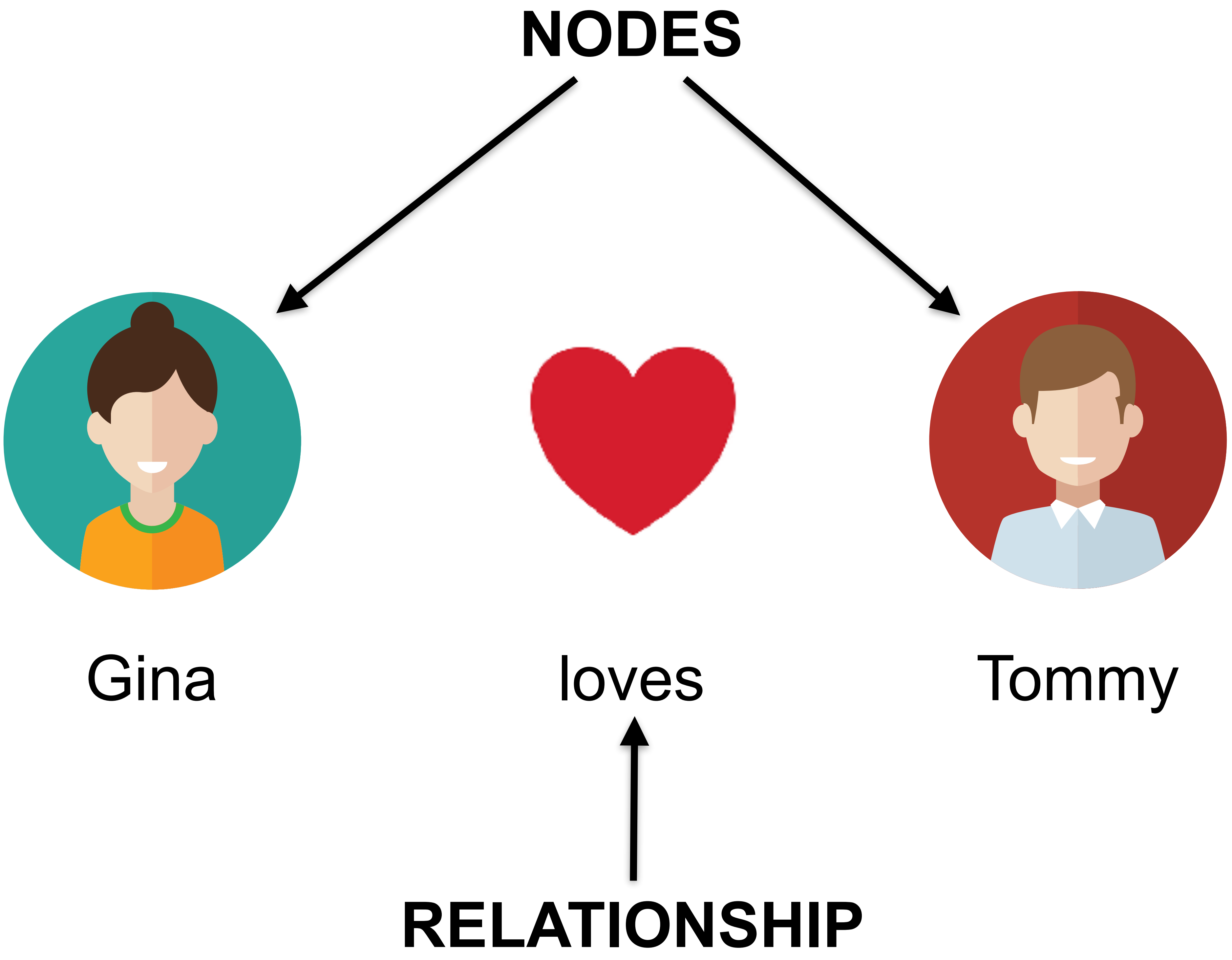
- Employ some graph representation model to store data in a graph
- Great for identifying relationships
- Great for joins
    - Some SQL queries might be pages long while the equivalent CYPHER query may be only a few lines
- Flexible/optional schema
- Some common operations are not efficient

Property Graph Model

- Used to model data in graph databases including Neo4j
- Nodes and **directed** relationships
- Node and relationship properties
- Node and relationship Labels

Thanks to https://neo4j.com

Property Graph Model



NODES

Gina

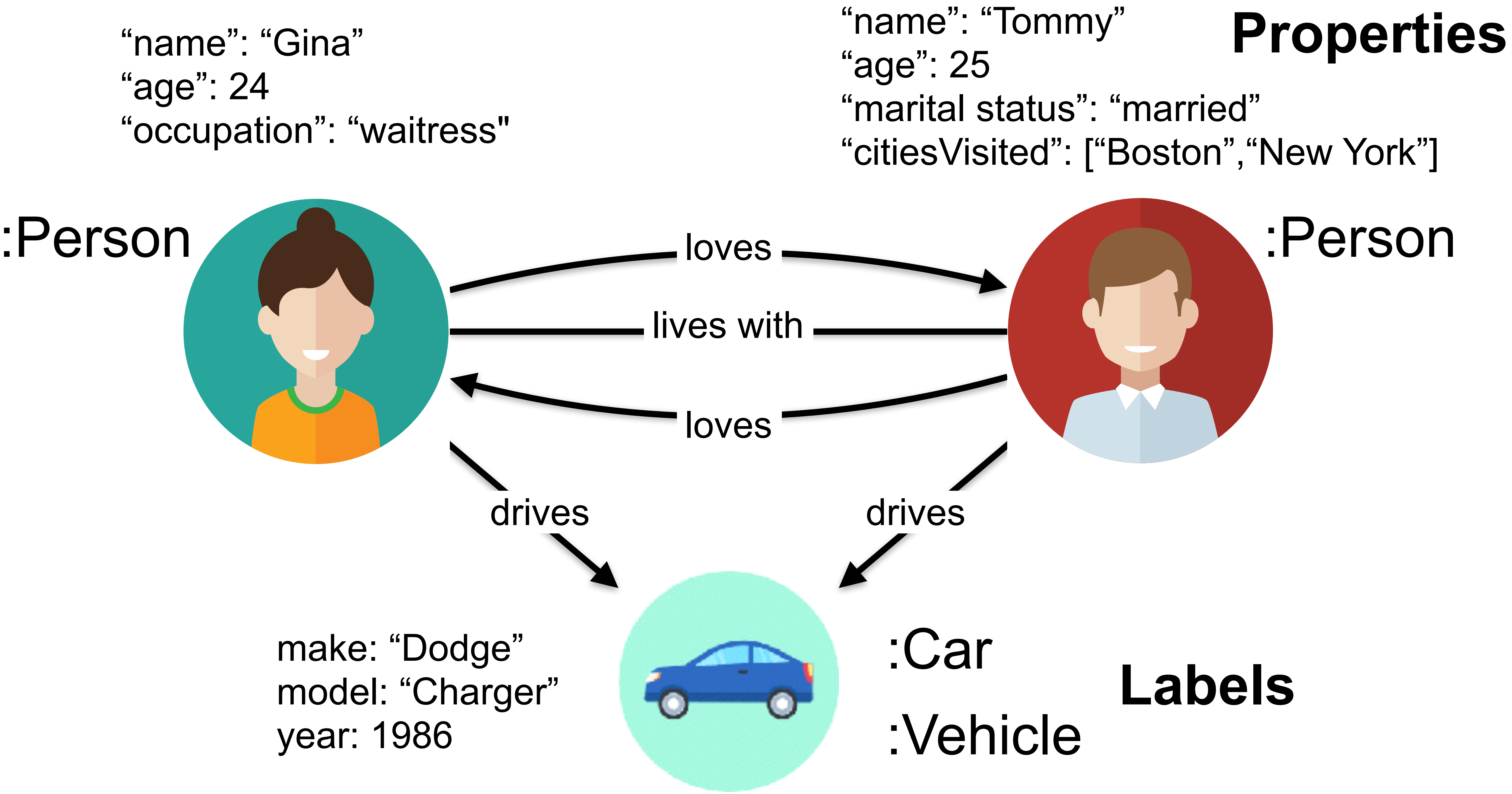
loves

Tommy

RELATIONSHIP

Thanks to https://www.freepik.com/free-vector/user-avatars-pack_762498.htm

Property Graph Model



Properties

“name”: “Gina”
“age”: 24
“occupation”: “waitress"

