Neo4j: Graph Database

Abdu Alawini

University of Illinois at Urbana-Champaign

CS411: Database Systems

Aggregation

common aggregation functions are supported:

count, sum, avg, min, and max

```
MATCH (p:Person)
RETURN count(*) as headcount;
```

"headcount"
"145"

```
MATCH (actor:Person)-[:ACTED_IN]->(movie:Movie)<-[:DIRECTED]-(director:Person)
RETURN actor,director,count(*) AS collaborations
```

"actor"	"director"	"collaborations"
{"born":"1946","name":"Susan Sarandon"}	{"born":"1965","name":"Lana Wachowski"}	"1"
{"born":"1960","name":"Annabella Sciorra"}	{"born":"1956","name":"Vincent Ward"}	"1"
{"born":"1956","name":"Tom Hanks"}	{"born":"1951","name":"Robert Zemeckis"}	"2"
{"born":"1953","name":"David Morse"}	{"born":"1959","name":"Frank Darabont"}	"1"

COLLECT

- ***Collect()*** function collects all aggregated values into a list

```
MATCH (m:Movie)<-[:ACTED_IN]-(a:Person)
RETURN m.title AS movie, collect(a.name) AS cast, count(*) AS actors
```

"movie"	"cast"	"actors"
"You've Got Mail"	["Dave Chappelle","Parker Posey","Steve Zahn","Meg Ryan","Tom Hanks","Greg Kinnear"]	"6"
"Apollo 13"	["Tom Hanks","Kevin Bacon","Ed Harris","Bill Paxton","Gary Sinise"]	"5"
"Johnny Mnemonic"	["Dina Meyer","Takeshi Kitano","Ice-T","Keanu Reeves"]	"4"
"Stand By Me"	["Marshall Bell","Kiefer Sutherland","John Cusack","Corey Feldman","Jerry O'Connell","River Phoenix","Wil Wheaton"]	"7"
"The Polar Express"	["Tom Hanks"]	"1"

Composing Statements: UNION

- UNION combines the results of two statements that have the same result structure

```
MATCH (p:Person)-[r:ACTED_IN]->(m:Movie)
RETURN p.name as name, type(r) as Acted_Directed, m.title as title
UNION
MATCH (p:Person)-[r:DIRECTED]->(m:Movie)
RETURN p.name as name, type(r) as Acted_Directed, m.title as title
```

Equivalent Query

```
MATCH (actor:Person)-[r:ACTED_IN|DIRECTED]->(movie:Movie)
RETURN actor.name AS name, type(r) AS acted_in, movie.title AS title
```

"name"	"Acted_Directed"	"title"
"Nathan Lane"	"ACTED_IN"	"Joe Versus the Volcano"
"Tom Hanks"	"ACTED_IN"	"Joe Versus the Volcano"
"Meg Ryan"	"ACTED_IN"	"Joe Versus the Volcano"
"Lilly Wachowski"	"DIRECTED"	"The Matrix"
"Lana Wachowski"	"DIRECTED"	"The Matrix"
"Rob Reiner"	"DIRECTED"	"When Harry Met Sally"

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

Composing Statements: WITH

- WITH clause combines individual parts of a query and declare which data flows from one to the other.

- WITH is like RETURN with the difference that it doesn't finish a query but prepares the input for the next part.

WITH Example

```
MATCH (person:Person)-[:ACTED_IN]->(m:Movie)
WITH person, count(*) AS appearances, collect(m.title) AS movies
WHERE appearances > 1
RETURN person.name, appearances, movies
```

"person.name"	"appearances"	"movies"
"Cuba Gooding Jr."	"4"	["A Few Good Men","Jerry Maguire","As Good as It Gets","What Dreams May Come"]
"Oliver Platt"	"2"	["Frost/Nixon","Bicentennial Man"]
"Philip Seymour Hoffman"	"2"	["Twister","Charlie Wilson's War"]
"Sam Rockwell"	"2"	["The Green Mile","Frost/Nixon"]
"Greg Kinnear"	"2"	["As Good as It Gets","You've Got Mail"]
"Zach Grenier"	"2"	["RescueDawn","Twister"]
"Rosie O'Donnell"	"2"	["A League of Their Own","Sleepless in Seattle"]

Source: https://neo4j.com/developer/cypher-query-language/

Indexing and Constraints

- Goal of indexing: find the starting point in the graph as fast as possible

```
CREATE INDEX ON :ACTOR(name);
```

```
MATCH (p:ACTOR {name: 'Michael'}) RETURN p
```

Interested in DB Tuning?
http://neo4j.com/docs/developer-manual/current/cypher/query-tuning/using/

- Unique constraints guarantee uniqueness of a certain property on nodes with a specific label.

