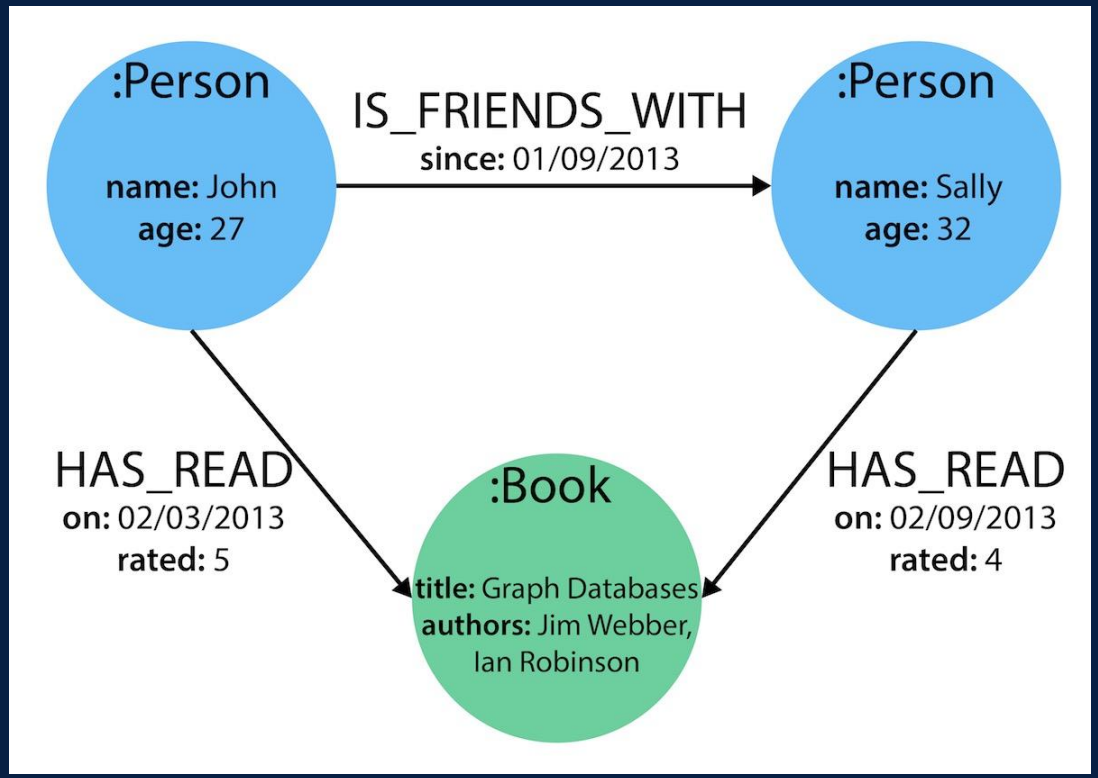
Neo4J

COSC 516 – Cloud Databases

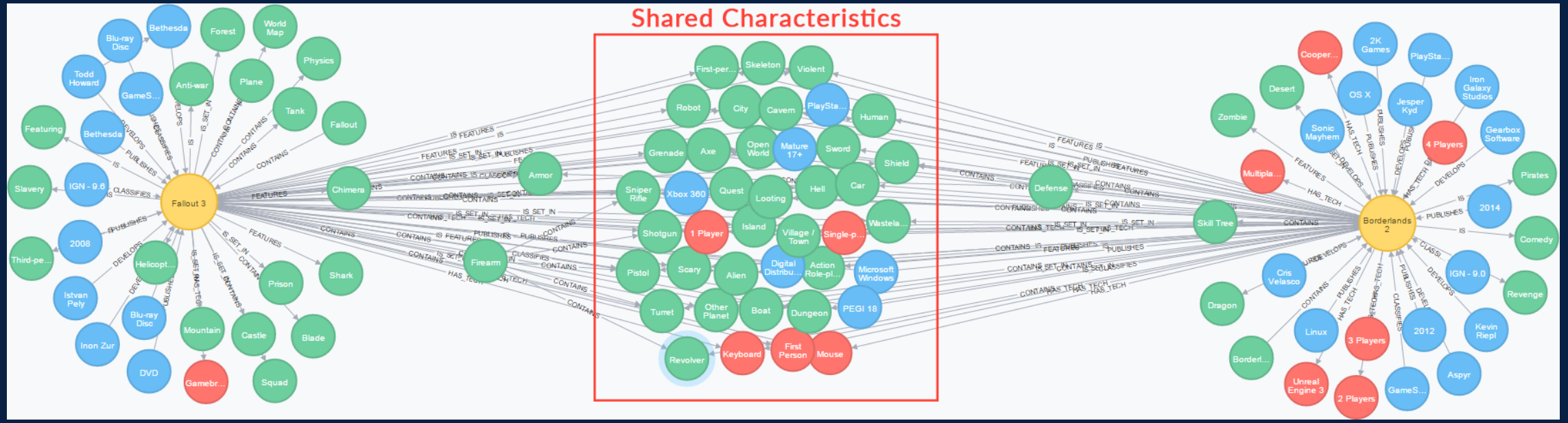
Graph Data Model

- A graph is just a collection of vertices and edges
- A set of nodes and the relationships that connect them
- Graphs represent entities as nodes and the ways in which those entities relate to the world as relationships



- **Label**: Person
- **Property**: (**key**, **value**)
  - **Key**: John
  - **Value**: 27
- **Relationships**: IS_FRIENDS_WITH, HAS_READ

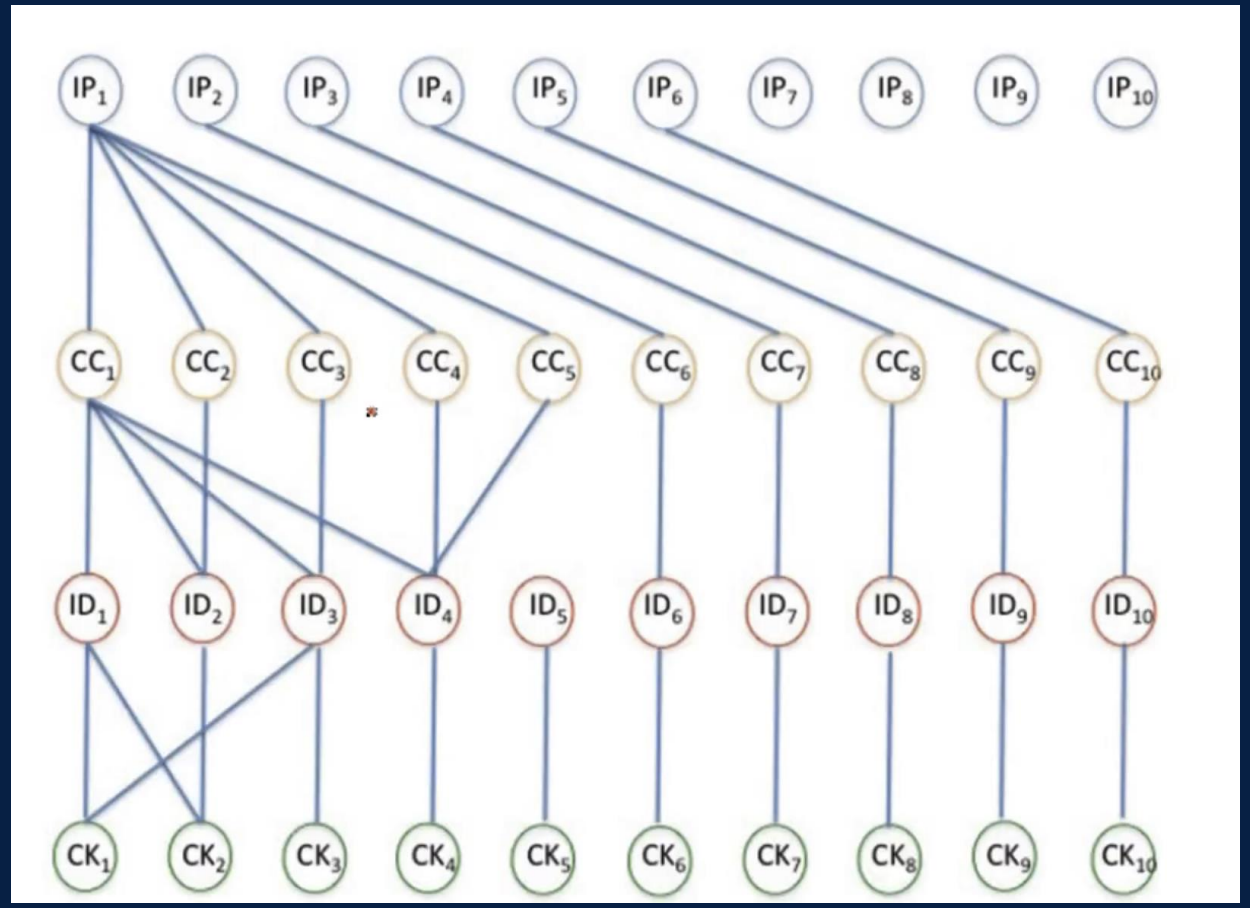
Graph Database Use Cases



Shared characteristics between video games

Graph Database Use Cases

- Fraud Detection
- Real Time Recommendation Engines
- Master Data Management
- Network and IT Operations
- Identity and Access Management



Walmart

Walmart uses Neo4j to optimize customer experience with personal recommendations

Walmart and eBay adopt graph database

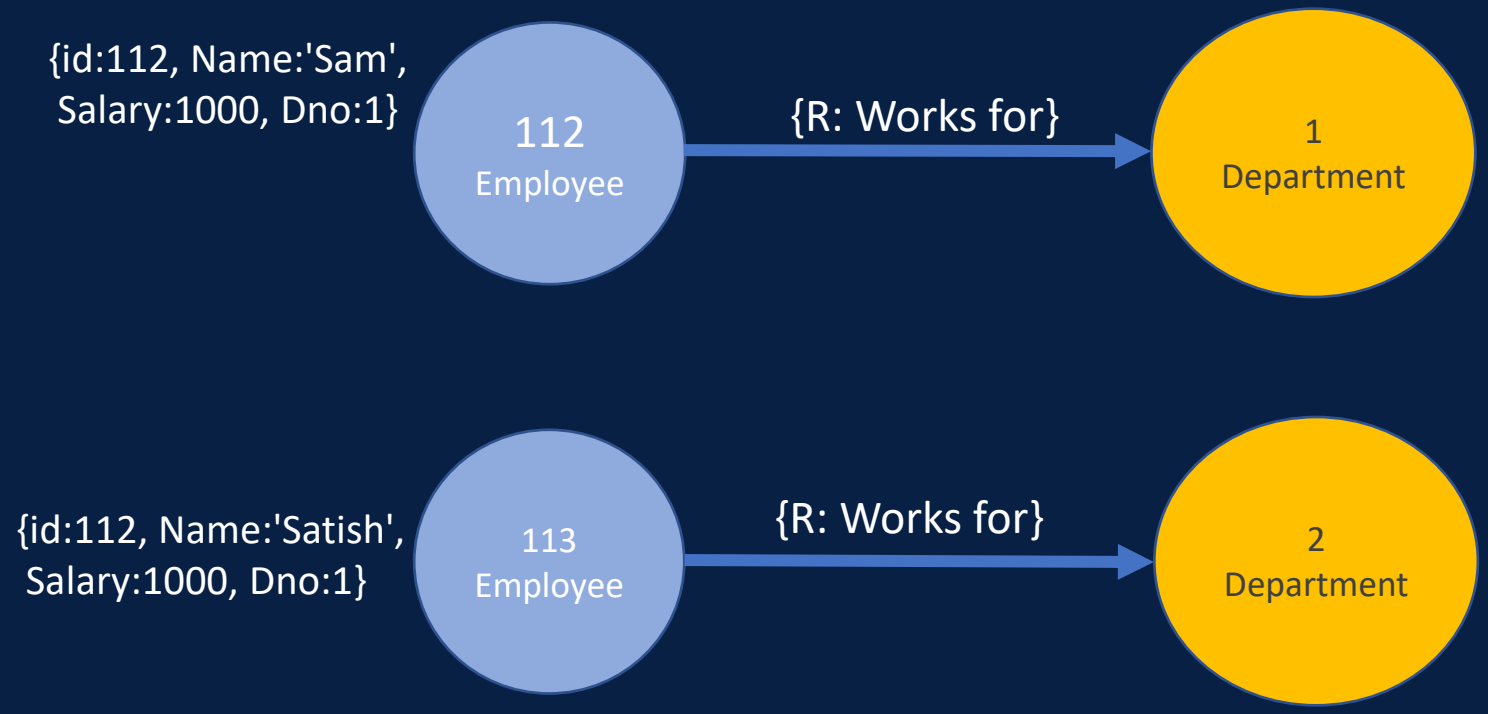
Retail giants turn to database technology to optimise operations, increase insight into customer demand and improve customer service to ultimately drive more sales

Large retailers including eBay and Walmart are using graph database Neo4j in mission-critical business applications it was announced by the database's developer Neo Technology.