:Person

“name”: “Tommy”
“age”: 25
“marital status”: “married”
“citiesVisited”: [“Boston”,“New York”]

:Person

make: “Dodge”
model: “Charger”
year: 1986

:Car
:Vehicle

Labels

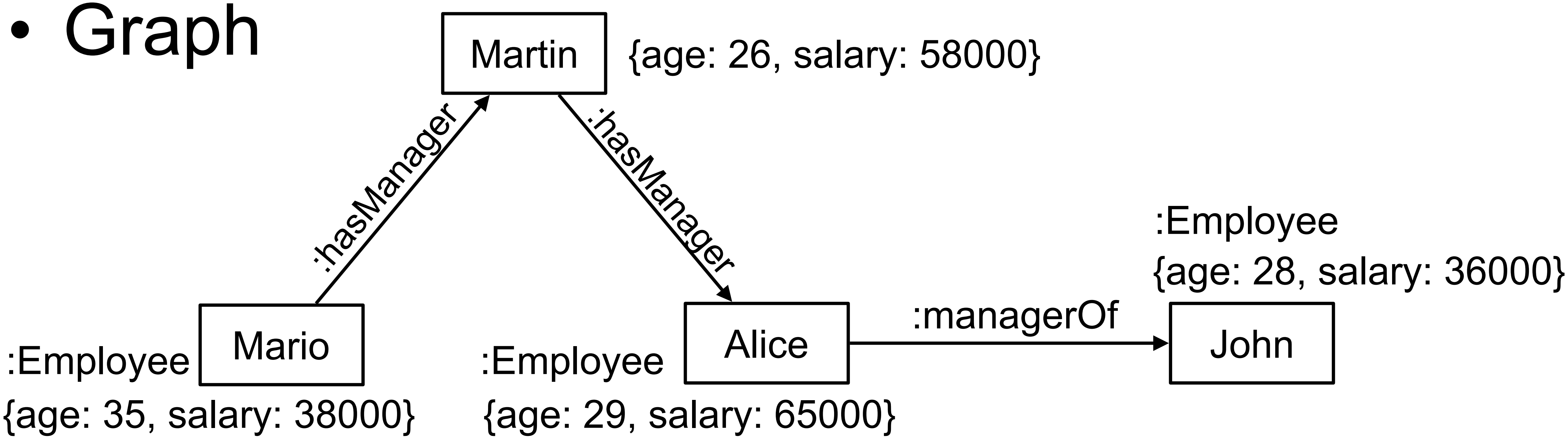
Thanks to https://www.freepik.com/free-vector/user-avatars-pack_762498.htm

Relational to Property Graph Model

- Tables

Employee			
Name	**Age**	**Salary**	**Manager**
Alice	29	65000	null
Martin	26	58000	Alice
John	28	36000	Alice
Mario	35	38000	Martin

- Graph



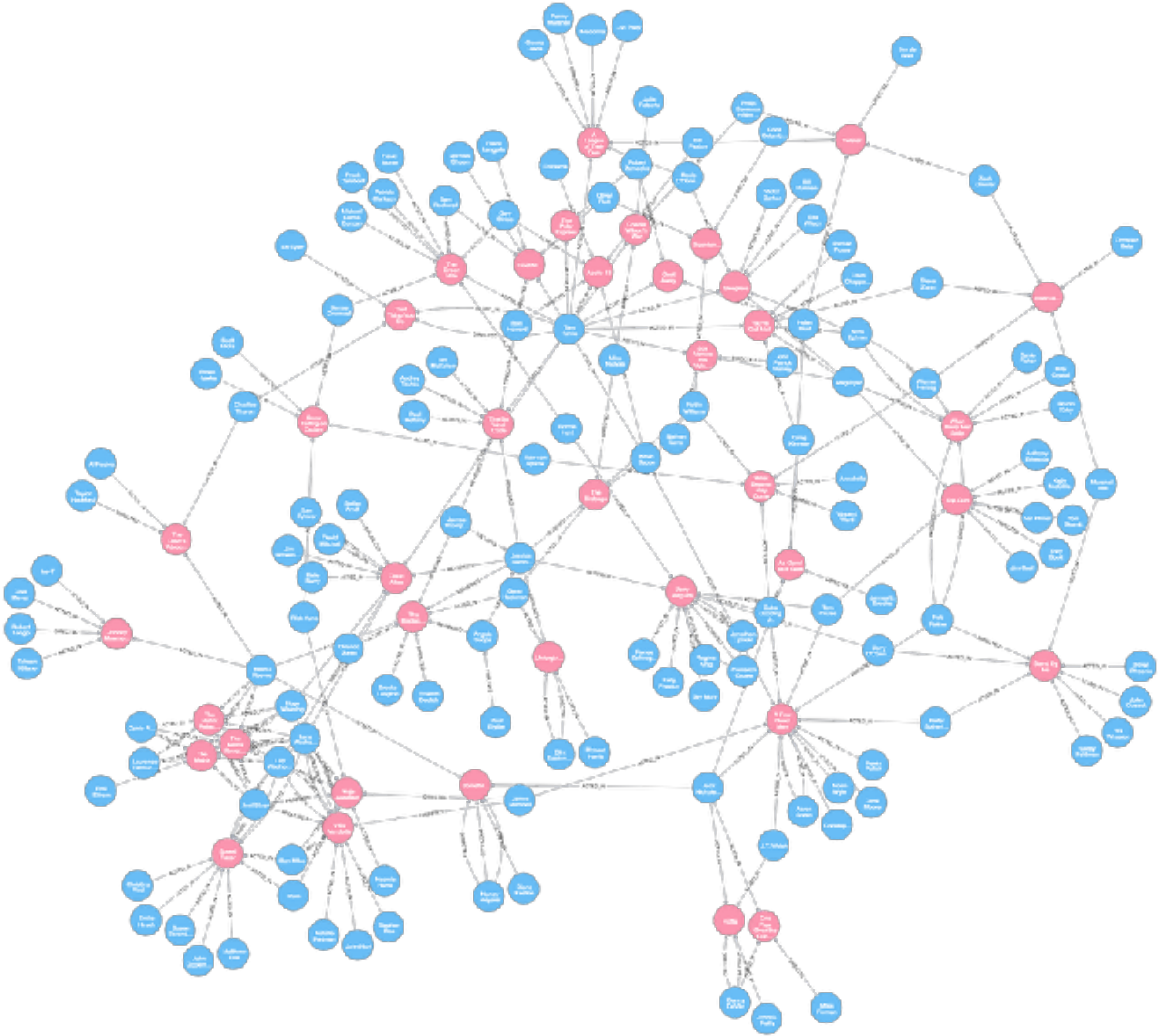
Cypher

- Pattern-Matching Query Language
- Declarative: Say what you want, not how
- Borrows ideas from well known query languages