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

Index Management

- Listing database indexes

```
CALL db.indexes
```

"description"	"state"	"type"
"INDEX ON :Person(name)"	"ONLINE"	"node_label_property"

- Dropping an Index

```
DROP INDEX ON :Person(name)
```

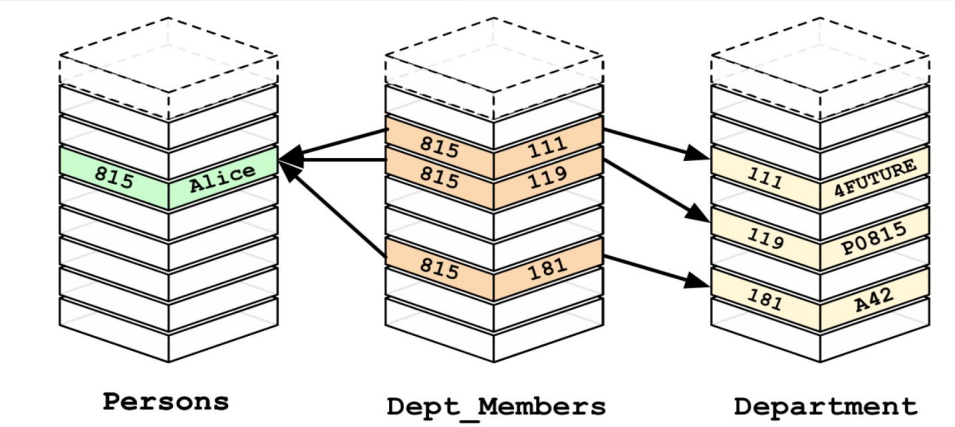
From Relational to Graph Databases

- Graph databases store relationships and connections as first-class entities: "Property Graph Model"

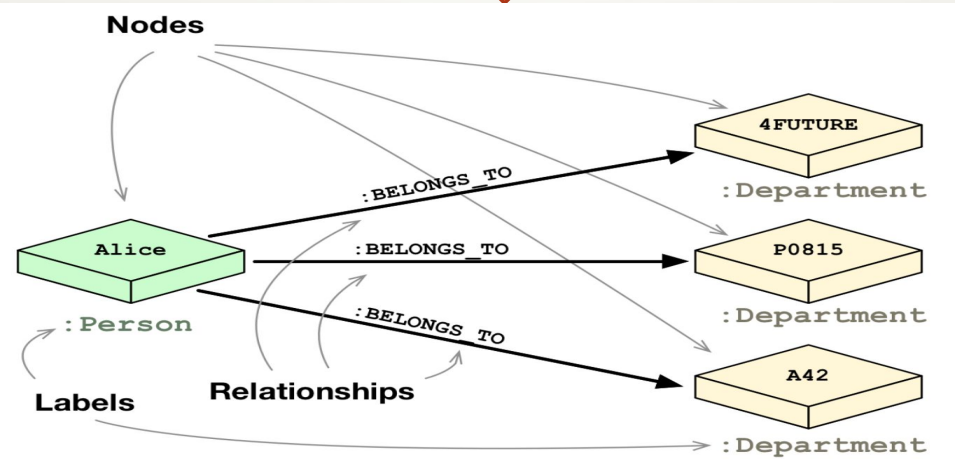
RDBMS	Graph Databases
Tables	Set of Nodes/Relationships
Rows	Nodes
Columns and data	Data properties and values
Constraints	Relationships
Joins	Traversals