Relational Data Model to Graph Data Model

Emp ID	Name	Salary	DNO
112	Sam	1000	1
113	Satish	2000	2

DID	DName
1	SCOPE
2	SBST



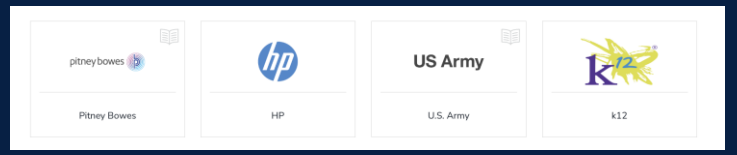
Labeled property graph

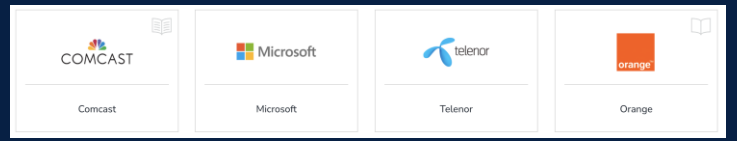
A labeled property graph has the following characteristics:

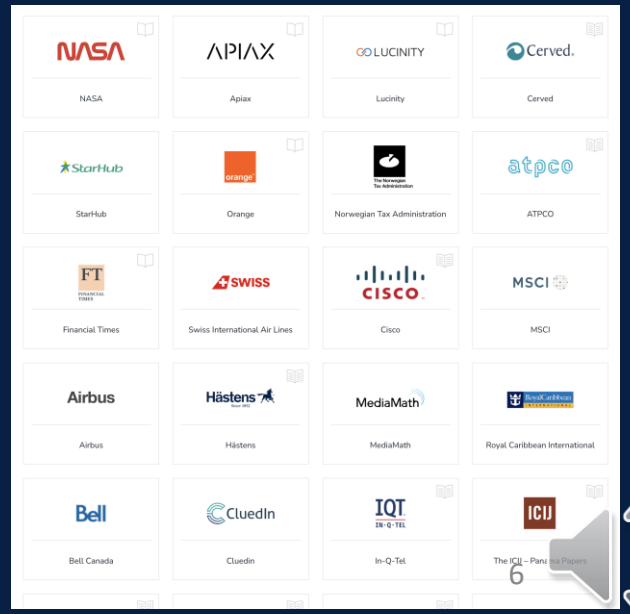
- It contains **nodes** and **relationships**
- Nodes contain **properties** (key-value pairs).
- Nodes can be labeled with one or more **labels**.
- Relationships are **named** and **directed**, and always have a **start** and **end node**
- Relationships can also contain **properties**.

Neo4j

- The development efforts on Neo4j began in 2000, when the co-founders, while on a project building a content management system, encountered a problem that couldn't be solved using relational databases – so they invented the property graph.
- Neo4j is an open-source, NoSQL, native graph database that provides an ACID-compliant transactional backend for your applications that has been publicly available since 2007.
- Neo4j is offered as a managed service via **AuraDB**. But you can also run Neo4j yourself with either Community Edition or Enterprise Edition. The Enterprise Edition includes all that Community Edition has to offer, plus extra enterprise requirements such as backups, clustering, and failover abilities.
- **Neo4j Desktop** is the new mission control center for developers. It's free with registration, and it includes a development license for Enterprise Edition.



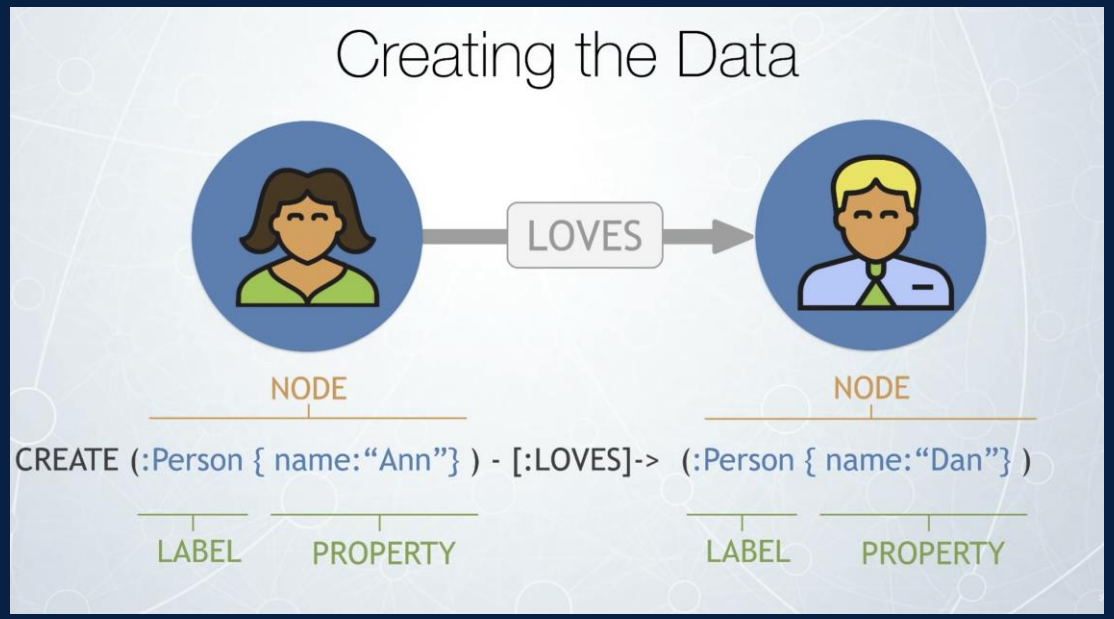


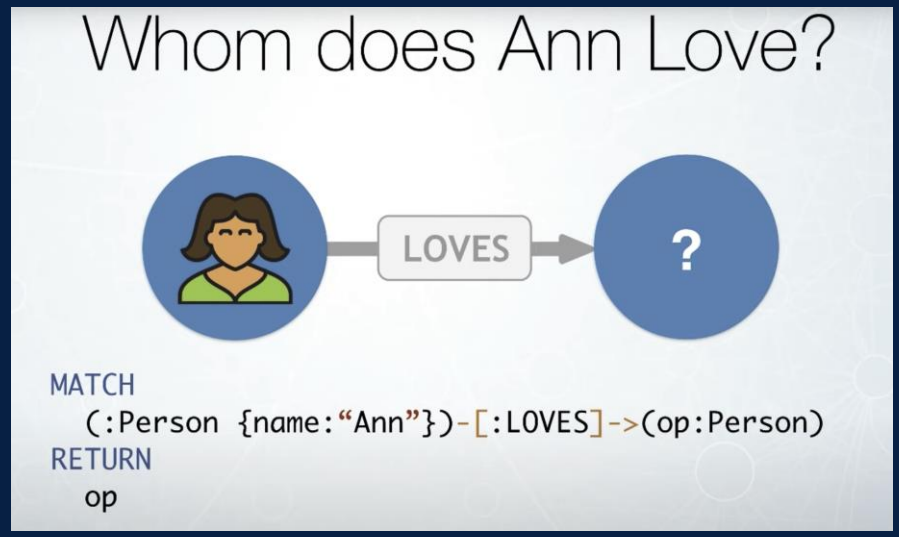


Source: https://neo4j.com/who-uses-neo4j/

Cypher Query Language

- **Cypher is Neo4j's graph query language that allows users to store and retrieve data from the graph database.**
- **Cypher is based on patterns**



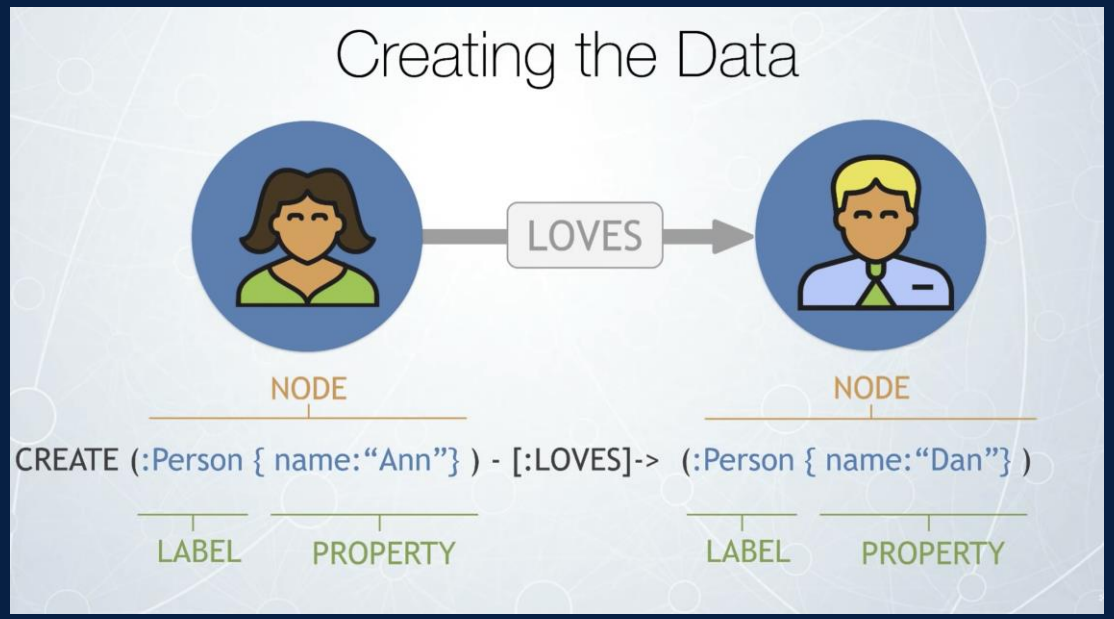


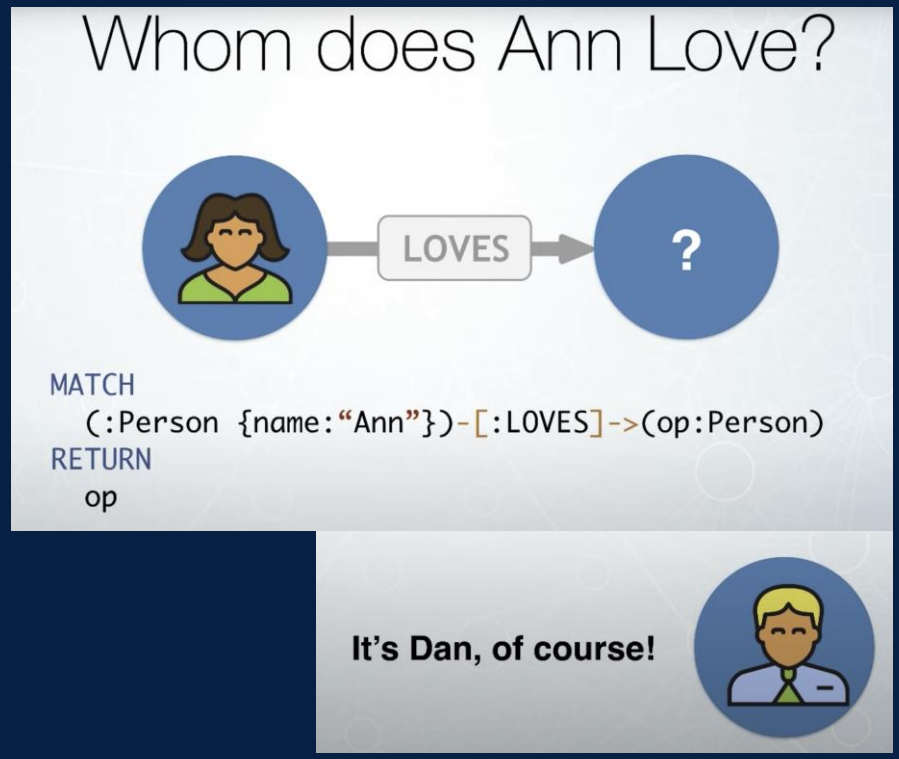
Relationship directions are specified using <, >

Source: https://neo4j.com/developer/cypher/intro-cypher/

Cypher Query Language

- **Cypher is Neo4j's graph query language that allows users to store and retrieve data from the graph database.**
- **Cypher is based on patterns**