Cypher Query Structure

- **MATCH <pattern>**
  WHERE <condition>
  RETURN <expr>
- MATCH describes the pattern
- WHERE enforces constraints
- RETURN | CREATE | DELETE | MERGE
  return the result of modify the graph

Sample Graph - Movies



Thanks to https://neo4j.com

Cypher - MATCH

- Find the titles of all movies that Tom Hanks has acted in

```
MATCH (a:Person)-[:ACTED_IN]->(b:Movie)
WHERE a.name = 'Tom Hanks'
RETURN b.title
```

b.title
"Charlie Wilson's War"
"The Polar Express"
"A League of Their Own"
"Cast Away"
"Apollo 13"
"The Green Mile"
"The Da Vinci Code"
"Cloud Atlas"
"That Thing You Do"
"Joe Versus the Volcano"
"Sleepless in Seattle"
"You've Got Mail"

Cypher - MATCH - Multiple patterns

- Find the titles of all movies that Tom Hanks has directed AND acted in

```
MATCH (a:Person)-[:ACTED_IN]-(b:Movie),
(a:Person)-[:DIRECTED]-(b:Movie)
WHERE a.name = 'Tom Hanks'
RETURN b.title
```

b.title
That Thing You Do

Cypher - RETURN - Aggregation

- Find all actor names along with all the movie titles they have acted in

```
MATCH (a:Person)-[:ACTED_IN]->(b:Movie)
RETURN a.name, collect(b.title)
```

a.name	collect(b.title)
"Charlize Theron"	["That Thing You Do", "The Devil's Advocate"]
"Orlando Jones"	["The Replacements"]
"Patricia Clarkson"	["The Green Mile"]
"Tom Skerritt"	["Top Gun"]
"Helen Hunt"	["Twister", "Cast Away", "As Good as It Gets"]
"Victor Garber"	["Sleepless in Seattle"]
"Ice-T"	["Johnny Mnemonic"]
	...

Cypher - OPTIONAL MATCH

- Print the names of all actors. If they have acted in a movie the title of which contains the word "Good" print the movie title as well.

```
MATCH (a:Person)-[:ACTED_IN]->()
OPTIONAL MATCH (a)-[:ACTED_IN]->(b)
WHERE b.title CONTAINS 'Good'
RETURN DISTINCT a.name, b.title
```

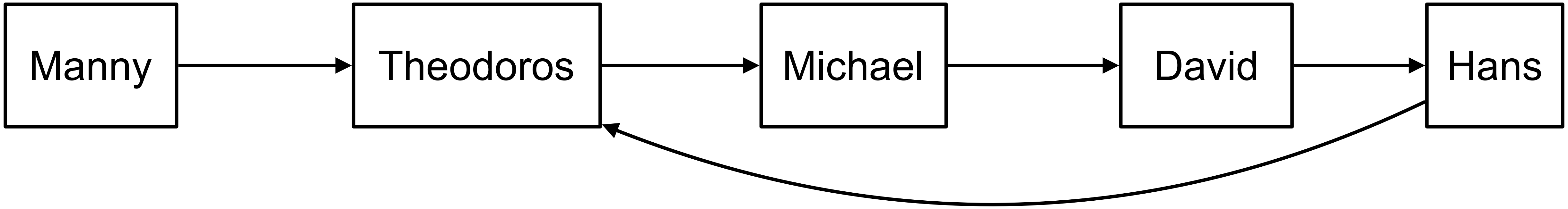
Cypher - OPTIONAL MATCH

- Print the names of all actors. If they have acted in a movie the title of which contains the word "Good" print the movie title as well.

a.name	b.title
"Keanu Reeves"	null
"Carrie-Anne Moss"	null
"Laurence Fishburne"	null
"Hugo Weaving"	null
"Emil Eifrem"	null
"Charlize Theron"	null
"Al Pacino"	null
"Tom Cruise"	"A Few Good Men"
"Jack Nicholson"	"As Good as It Gets"
"Jack Nicholson"	"A Few Good Men"
"Demi Moore"	"A Few Good Men"
"Kevin Bacon"	"A Few Good Men"

...

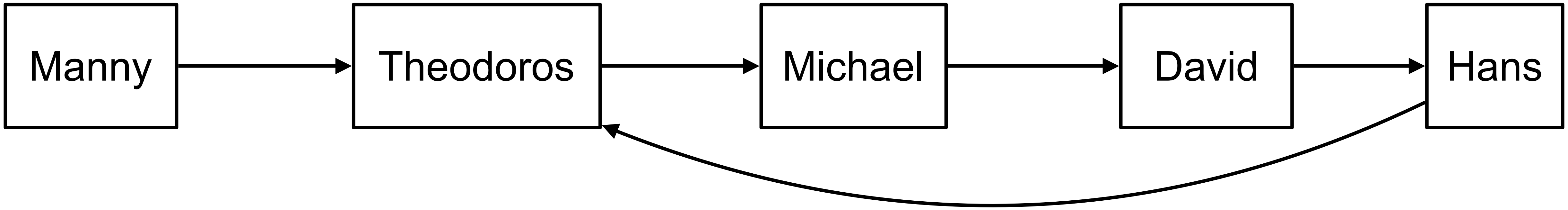
Cypher - Variable Length Paths



- Find all paths from "Theodoros" to "David"

```
MATCH p=(a)-[:KNOWS*]->(b)
WHERE a.name = 'Theodoros'
AND b.name = 'David'
RETURN p
```

Cypher - Variable Length Paths



- Find the length of the shortest path from "Theodoros" to "David"

```
MATCH p=shortestPath((a)-[:KNOWS*]->(b))
WHERE a.name = 'Theodoros'
AND b.name = 'David'
RETURN length(p)
```

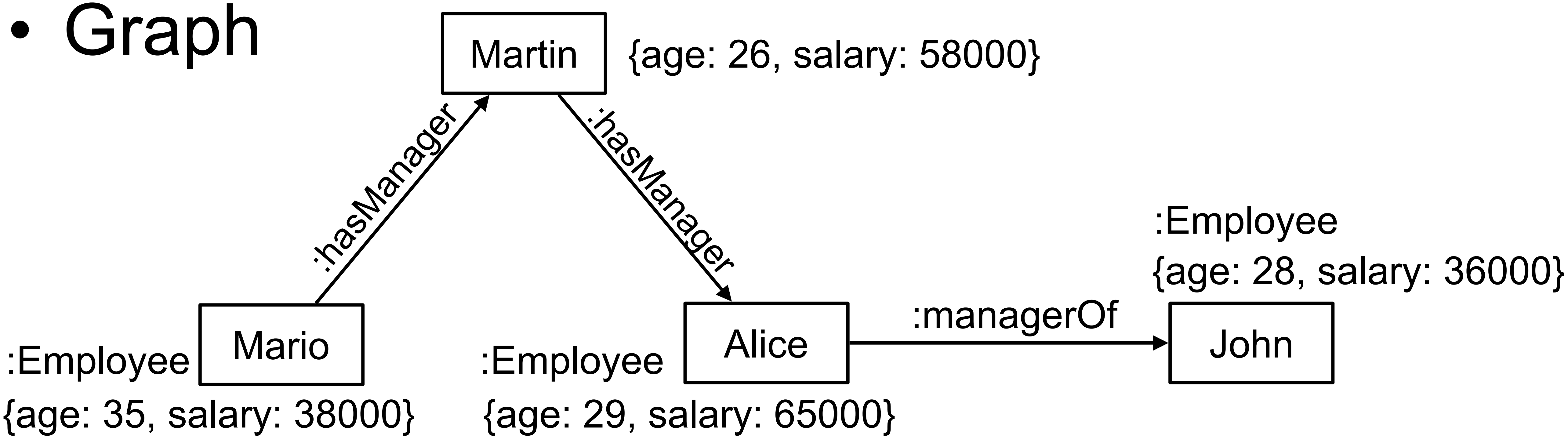
2

SQL vs Cypher

- Tables

Employee			
Name	**Age**	**Salary**	**Manager**
Alice	29	65000	null
Martin	26	58000	Alice
John	28	36000	Alice
Mario	35	38000	Martin