From Relational to Graph Databases

Relational Model

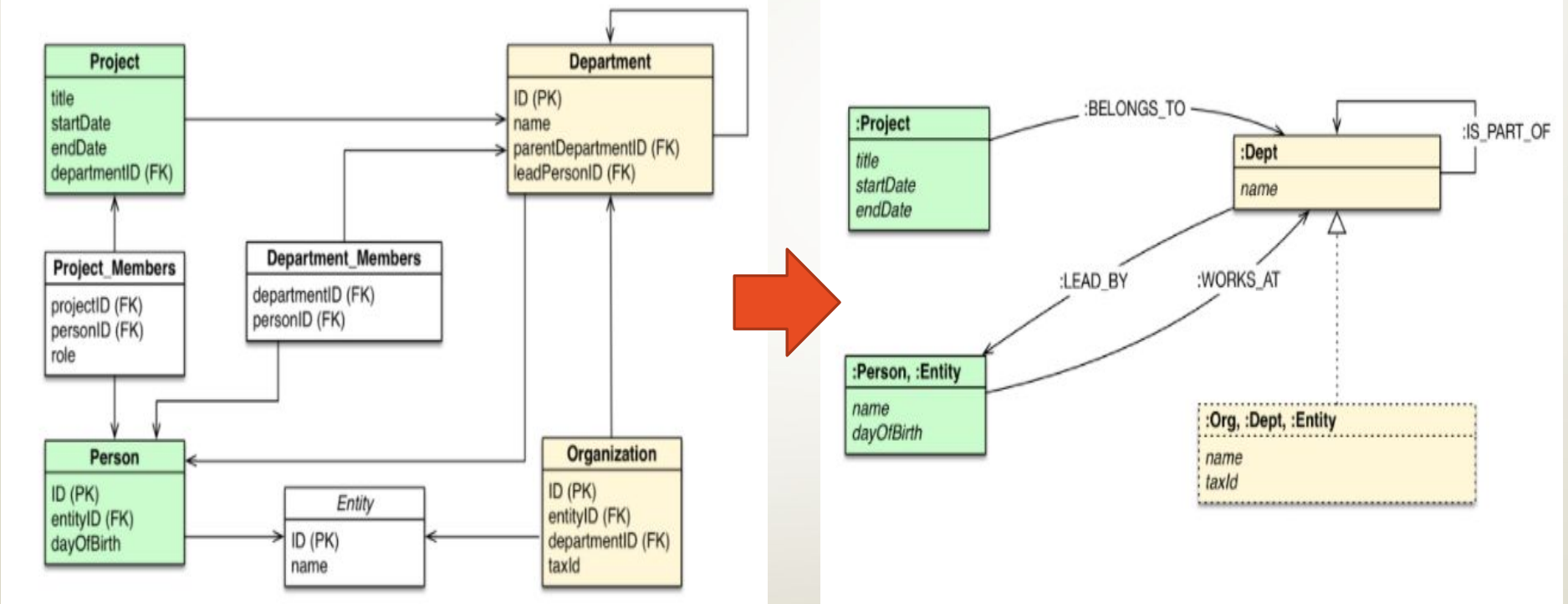


Graph Model