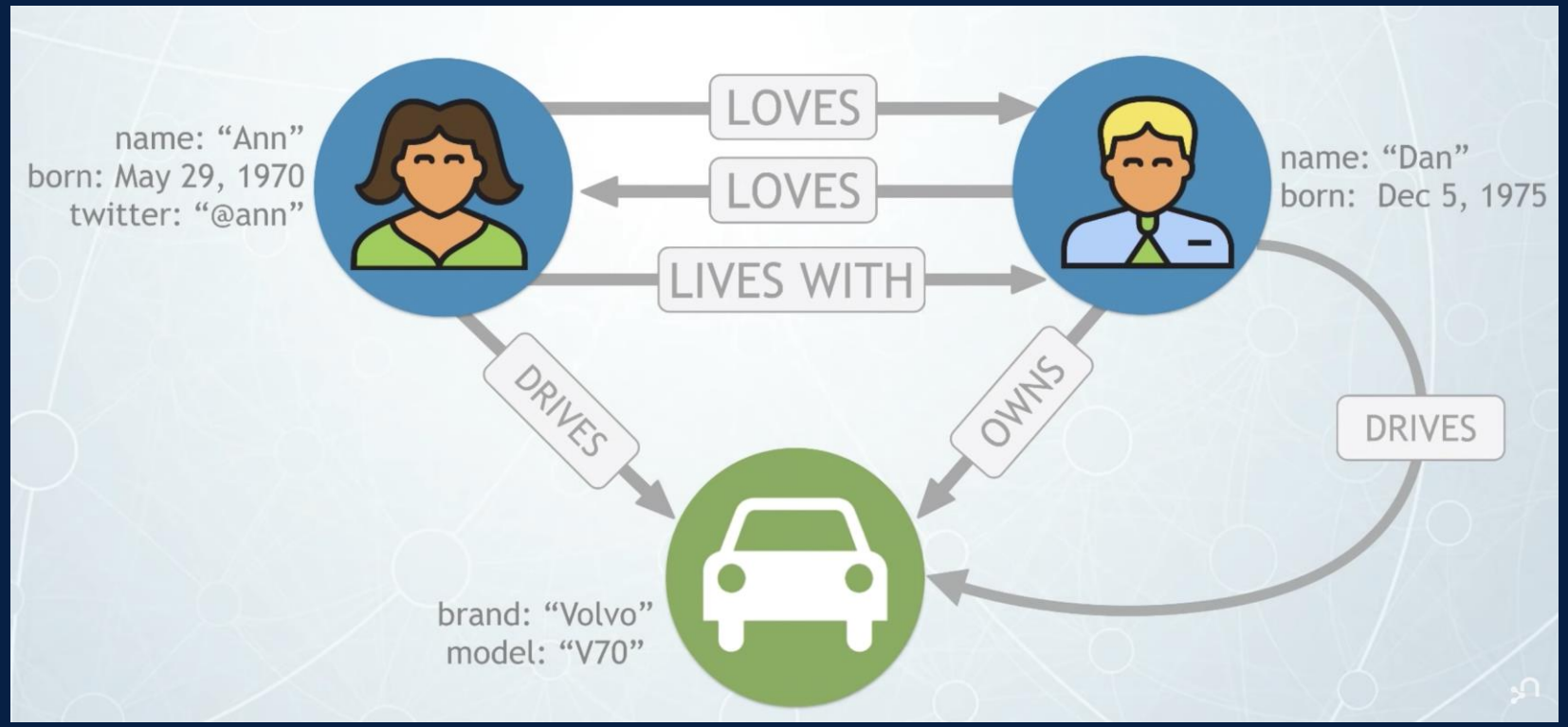


Relationship directions are specified using <, >

Source: https://neo4j.com/developer/cypher/intro-cypher/

Cypher Query Language

- **Let's say we have now created this graph**



Source: https://neo4j.com/developer/cypher/intro-cypher/

Cypher Query Language

How do I find Ann's Car?

```
MATCH
  (:Person {name: 'Ann'})-[:DRIVES]->(c:Car)
RETURN
  c
```

OR

```
MATCH
  (a:Person)-[:DRIVES]->(c:Car)
WHERE
  a.name='Ann'
RETURN
  c
```

UPDATE Clause

```
MATCH
  (a:Person)-[:DRIVES]->(c:Car)
WHERE
  a.name='Ann'
SET
  c.brand='Volvo',
  c.model='V70'
RETURN
  c
```

- Add a property to a specific node – use SET
- Remove a property for a node – use REMOVE
- Update a specific property for a node – use REMOVE and SET
- Delete a specific node – use MATCH and DELETE
- Delete all nodes – `match(n) delete(n)`
- Delete all nodes with relationships – `match(n) detach delete n`

Source: https://neo4j.com/developer/cypher/intro-cypher/

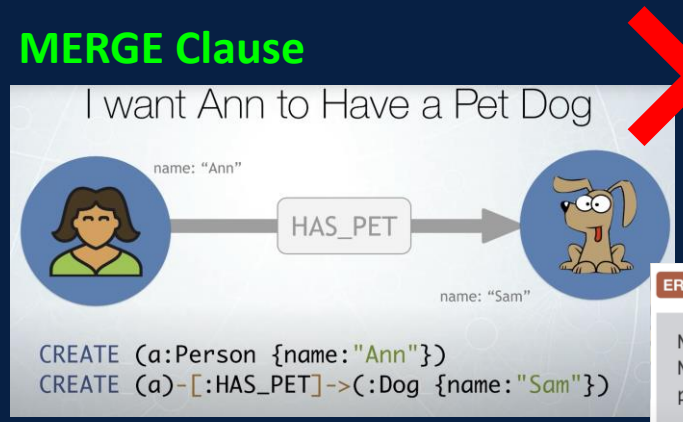
Cypher Query Language

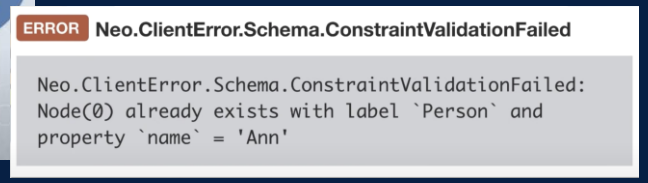
Adding CONSTRAINTS

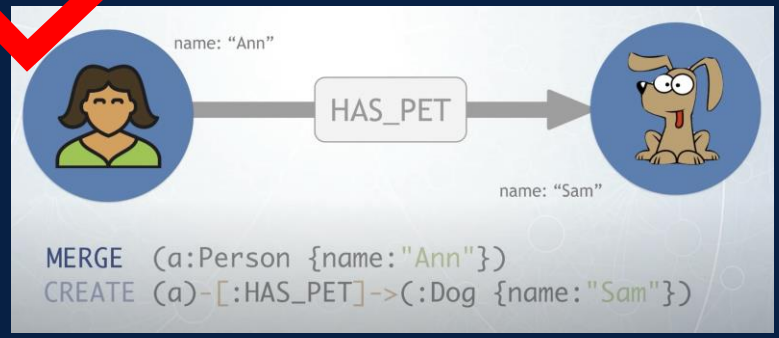
```
CREATE CONSTRAINT ON (p:Person)
ASSERT p.name IS UNIQUE
```

- Unique Constraint
- Property Existence Constraints, eg – `create constraint on ()-[m:manages]0() assert exists (m.joined)`
- Node Key Constraints, eg – `create constraint on (e:employee) assert (e.empid) is node key`

MERGE Clause







MERGE ON CREATE Clause

- Merge a node and set properties if the node needs to be created
- Example: `merge(a:Person {name:"Bill"}) on create set a.born=1996 return a`

MERGE ON MATCH Clause

- Merging nodes and setting properties on found nodes
- Example: `merge(a:Person {name:"Bill"}) on match set a.born=1996 return a`

Operators available:
AND, IN, OR (similar to SQL)

Source: https://neo4j.com/developer/cypher/intro-cypher/

Cypher Query Language

Incoming and Outgoing Relations:

To get all incoming or outgoing relations, use - -> or <- -

Example: `match(d:department{name:'COSC'})<- -(e) return d,e`

Matching multiple relationship types:

Use | operator in the relationship. Example: `match(e)-[m:manages|:worksfor]->(d) return e,m,d`

Sort:

Use ORDER BY clause with asc, desc same as SQL. Example: `match(e:employee) return e.name order by e.name`

Aggregate functions:

All SQL aggregate functions are supported by Cypher – SUM, AVG, COUNT, MIN, MAX

Examples: `match(e:emp)-[w:worksfor]->(d:dept) return d.name, count(*)`

`match(e:emp) return sum(e.salary)`

Other Clauses available:

SKIP - Skips top 'n' records

LIMIT – Returns top 'n' records similar to SQL

UNION, UNION ALL – Same as SQL, example: `match(e:emp{name:'Sam'}) return e.name union`

`match(e:emp{age:18}) return e.name`

STARTS WITH, ENDS WITH, CONTAINS

Source: https://neo4j.com/developer/cypher/intro-cypher/

SQL vs Cypher Query Language

Plsql

```
SELECT DISTINCT c.CompanyName
FROM customers AS c
JOIN orders AS o ON (c.CustomerID = o.CustomerID)
JOIN order_details AS od ON (o.OrderID = od.OrderID)
JOIN products AS p ON (od.ProductID = p.ProductID)
WHERE p.ProductName = 'Chocolade';
```

Cypher

```
MATCH (p:Product {productName:"Chocolade"})<-[:PRODUCT]-(:Order)<-[:PURCHASED]-(c:Customer)
RETURN distinct c.companyName;
```

```
SELECT e.EmployeeID, count(*) AS Count
FROM Employee AS e
JOIN Order AS o ON (o.EmployeeID = e.EmployeeID)
GROUP BY e.EmployeeID
ORDER BY Count DESC LIMIT 10;
```

```
MATCH (:Order)<-[:SOLD]-(e:Employee)
RETURN e.name, count(*) AS cnt
ORDER BY cnt DESC LIMIT 10
```

```
SELECT p.ProductName
FROM Product AS p
JOIN ProductCategory pc ON (p.CategoryID = pc.CategoryID AND pc.CategoryName = "Dairy Products")

JOIN ProductCategory pc1 ON (p.CategoryID = pc1.CategoryID
JOIN ProductCategory pc2 ON (pc2.ParentID = pc2.CategoryID AND pc2.CategoryName = "Dairy Products")

JOIN ProductCategory pc3 ON (p.CategoryID = pc3.CategoryID
JOIN ProductCategory pc4 ON (pc3.ParentID = pc4.CategoryID)
JOIN ProductCategory pc5 ON (pc4.ParentID = pc5.CategoryID AND pc5.CategoryName = "Dairy Products")
;
```

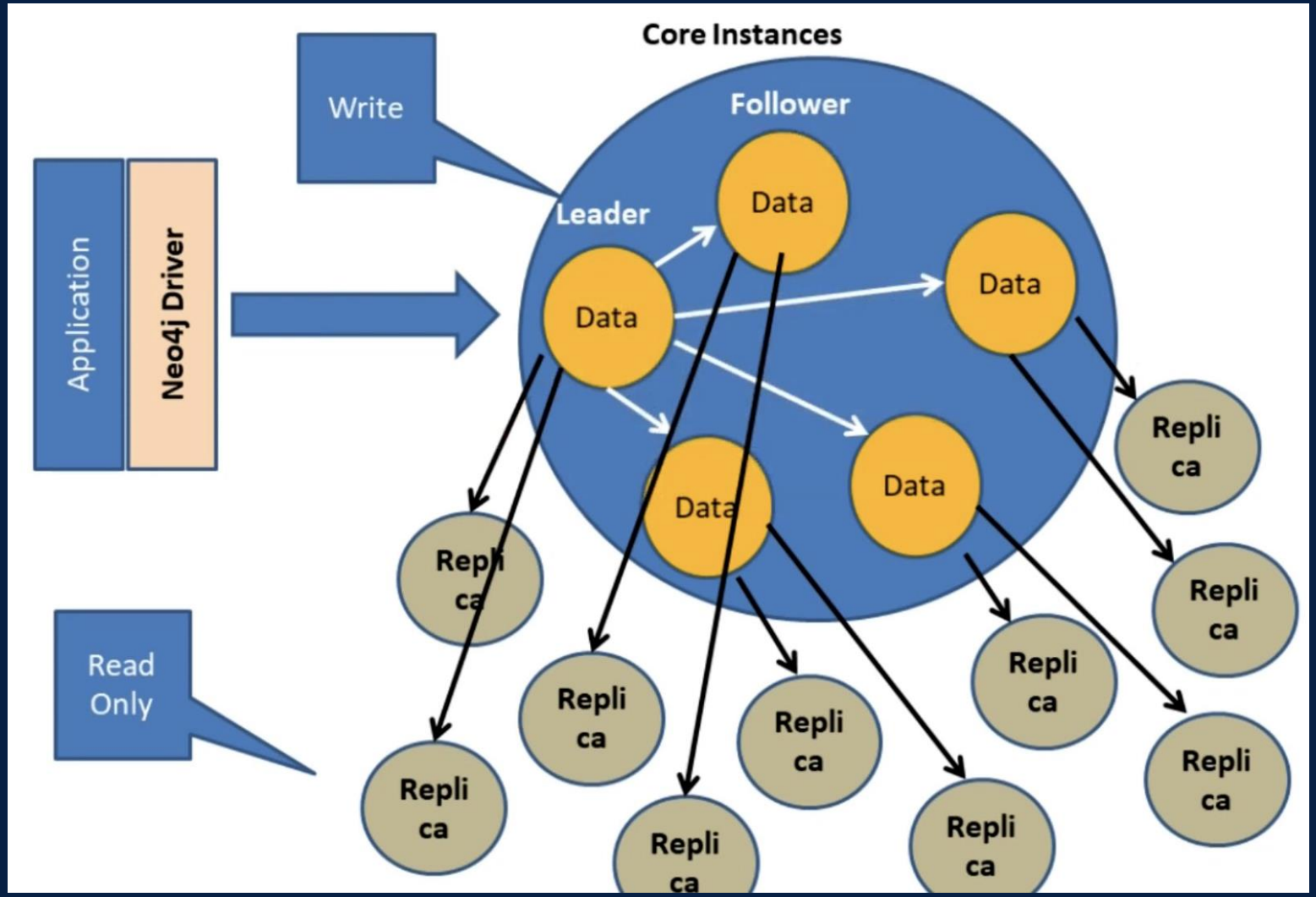
```
MATCH (p:Product)-[:CATEGORY]->(l:ProductCategory)-
[:PARENT*0..]-(:ProductCategory {name:"Dairy Products"})
RETURN p.name
```