- Graph



Martin {age: 26, salary: 58000}

:hasManager

:hasManager

:Employee {age: 28, salary: 36000}

:managerOf

:Employee Mario {age: 35, salary: 38000}

:Employee Alice {age: 29, salary: 65000}

John

SQL vs Cypher

- What is the salary of the manager of Mario?

- SQL

```
SELECT b.salary
FROM employee AS a, employee AS b
WHERE a.name='Mario'AND a.manager=b.name
```

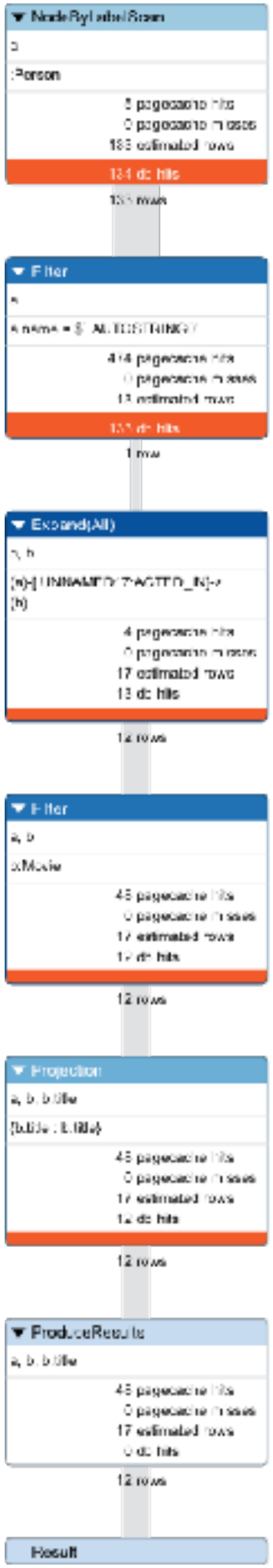
- CYPHER

```
MATCH ({name: 'Mario'})-(:hasManager)->(b)
RETURN b.salary
```

Query Processing in Neo4j

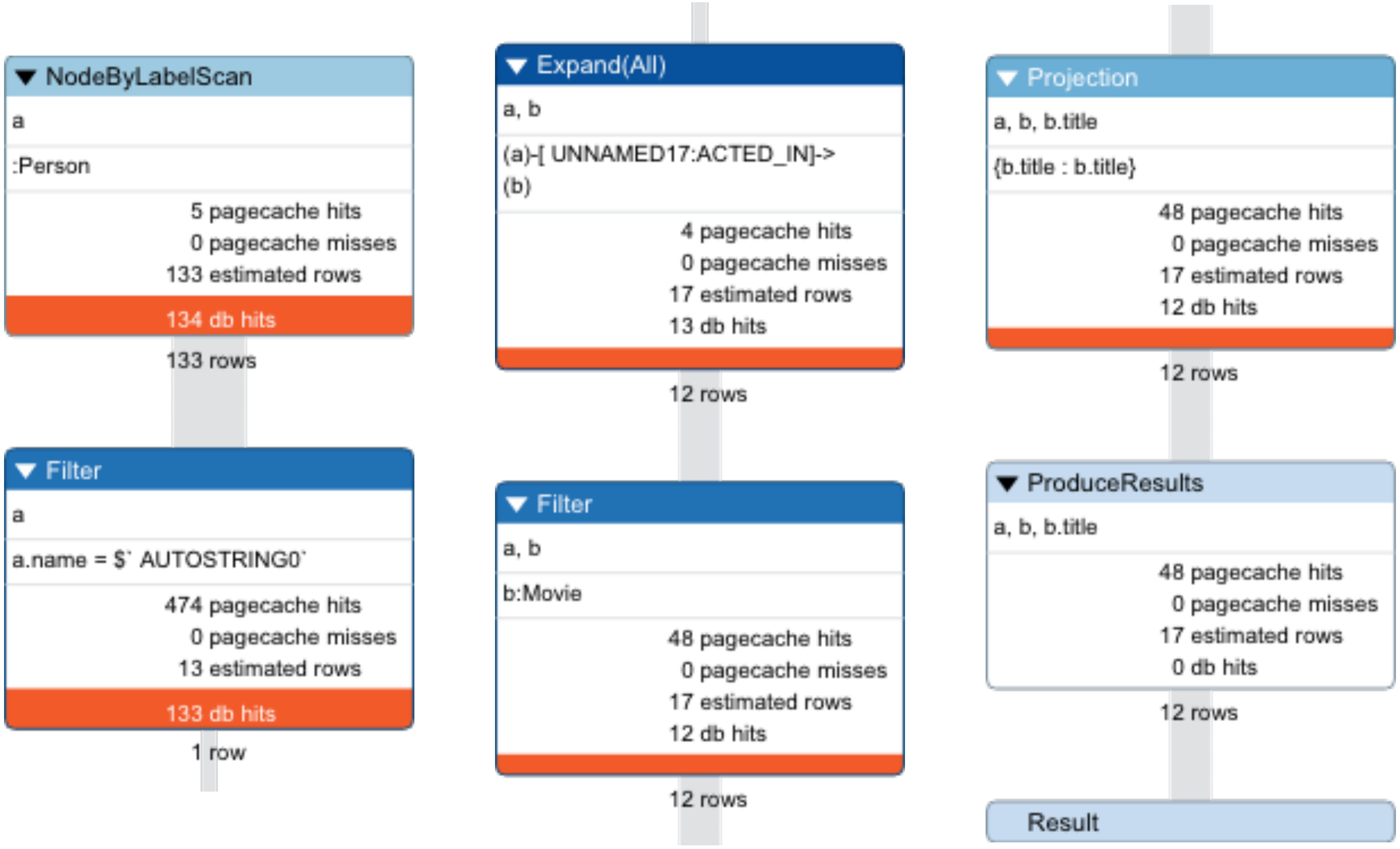
- Find the titles of all movies that Tom Hanks has acted in

```
MATCH (a:Person)-[:ACTED_IN]->(b:Movie)
WHERE a.name = 'Tom Hanks'
RETURN b.title
```



Thanks to https://neo4j.com

Query Processing in Neo4j



▼ NodeByLabelScan

a

:Person

5 pagecache hits
0 pagecache misses
133 estimated rows

134 db hits

133 rows

▼ Filter

a

a.name = $` AUTOSTRING0`

474 pagecache hits
0 pagecache misses
13 estimated rows

133 db hits

1 row

▼ Expand(All)

a, b

(a)-[UNNAMED17:ACTED_IN]->
(b)