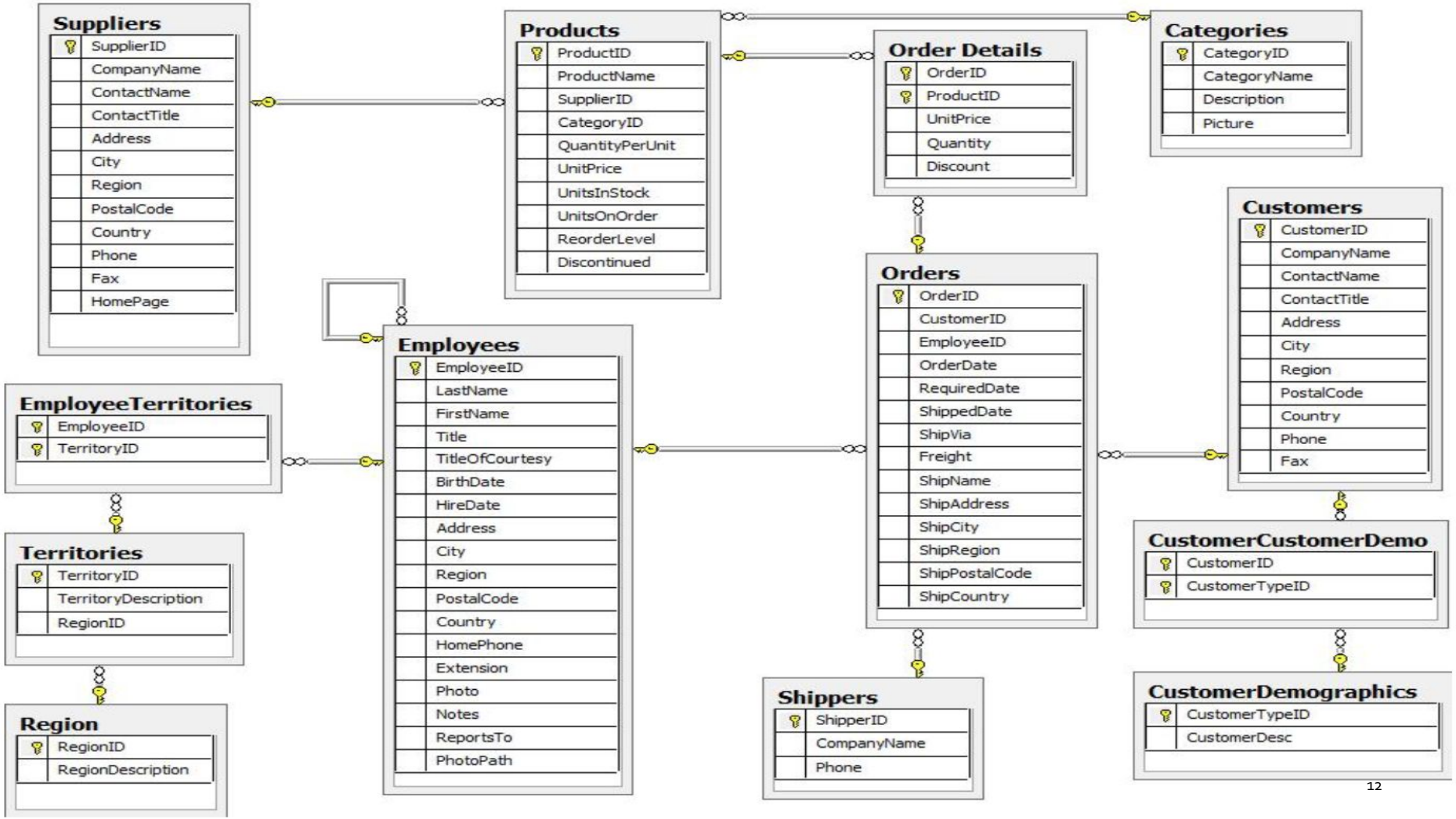
Source: https://neo4j.com/developer/graph-db-vs-rdbms/#_from_relational_to_graph_databases