Source: https://neo4j.com/developer/cypher/intro-cypher/

Clusters in Neo4j

Types of Servers in Neo4j Causal Cluster:

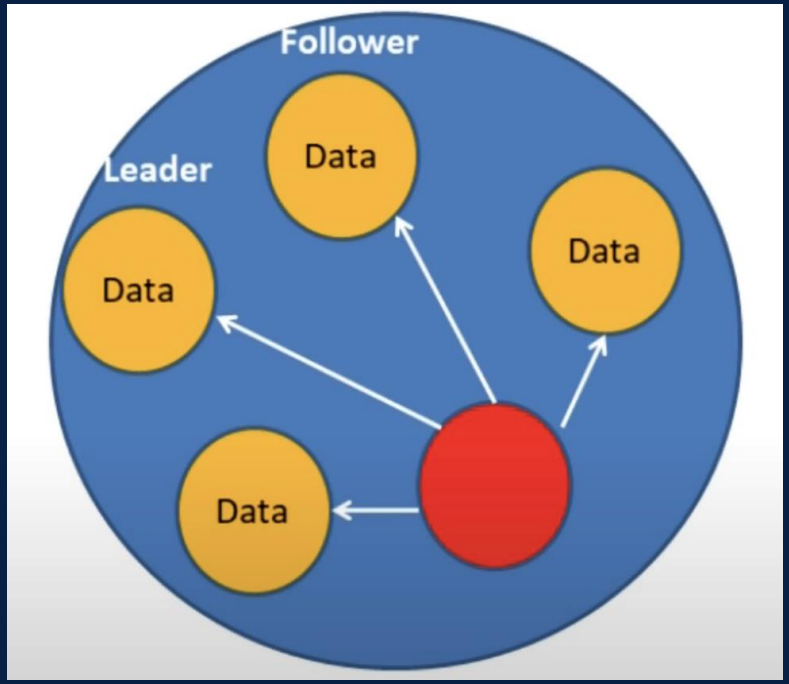
- **Core Servers**
- **Replica Servers**



Source: https://youtu.be/1YtvxGVXkeE

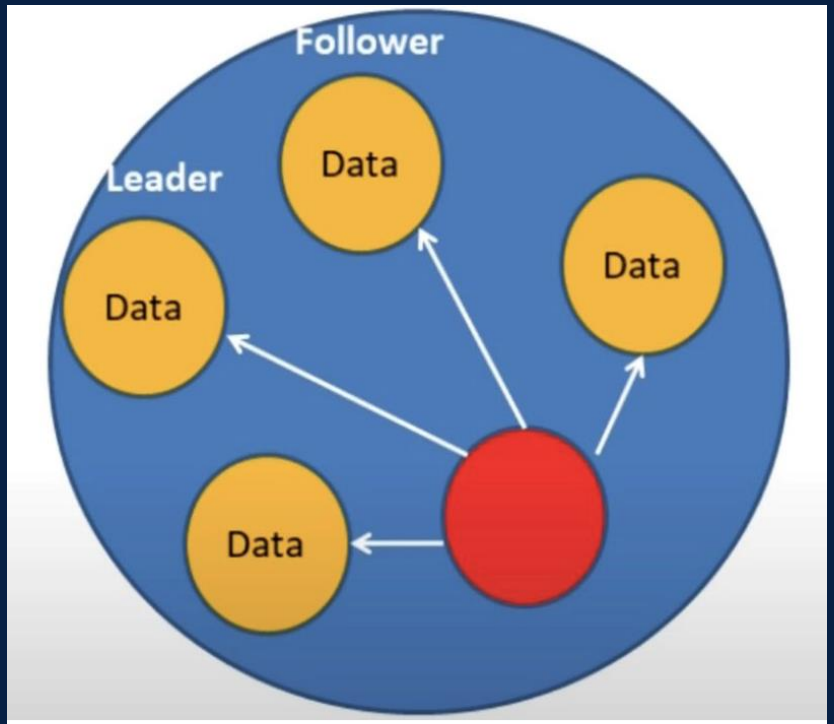
Discovery Protocol

- **The discovery protocol takes information from *causal_clustering.intital_discovery_* members in neo4j.conf, which lists the IP addresses and ports that form the cluster on start-up.**
- When consuming this information, the server will try to handshake with the other listed servers.
- On successful handshake with another server (or servers), the current server will discover the whole current topology.



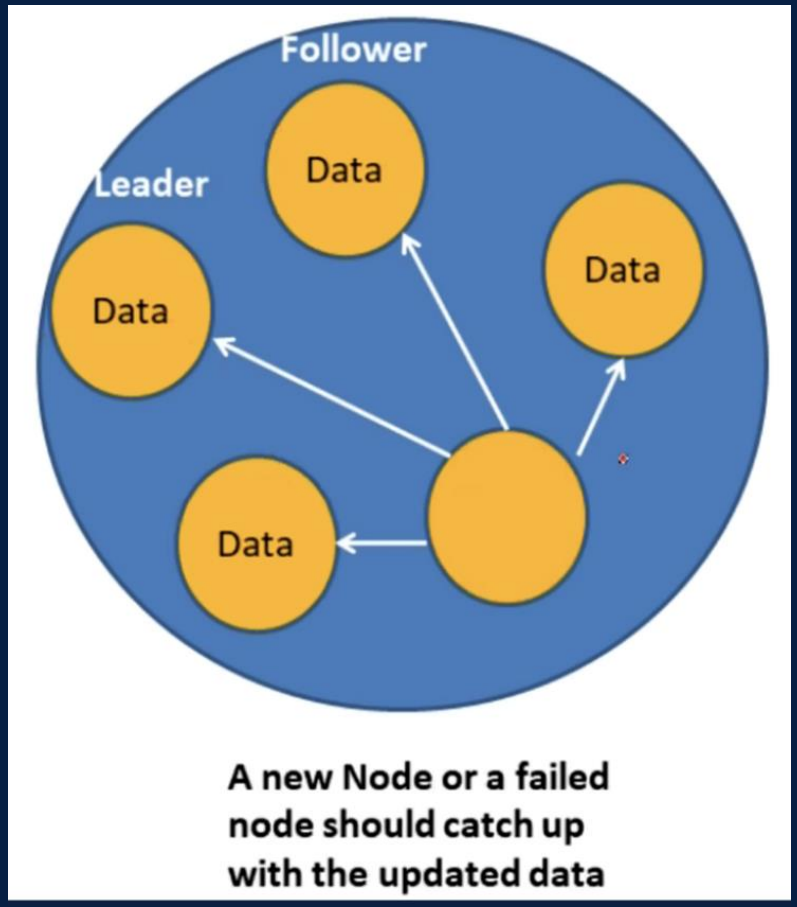
RAFT Protocol

- Once the Core Server that is performing discovery, has made a connection to one of the existing Core Servers, it then joins the RAFT protocol.
- RAFT is a distributed algorithm for maintaining a consistent log across multiple shared-nothing servers
- RAFT handles cluster membership by making it a normal part of keeping a distributed log in sync.
- Joining a cluster involves the insertion of a cluster membership entry into the RAFT log which is then reliably replicated around the existing cluster.
- Transactions happen via the RAFT protocol. When majority of the instances apply the updates, then the Leader issues commit.
- RAFT protocol also handles leader election.



Catch up Protocol

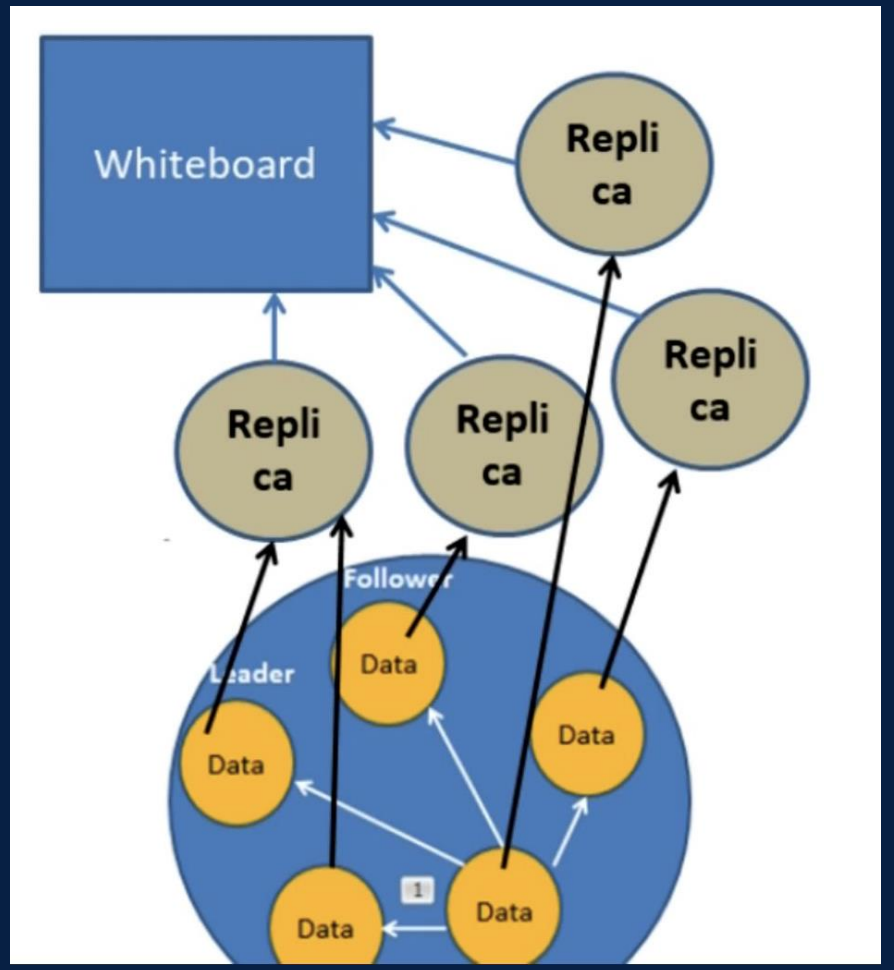
- Keeping the data between the nodes in sync
  - Polling
  - Copying the database store directly



A new Node or a failed node should catch up with the updated data

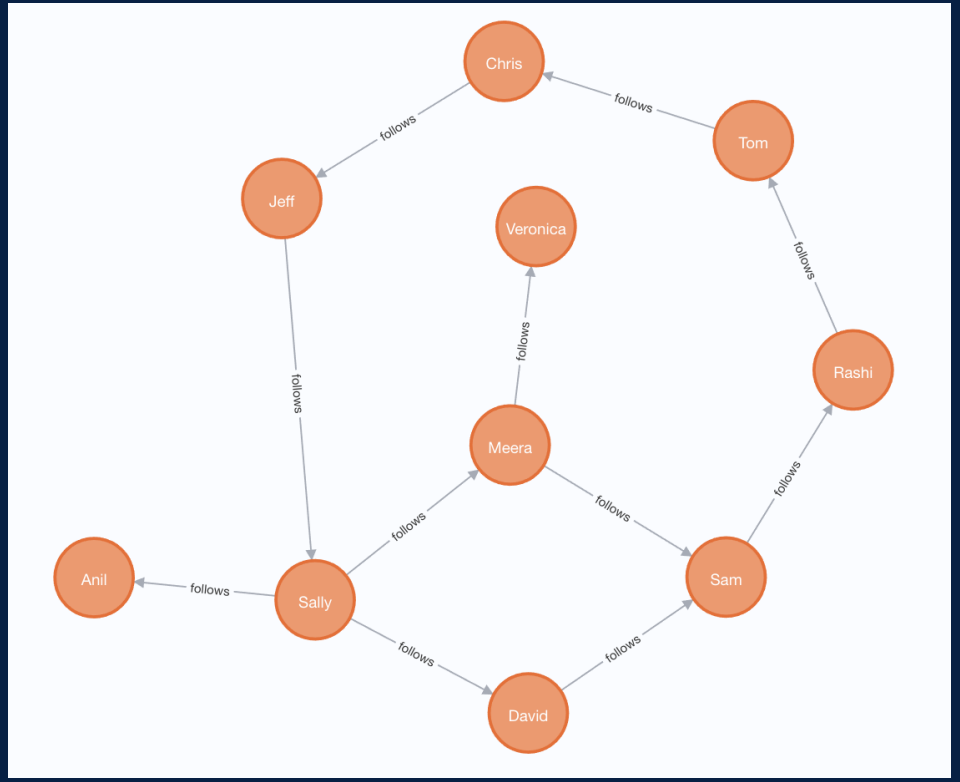
Replica Management

- When a Read Replica performs discovery, once it has made a connection to any of the available Core clusters, it proceeds to add itself into a shared whiteboard.
- This whiteboard provides a view of all live Read Replicas and is used both for routing requests from database drivers that support end-user applications and for monitoring the state of the cluster.
- The whiteboard is kept up to date as Read Replicas join and leave the cluster.