4 pagecache hits
0 pagecache misses
17 estimated rows
13 db hits

12 rows

▼ Filter

a, b

b:Movie

48 pagecache hits
0 pagecache misses
17 estimated rows
12 db hits

12 rows

▼ Projection

a, b, b.title

{b.title : b.title}

48 pagecache hits
0 pagecache misses
17 estimated rows
12 db hits

12 rows

▼ ProduceResults

a, b, b.title

48 pagecache hits
0 pagecache misses
17 estimated rows
0 db hits

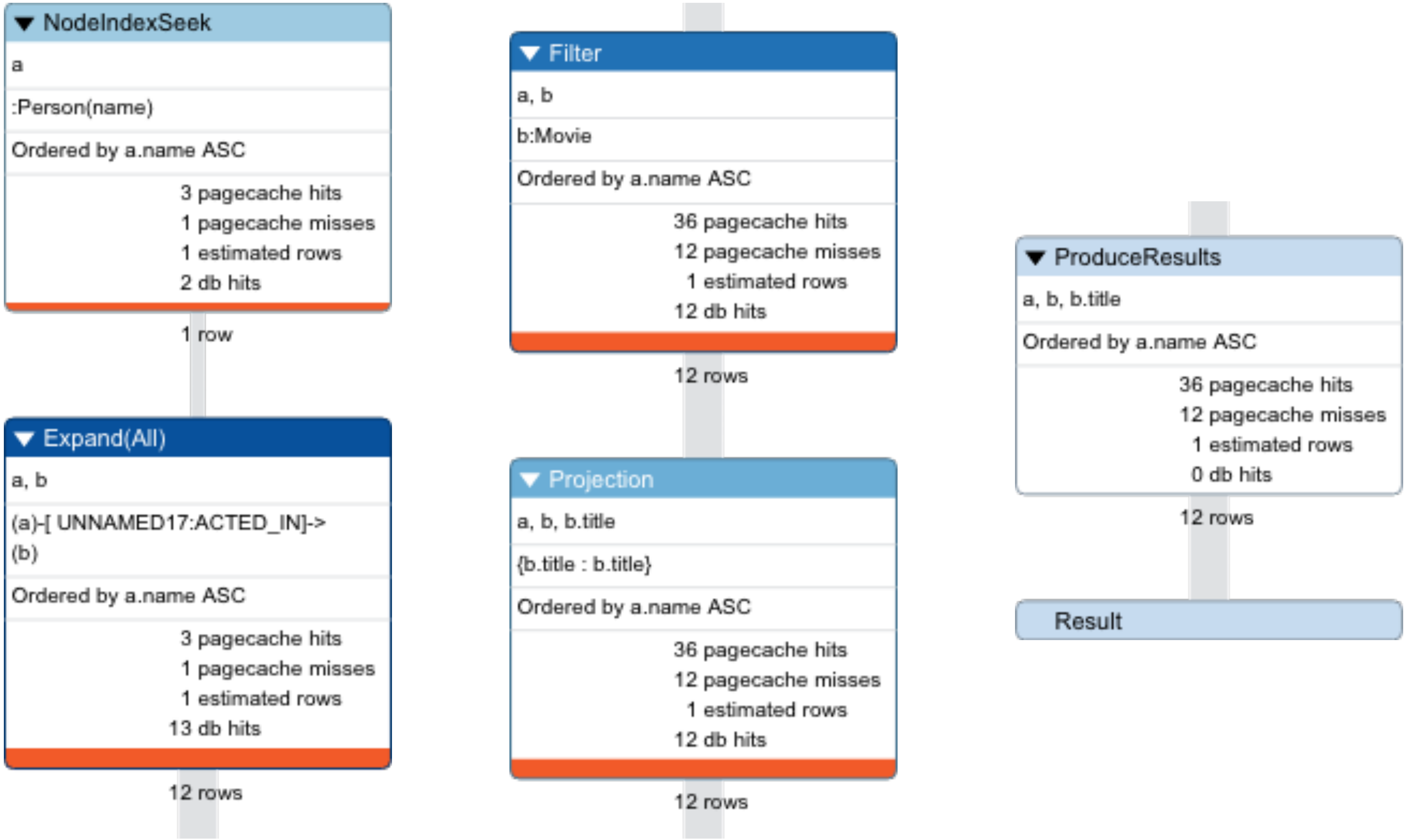
12 rows

Result

Without index on 'name'

Thanks to https://neo4j.com

Query Processing in Neo4j



▼ NodeIndexSeek

a

:Person(name)

Ordered by a.name ASC

3 pagecache hits
1 pagecache misses
1 estimated rows
2 db hits

1 row

▼ Expand(All)

a, b

(a)-[UNNAMED17:ACTED_IN]->
(b)