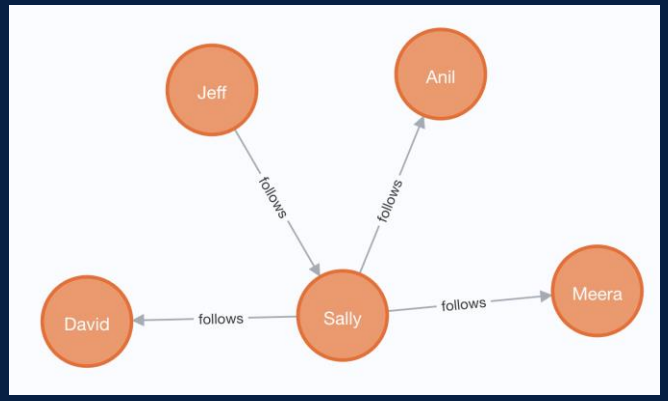


Variable Length Relationships

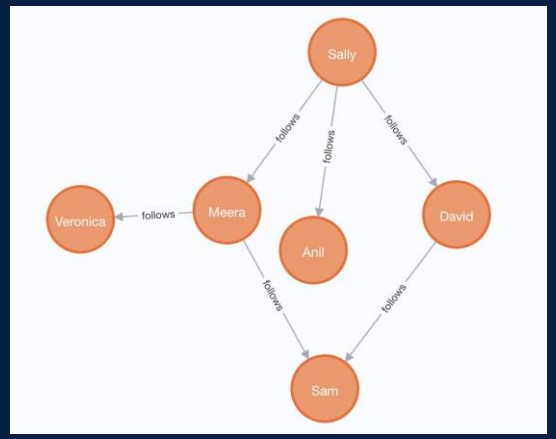
- Nodes that are a variable number of relationship->node hops away, can be found using the following syntax: `[:TYPE*minHops..maxHops]->`
- minHops and maxHops are optional and default to 1 and infinity respectively.
- When no bounds are given, dots can be omitted.
- The dots may also be omitted when setting only one bound and this implies a fixed length pattern.



Original graph model



Node 1 relationship distance away from Sally



Nodes two relationship distance away from Sally

Graph Analytics using Neo4j

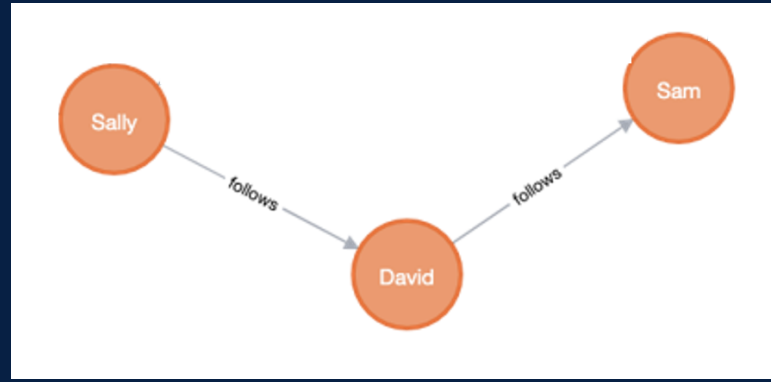
In order to use libraries built for graph analytics, add plugins APOC and GRAPH ALGORITHMS

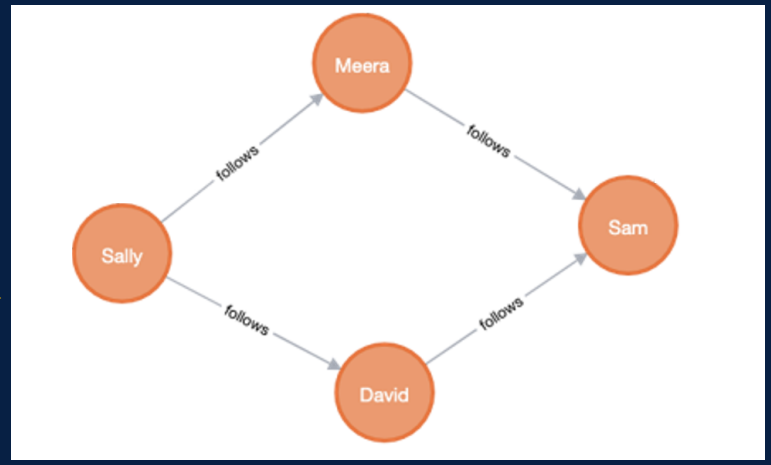
Shortest Path Algorithm:

- This can be based either on the number of hops or the weighted relationship value.

- What is the shortest path between Sally and Sam with maximum 2 paths away?
  - Hint: Use `match` with `apoc.algo.shortestpath(<condition>)` with hop counts

- List all the shortest paths between Sally and Sam:
  - Hint: Use `match` with `apoc.algo.allshortestpath (<condition>)`





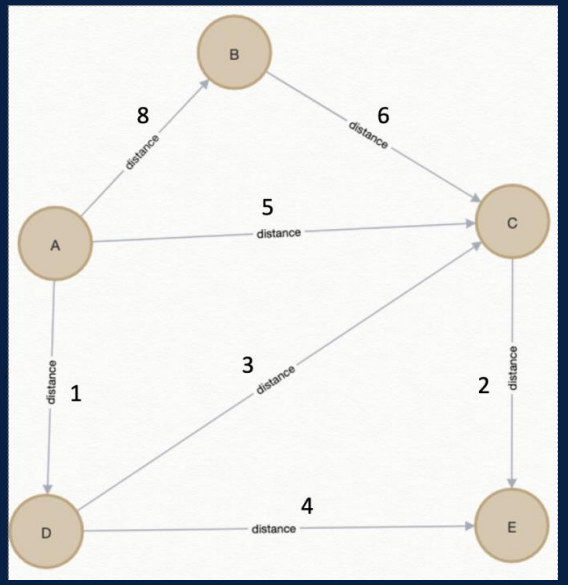
Graph Analytics using Neo4j

Weighted Graphs

- Determine the weighted shortest path between A and E (ignore directions)

```
match(l:location{name:'A'}),(l1:location{name:'E'})
call algo.shortestPath.stream(l,l1,"cost")
yield nodeId, cost
return algo.getNodeById(nodeId).name, cost
```

"algo.getNodeById(nodeId).name"	"cost"
"A"	0.0
"D"	1.0
"E"	5.0



- Display weighted shortest paths between all pairs of nodes in the graph:

```
call algo.allShortestPaths.stream("cost")
yield sourceNodeId, targetNodeId, distance
where sourceNodeId < targetNodeId
return
algo.getNodeById(sourceNodeId).name,
algo.getNodeById(targetNodeId).name, distance
order by distance
```

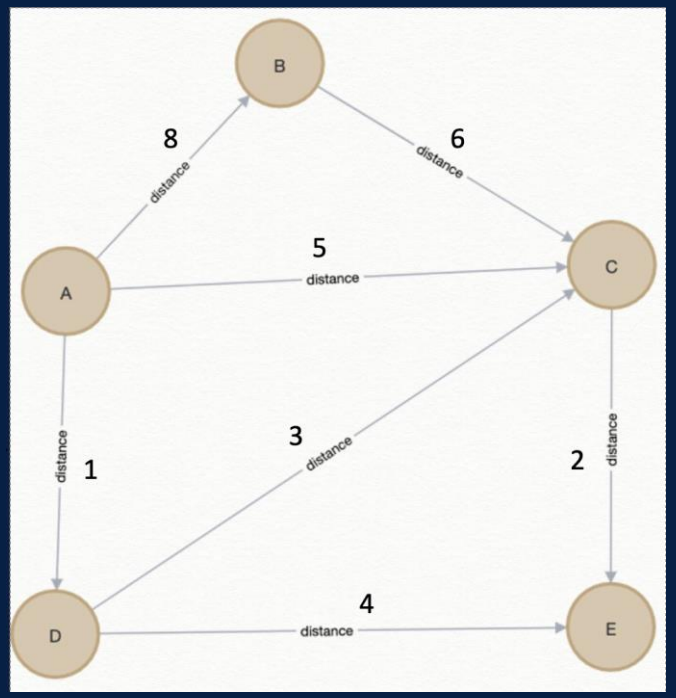
algo.getNodeById(sourceNodeId).name	algo.getNodeById(targetNodeId).name	distance
"A"	"D"	1.0
"C"	"E"	2.0
"C"	"D"	3.0
"A"	"C"	4.0
"D"	"E"	4.0
"A"	"E"	5.0
"C"	"B"	6.0
"A"	"B"	8.0
"B"	"E"	8.0

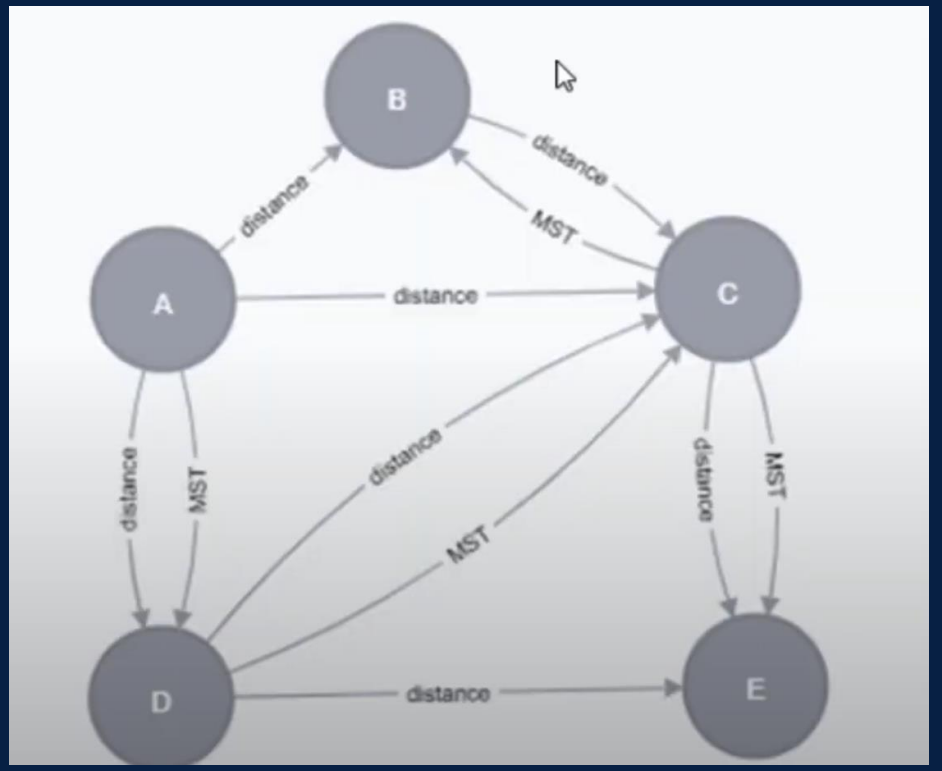
Graph Analytics using Neo4j

Minimum Spanning Tree

- To determine the best route to visit all nodes
- The tree should not contain cycles.

- Determine the MST for the given graph, starting node A.
  - Hint: Use `match` with `algo.spanningTree.minimum("location","distance","cost",id(?))`



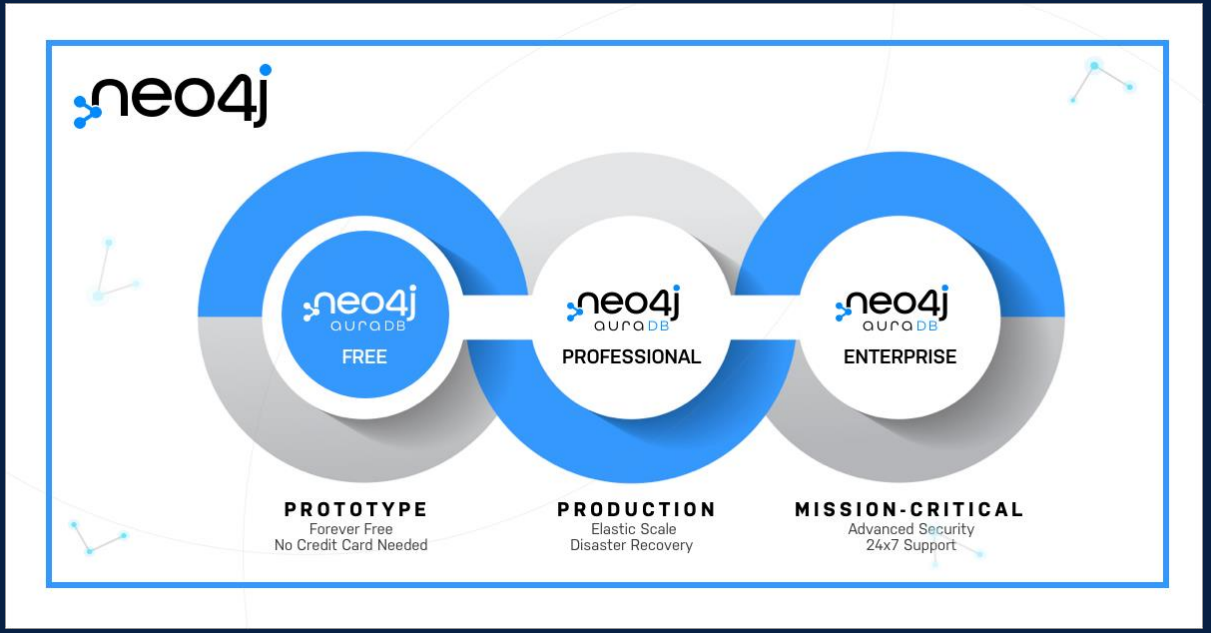


Cloud Deployments

There are different options for deploying Neo4j in the cloud. Neo4j can be deployed in:

- AWS
- Google Cloud Platform
- Microsoft Azure

Neo4j AuraDB is a fully managed Neo4j database, hosted in the cloud and requires no installation.
https://neo4j.com/cloud/platform/aura-graph-database/



Cloud Deployments

AWS

Below are the total costs for these different subscription durations.
Additional taxes or fees may apply.