Ordered by a.name ASC

3 pagecache hits
1 pagecache misses
1 estimated rows
13 db hits

12 rows

▼ Filter

a, b

b:Movie

Ordered by a.name ASC

36 pagecache hits
12 pagecache misses
1 estimated rows
12 db hits

12 rows

▼ Projection

a, b, b.title

{b.title : b.title}

Ordered by a.name ASC

36 pagecache hits
12 pagecache misses
1 estimated rows
12 db hits

12 rows

▼ ProduceResults

a, b, b.title

Ordered by a.name ASC

36 pagecache hits
12 pagecache misses
1 estimated rows
0 db hits

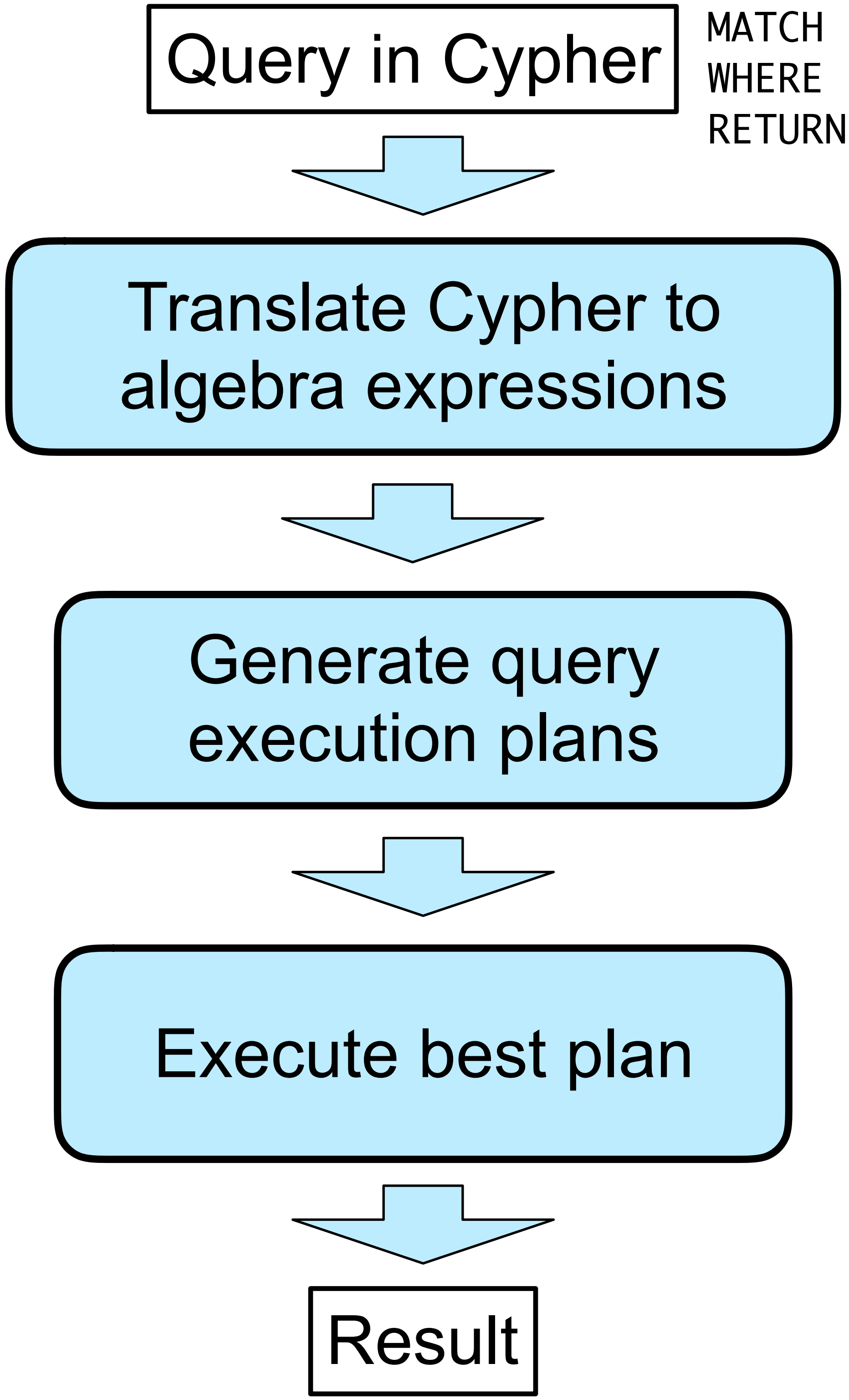
12 rows

Result

With index on 'name'

Thanks to https://neo4j.com

Query Processing - Flowchart



Query in Cypher

Translate Cypher to algebra expressions

Generate query execution plans

Execute best plan

Result

```
MATCH (a:Person)-[:ACTED_IN]->(b:Movie)
WHERE a.name = 'Tom Hanks'
RETURN b.title
```

b.title
"Charlie Wilson's War"
"The Polar Express"
"A League of Their Own"
"Cast Away"
"Apollo 13"
"The Green Mile"
"The Da Vinci Code"
"Cloud Atlas"
"That Thing You Do"
"Joe Versus the Volcano"
"Sleepless in Seattle"
"You've Got Mail"

Graph Database: The Future (our ongoing work)

Graph vs Relational Databases

- Graph databases are clearly not yet mature enough to compete with RDBMS
- Many graph-oriented operations are executed faster in relational than graph DBMS
- Our current work:
  - Indexing structures for graph-oriented operations
  - Cost-based query optimisation
  - Graph analytics
  - and more

Graph vs Relational Databases

- Graph databases are clearly not yet mature enough to compete with RDBMS
- Many graph-oriented operations are executed faster in relational than graph DBMS
- Our current work:
  - **Indexing structures for graph-oriented operations**
  - Cost-based query optimisation
  - Graph analytics
  - and more

Traversal Indices on Neo4j

- Adapt preprocessing-based methods from the memory to the database
- Current implementations
  - ALT (A*-search - Landmarks - Triangle inequality)
  - Contraction Hierarchies

```
CREATE TRAVERSAL INDEX ON :RELTYPE('myweight')
```

ALT Algorithm - Triangle Inequality

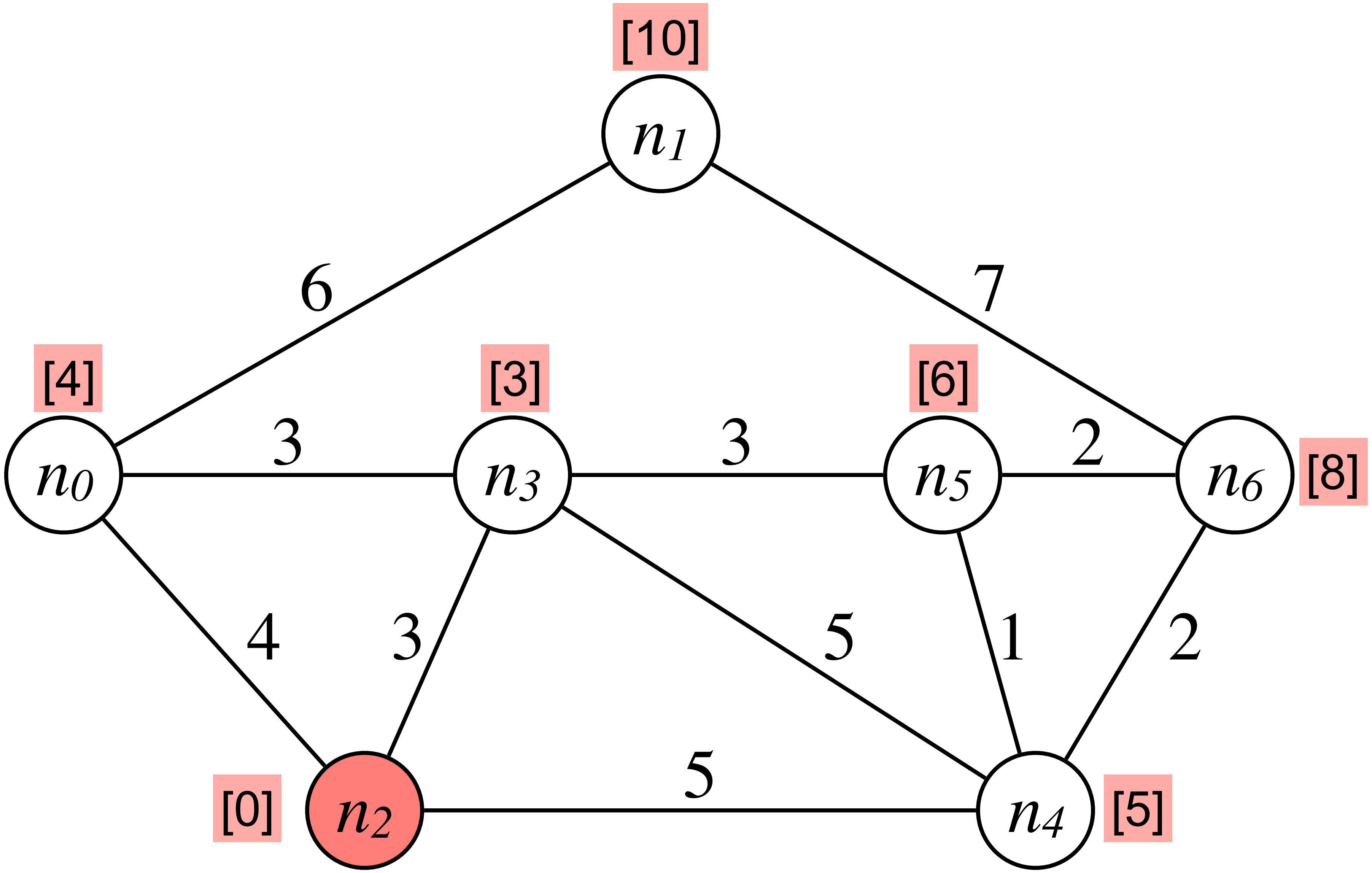
- The network distance satisfies the triangle inequality