Neo4j AuraDB Enterprise

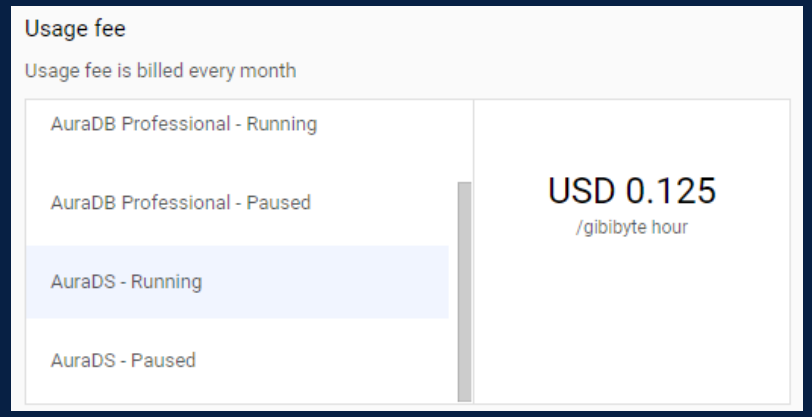
Units	Description	12 MONTHS
Aura Enterprise	Aura Enterprise 32 GB of Memory Capacity Reservation	$70,080

Additional usage fees

You will be billed monthly for additional usage costs if your usage exceeds your contract. Your additional usage costs will be determined by the number of units you use above your contract.

Description	Fees
Overage for Aura Enterprise	$1/unit

Google Cloud Platform



Single cluster monthly total: $476.50
Double cluster monthly total: $35,536.44

Microsoft Azure

Discovery Bundle Get it now	For other configurations, private offers or custom EULAs, contact marketplace-sales@neo4j.com	$66,000.00/one-time payment	1-year	$66,000.00

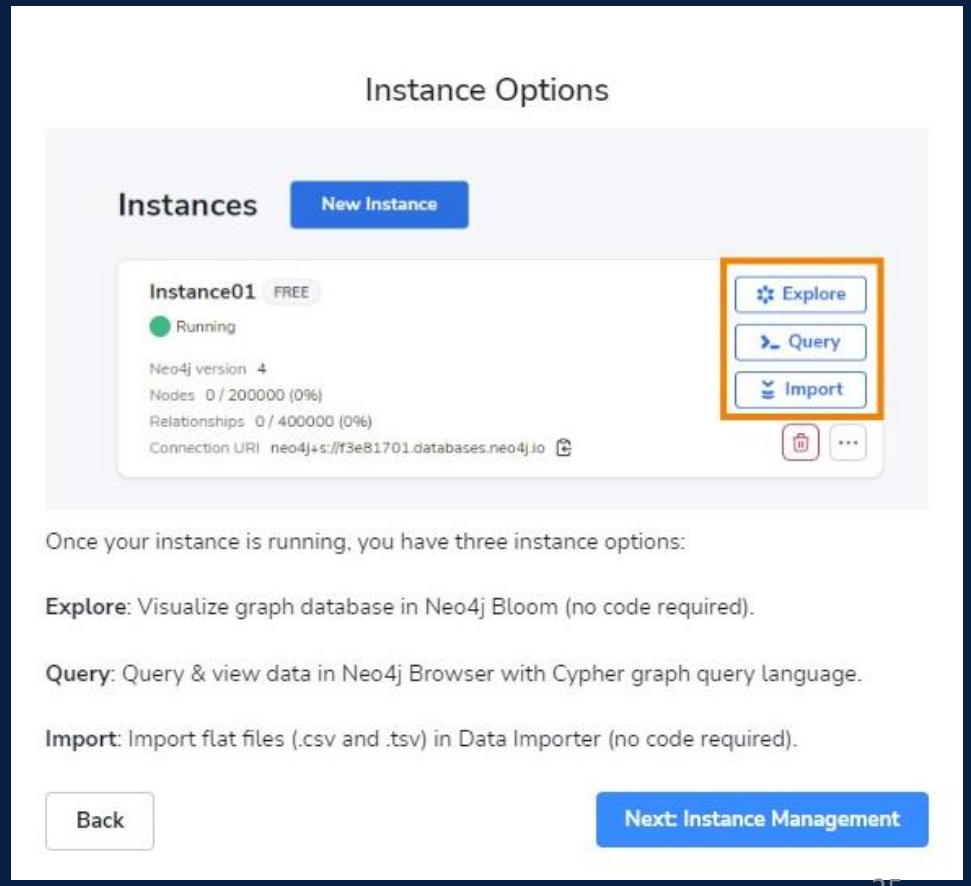
Neo4j AuraDB

Neo4j AuraDB – Fully managed cloud database for developers

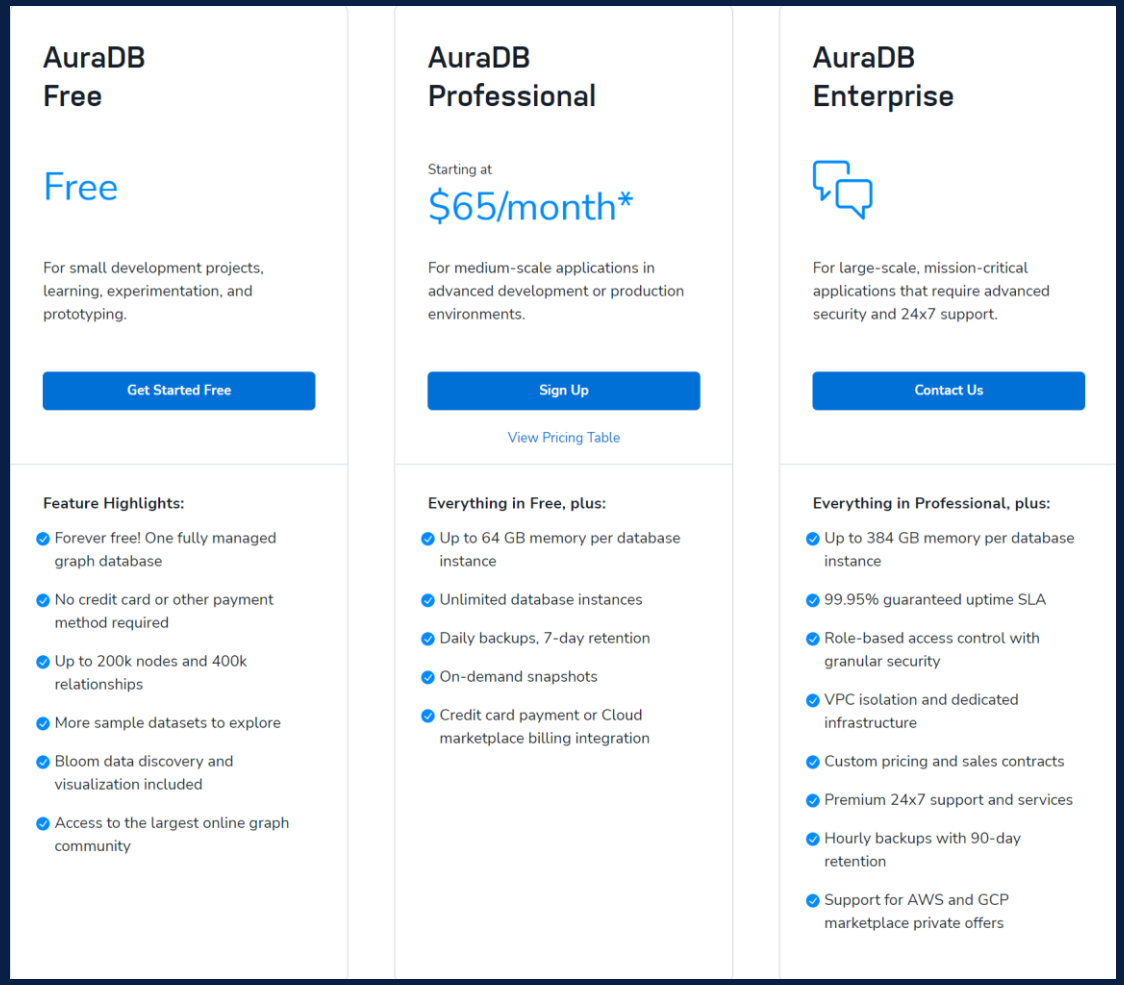
- Fully automated backups, provisioning, upgrades
- Scalable on demand (no interruption)
- Always on – secure, reliable, ACID compliant
- Consumption based pricing

Neo4j AuraDS – Graph Data science

Built for data scientists.
Fully managed graph data science engine.
Improves predictions and models at scale to bring projects into production faster. It is deigned to take proof of concept into production.