- Given a graph $G = (N, E)$ and nodes $u,v,w \in N$

$$dist(u,v) \leq dist(u,w) + dist(w,v)$$

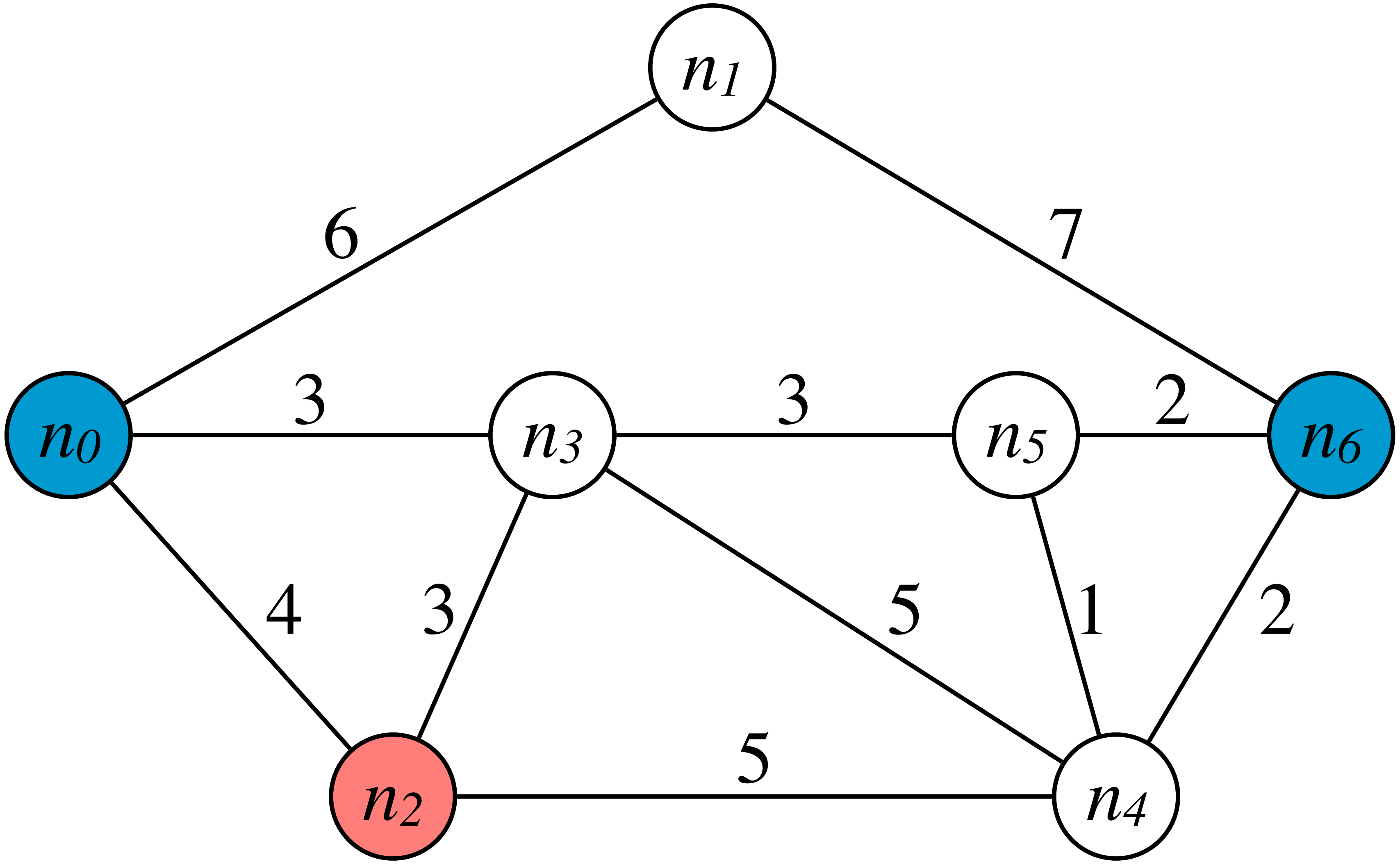
ALT Algorithm - Triangle Inequality

- Shortest path $p(n_0 \rightarrow n_5)$
  Landmarks: n_2



ALT Algorithm - Triangle Inequality

- Shortest path $p(n_0 \rightarrow n_5)$
  Landmarks: n_2



ALT Algorithm - Triangle Inequality

- The network distance satisfies the triangle inequality

- Given a graph $G = (N, E)$ and nodes $u,v,w \in N$

$$dist(u, v) \leq dist(u, w) + dist(w, v)$$

- The equality applies when w is on the shortest path from u to v

ALT Algorithm - Upper Bounds

- Let l be an arbitrary node chosen as landmark and u-v be a random pair of nodes:

$$dist(u,v) \leq dist(u,l) + dist(l,v)$$

ALT Algorithm - Lower Bounds

- Let *l* be an arbitrary node chosen as landmark and *u*-*v* be a random pair of nodes:

$$dist(u,l) \leq dist(u,v) + dist(v,l) \Rightarrow$$

$$\Rightarrow dist(u,l) - dist(v,l) \leq dist(u,v)$$

AND

$$dist(l,v) \leq dist(l,u) + dist(u,v) \Rightarrow$$

$$\Rightarrow dist(l,v) - dist(l,u) \leq dist(u,v)$$

ALT Algorithm - Lower Bounds

- Given a query from s to t lower bound, $\forall u \in N$

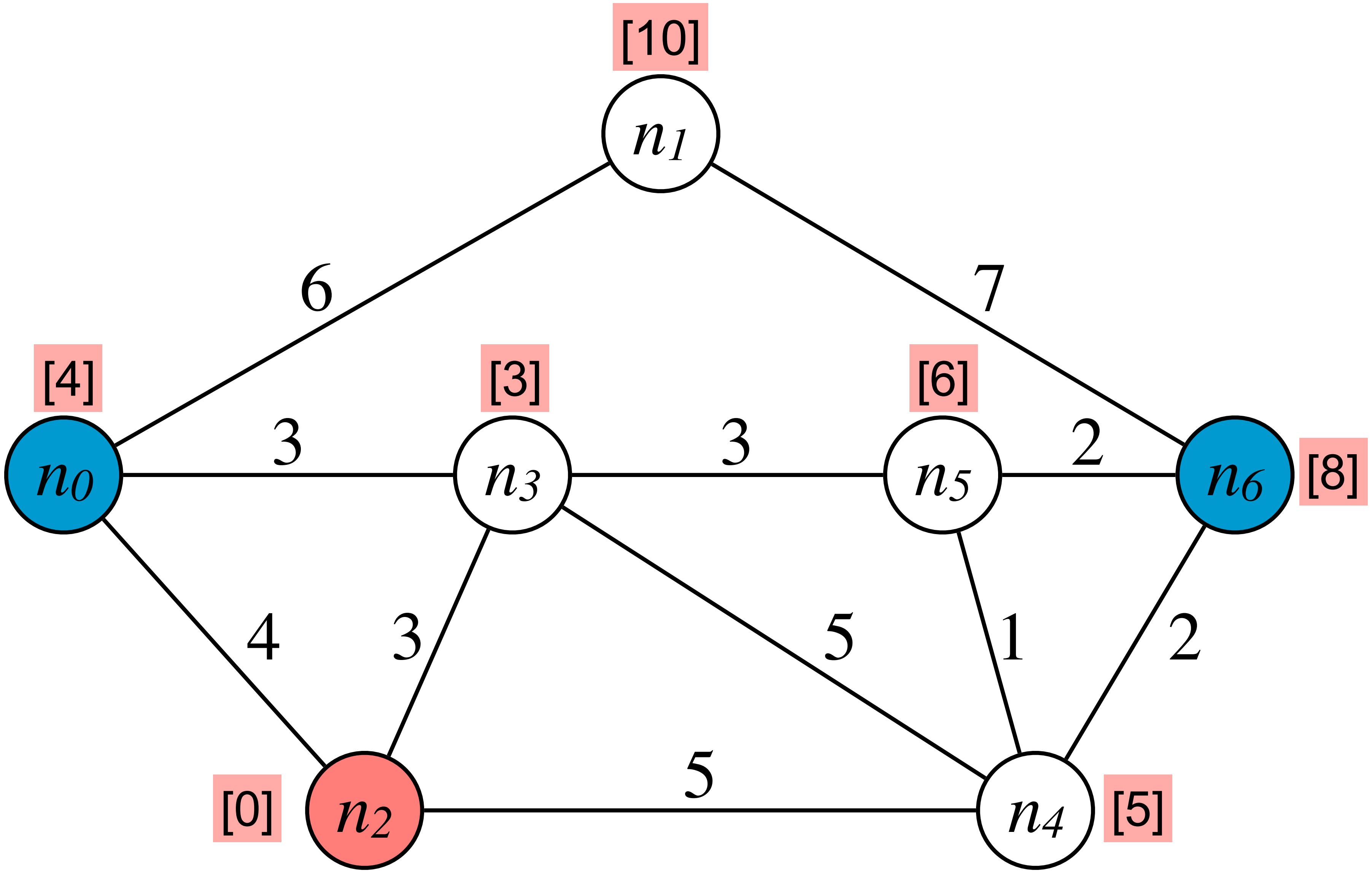
$$h(u) = max(dist(u,l) - dist(t,l), dist(l,t) - dist(l,u))$$

- For undirected graphs

$$h(u) = |dist(u,l) - dist(t,l)|$$

ALT Algorithm - Triangle Inequality

- Shortest path $p(n_0 \rightarrow n_5)$
Landmarks: n_2



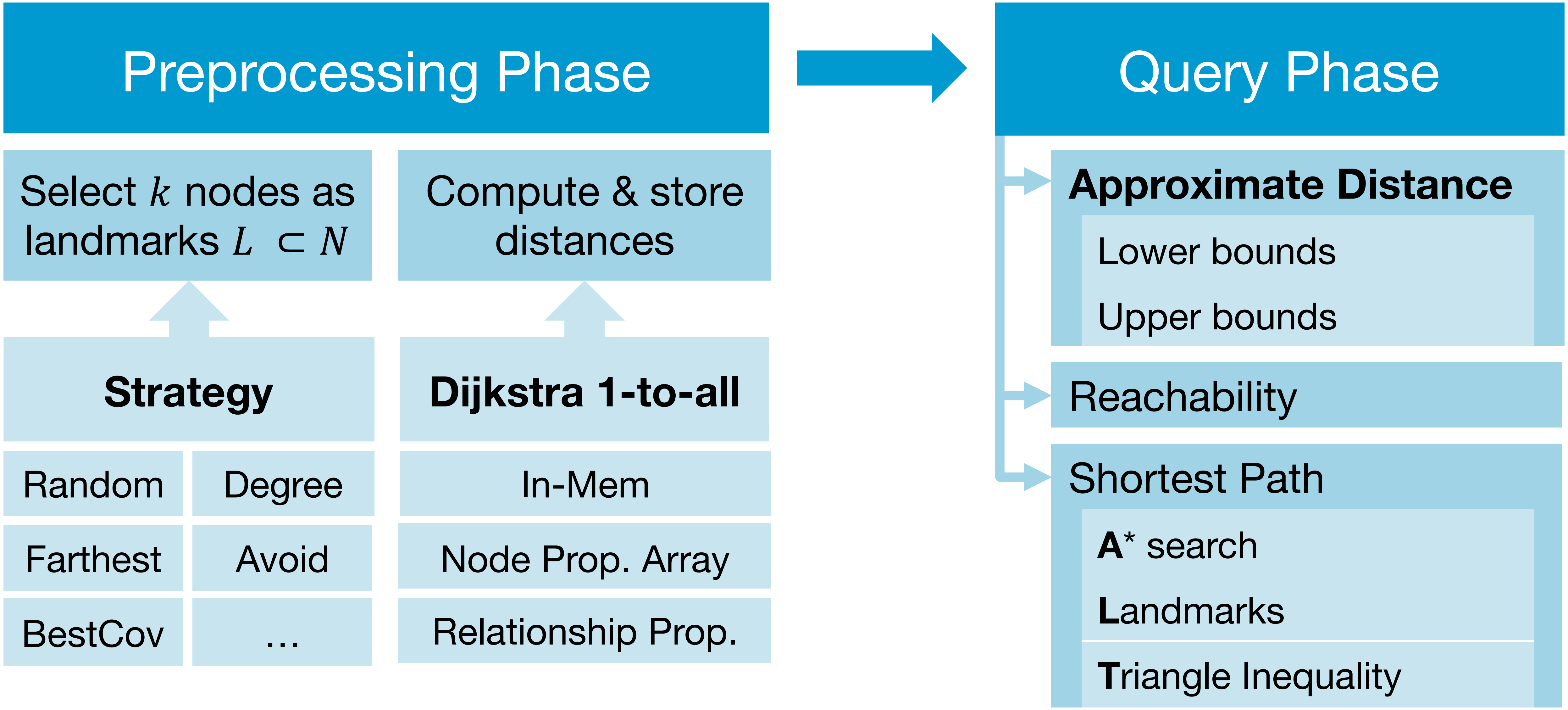
ALT Algorithm - Bounds

- For undirected graphs

$$|dist(u,l) - dist(t,l)| \leq dist(u,v) \leq dist(u,l) + dist(v,l)$$

- Adjustment is needed for directed graphs
- Bounds can be used as-is for approximate distance query processing
- Lower bounds can be used by A*-search

Landmark Embedding on Neo4j



Preprocessing Phase

Select k nodes as landmarks $L \subset N$

Compute & store distances

Strategy

Random

Degree

Farthest

Avoid

BestCov

…

Dijkstra 1-to-all

In-Mem

Node Prop. Array

Relationship Prop.

Query Phase

Approximate Distance

Lower bounds

Upper bounds

Reachability

Shortest Path

A* search

Landmarks

Triangle Inequality

Landmark Embedding on Neo4j

- Cypher query for relationship-based implementation

```
MATCH
  (s)-[rsL:L_REL]->(l:L), (l:L)-[rLs:L_REL]->(s),
  (t)-[rtL:L_REL]->(l:L), (l:L)-[rLt:L_REL]->(t)
WHERE s.name = 's' AND t.name = 't'
UNWIND
  [rsL.dist - rtL.dist, rLt.dist - rLs.dist] AS est
RETURN max(est) as tightestLower
```

What's next

- Support for multi-labeled graphs
- Support for dynamic graphs and automated index maintenance
- Graph statistics for landmark selection (number of landmarks, locations etc.)
  - The type of the underlying graph matters

Graph Databases - Conclusion

- Graph databases are a fairly new and very promising technology
- Graph analysis is a hot topic at the moment
- Premature technology
  - A lot of work needs to be done
- Can graph databases replace relational ones for general purpose scenarions?
  - Probably not but many ideas and concepts from graphs are already integrated in relational DBMS

Credits

1. A. Jayaraman, K. Jamil and H. Khan: Protein-protein integration image from *“Identifying new targets in leukemogenesis using computational approach”*, Saudi Journal of Biological Sciences, vol. 21, no. 5, 2015

Thank you!

theodoros.chondrogiannis@uni-konstanz.de