Pricing

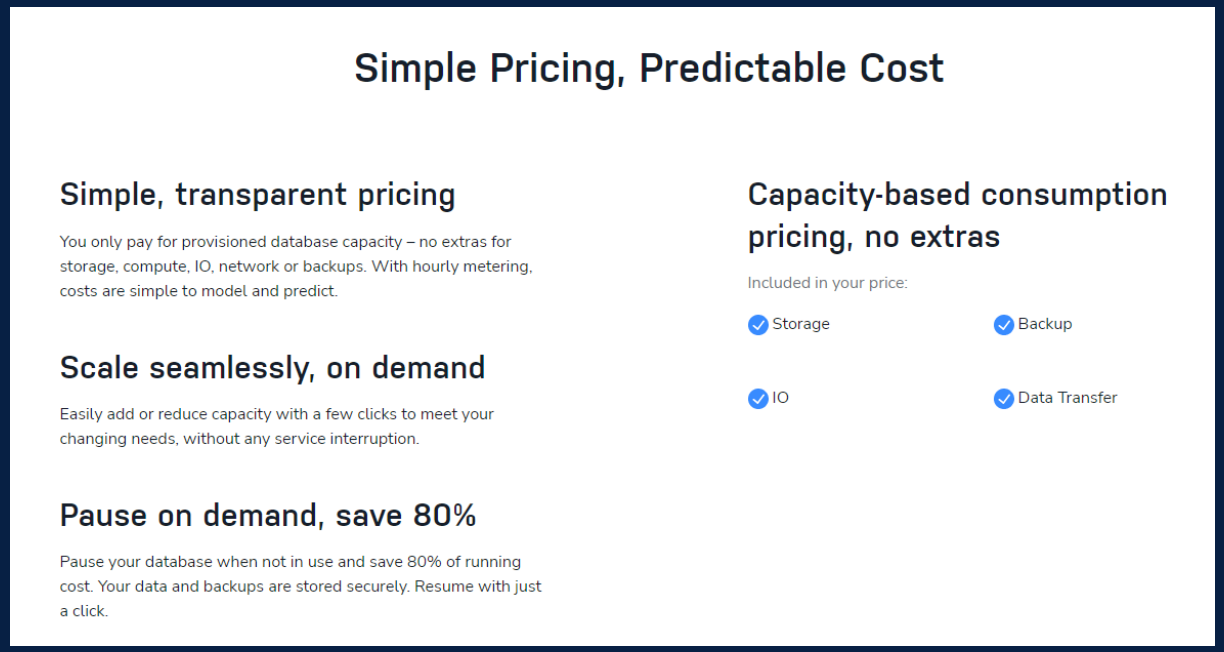


Cloud – AuraDB

Professional – pay as you go

Enterprise – prepaid contracts with custom pricing

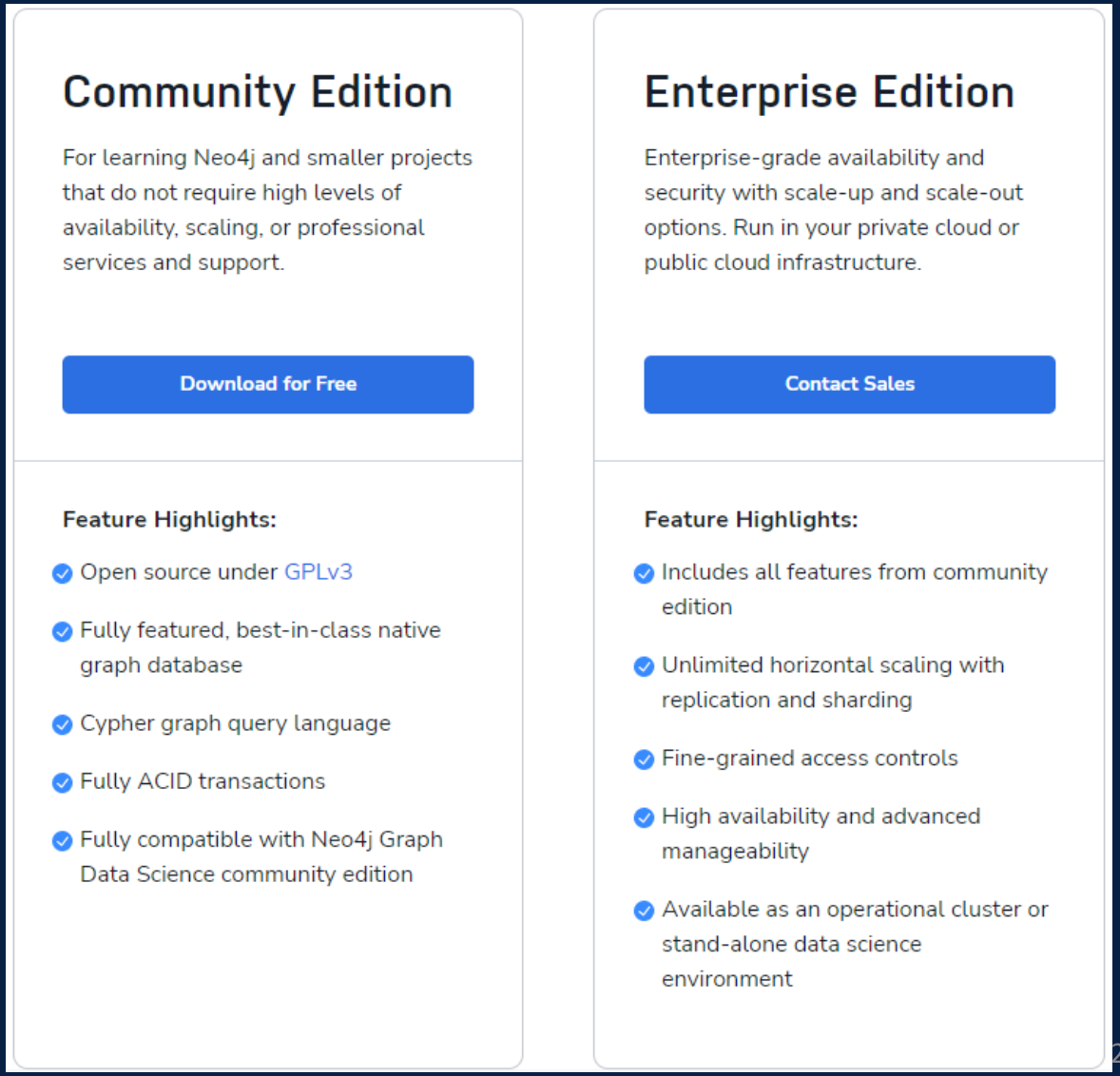
Fast, scalable, graph platform – cloud service



Pricing

Self – Managed:

- Pricing varies per usage.
- “The best way to run Neo4j on-premises, in your private cloud, or public cloud infrastructure.”



Community Edition

For learning Neo4j and smaller projects that do not require high levels of availability, scaling, or professional services and support.

Download for Free

Feature Highlights:

- Open source under GPLv3
- Fully featured, best-in-class native graph database
- Cypher graph query language
- Fully ACID transactions
- Fully compatible with Neo4j Graph Data Science community edition

Enterprise Edition

Enterprise-grade availability and security with scale-up and scale-out options. Run in your private cloud or public cloud infrastructure.

Contact Sales

Feature Highlights:

- Includes all features from community edition
- Unlimited horizontal scaling with replication and sharding
- Fine-grained access controls
- High availability and advanced manageability
- Available as an operational cluster or stand-alone data science environment

Developer Accessibility

Full suite of developer tools to attract range of users.

Application Development:
- **Spring Data Neo4j** framework for web applications.
- **GRANDstack** translate GraphQL queries to Cypher.

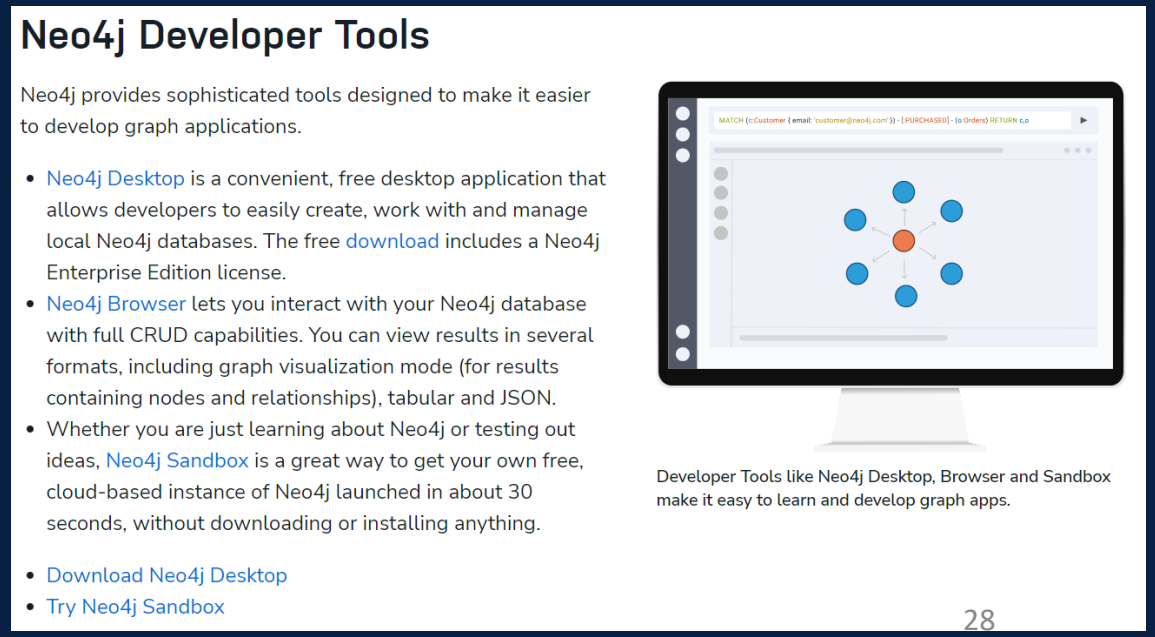
Easy migration from Neo4j to AuraDB – drag and drop local .dump file.

Graph Apps – visualization and querying with Neo4j, Browser and Bloom, relational database import tools, and monitoring tools.

- Developed by community, partners, enterprises, etc.

Neo4j Officially supported drivers:
NodeJS | Python | JavaScript | Go | .NET | Java

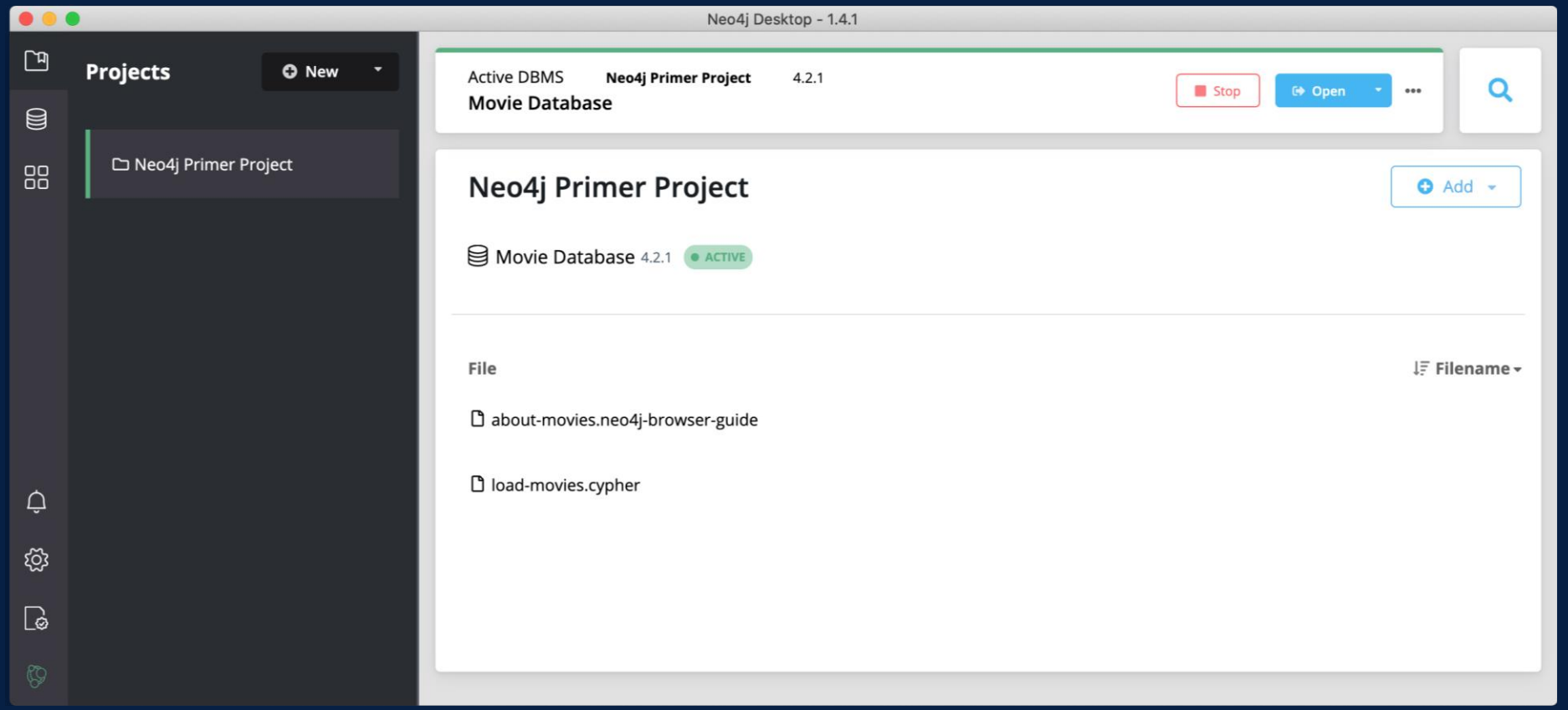
Community drivers:
PHP | Ruby | R | Erlang | Clojure | C/C++ | etc



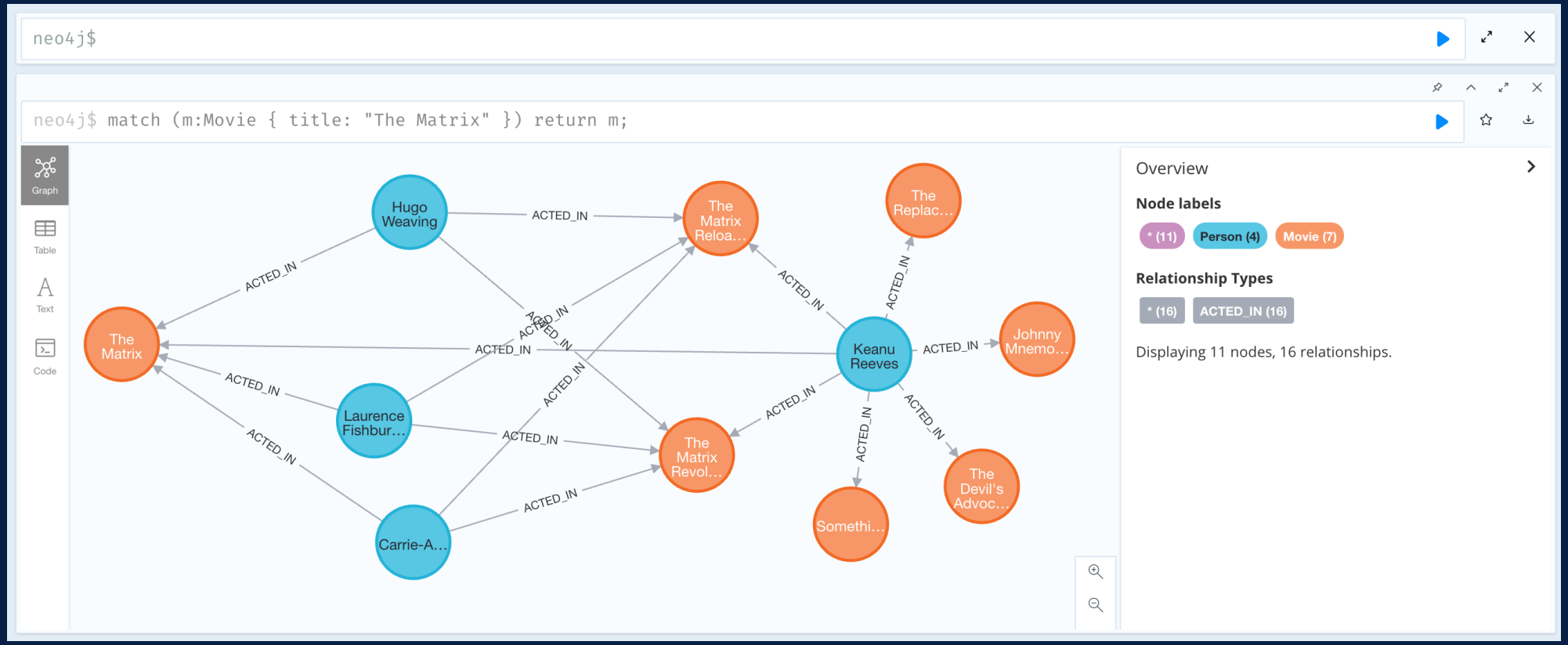
Neo4j Desktop

Neo4j Desktop

- Developer IDE for Neo4j instances.
- "Similar to Oracle Enterprise Manager."
- Connect and manage local and remote Neo4j servers.
- Comes with free development license of Neo4j Enterprise Edition.



Neo4j Browser



Neo4j Browser

- **Cypher command shell.**
- **Interact with graph and visualize data**
- **Bundled with Neo4j and available in all editions.**

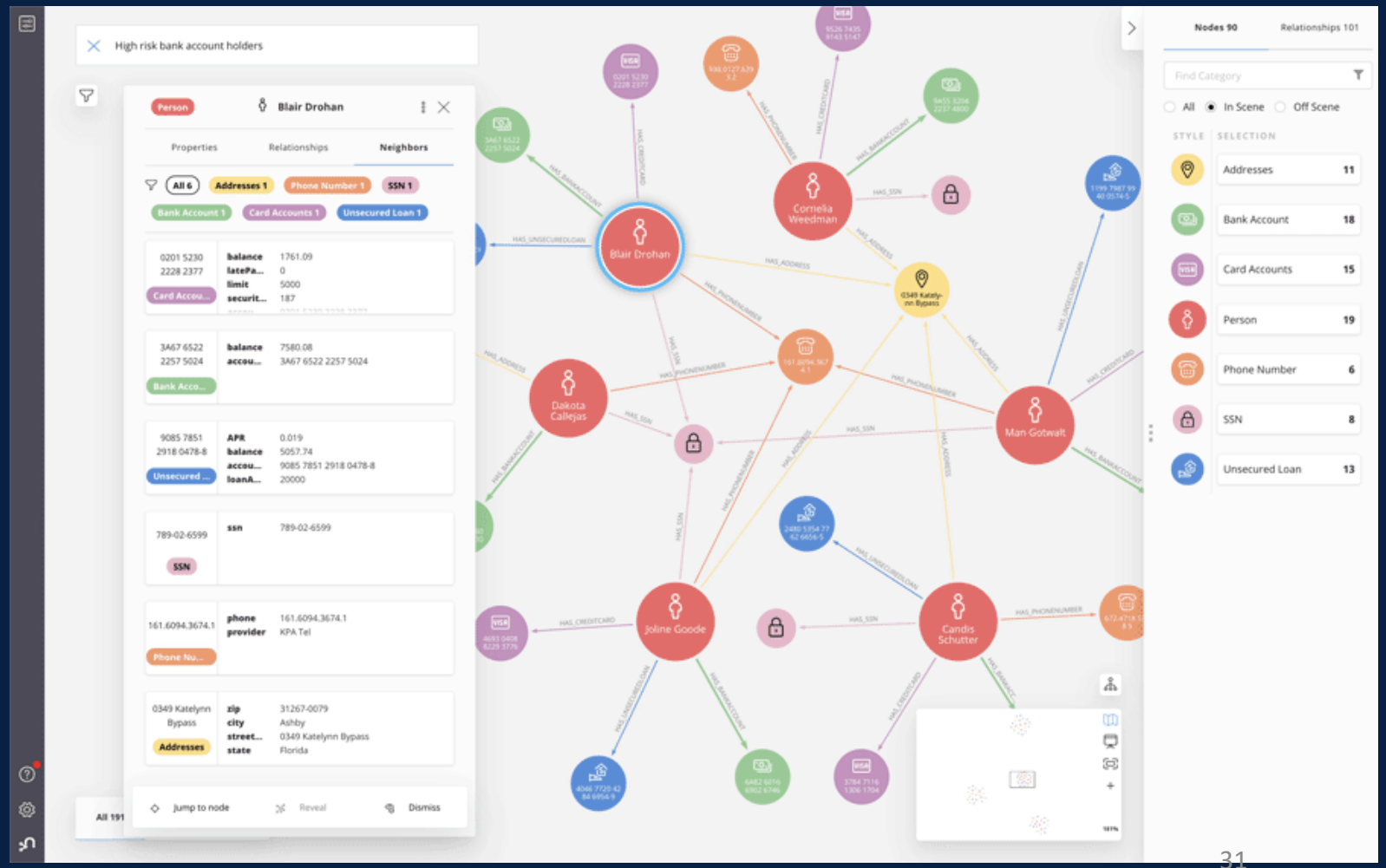
Neo4j Bloom

Neo4j Bloom – data exploration tool used to visualize data in the graph.

Allows users to navigate and query data without query language or programming. Users can write patterns to retrieve data.

Accessing Bloom:

- Neo4j Bloom local via Neo4j Desktop (local instances).
- Neo4j Bloom server via web browser.
- Neo4j Bloom via sandbox.
- Neo4j Bloom via Neo4j Database (AuraDB).

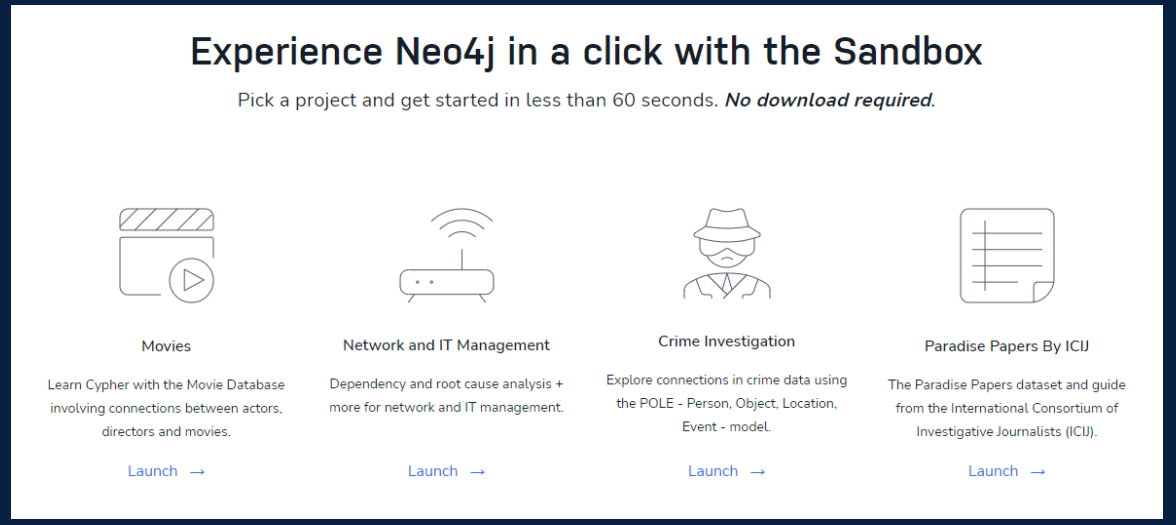


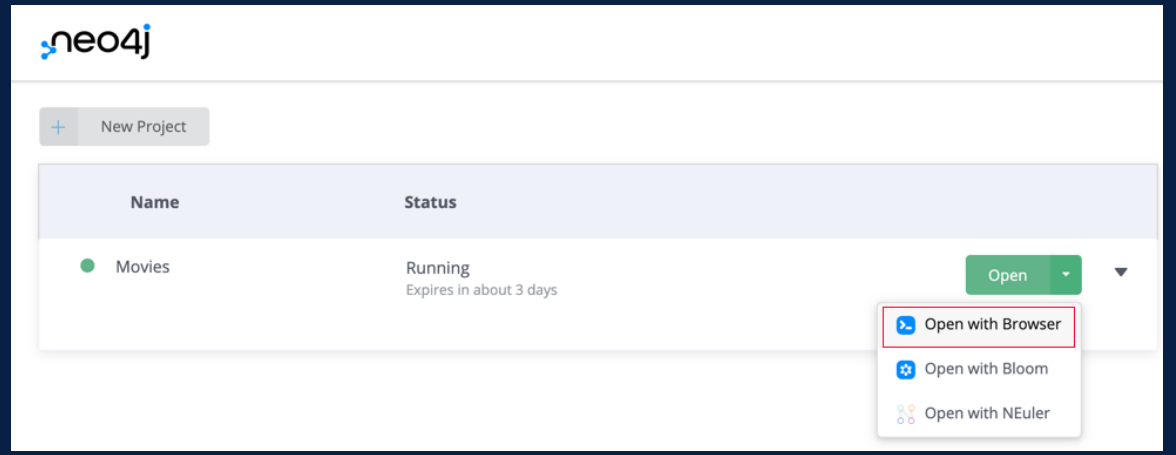
Neo4j SandBox

Neo4j Sandbox – free cloud based instance of Neo4j.

- **Free. No Download. Fast launch.**
- **Learn Neo4j, test ideas, gain experience with pre-built data examples.**

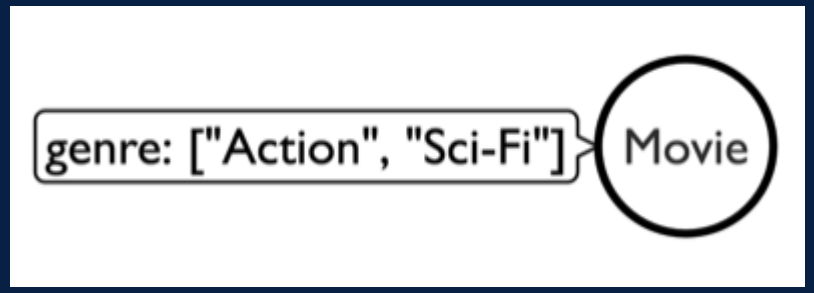
Launching a project opens Neo4j Browser as a web application and automatically connects to Neo4j instance created by sandbox without and connection URI.



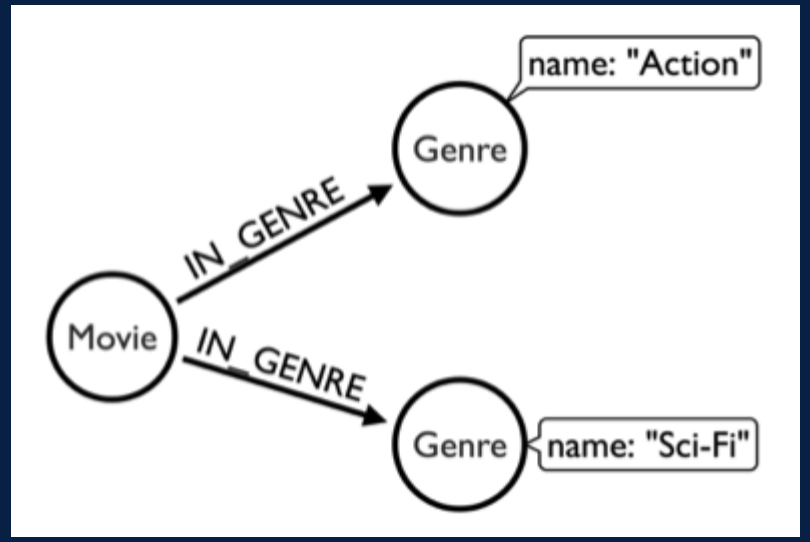


Import Relational Data

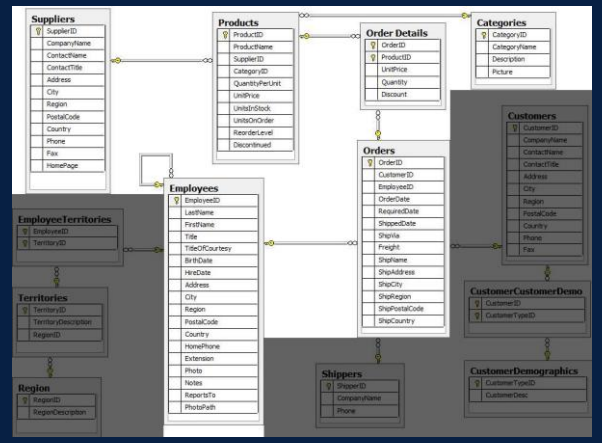
Data model design is very important. Neo4j is schema-free meaning your data model can adapt and change. Can easily import CSV files using Cypher to transform contents into graph structure.



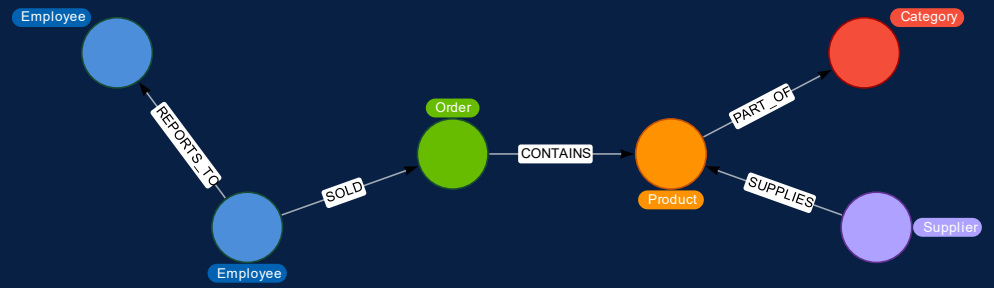
Property on a node vs relationship will determine structure.



Row – node
Table name – label
Joins (fk) – relationship
There are no nulls.



1. **Create nodes**
2. **Create indexes and constraints on nodes**
3. **Create relationships between nodes**
4. **Querying the graph is ready**



Conference

Free educational Neo4j conference.

Keynote speaker.

12 Featured speakers from withing industry/company.

Beginner, Intermediate and advanced agendas.

References

- https://neo4j.com/docs/
- https://neo4j.com/developer/data-import/
- https://www.youtube.com/watch?v=8jNPelugC2s
- https://youtu.be/REVkXVxvMQE


THE UNIVERSITY OF BRITISH COLUMBIA