DB=>Graph Data Model Transformation

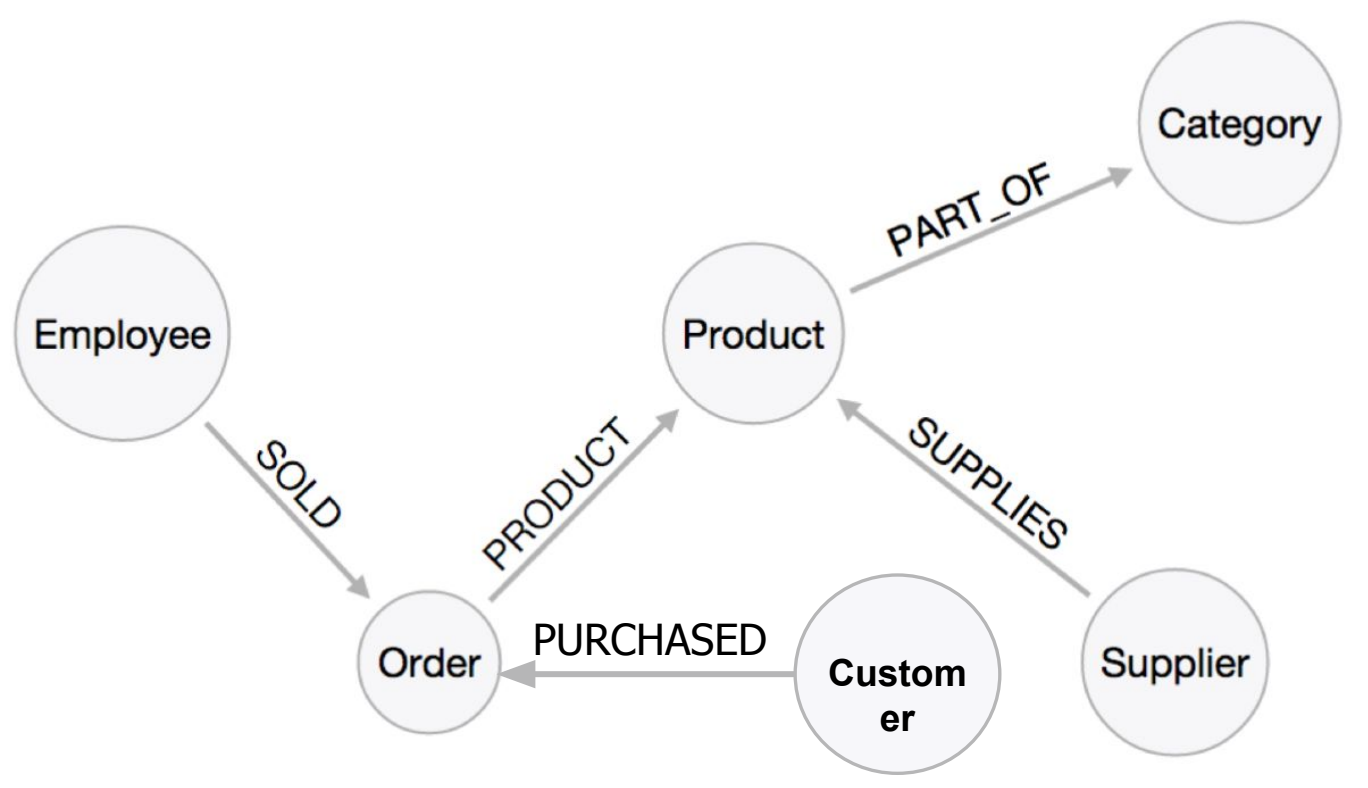


Source: https://neo4j.com/developer/graph-db-vs-rdbms/#_from_relational_to_graph_databases

Northwind Example



Northwind: Graph Model



Source: https://neo4j.com/developer/cypher-query-language/

Querying Northwind DB: SQL vs. Neo4J

- Select everything from the products table

SQL

```
SELECT p.*
FROM products as p;
```

Cypher

```
MATCH (p:Product)
RETURN p;
```


Source: https://neo4j.com/developer/cypher-query-language/

SQL vs. Neo4J: Projection

- Select Product Name and Price from products table

SQL

```
SELECT p.ProductName,
p.UnitPrice
FROM products as p
ORDER BY p.UnitPrice DESC
LIMIT 10;
```

Cypher

```
MATCH (p:Product)
RETURN p.productName,
p.unitPrice ORDER BY p.unitPrice
DESC
LIMIT 10;
```

Source: https://neo4j.com/developer/cypher-query-language/

SQL vs. Neo4J: Filtering

- Select Product Name and Price for “Chocolate”

SQL

```
SELECT p.ProductName, p.UnitPrice
FROM products AS p
WHERE p.ProductName = 'Chocolate';
```

Cypher

```
MATCH (p:Product)
WHERE p.productName = "Chocolate"
RETURN p.productName, p.unitPrice;
```

OR

```
MATCH (p:Product
{productName:"Chocolate"})
RETURN p.productName, p.unitPrice;
```

Source: https://neo4j.com/developer/cypher-query-language/

SQL vs. Neo4J: Filtering

- List expensive products that starts with C.

SQL

```
SELECT p.ProductName,
p.UnitPrice FROM products AS p
WHERE p.ProductName LIKE 'C%'
AND p.UnitPrice > 100;
```

Cypher

```
MATCH (p:Product)
WHERE p.productName STARTS WITH
"C" AND p.unitPrice > 100
RETURN p.productName, p.unitPrice;
```


Source: https://neo4j.com/developer/cypher-query-language/

SQL vs. Neo4J: Joining vs. Traversing

- Who bought Chocolate?

SQL

```
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 = 'Chocolate';
```

Cypher

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


Source: https://neo4j.com/developer/cypher-query-language/

SQL vs. Neo4J: Aggregation

- Find top-selling employees

SQL

```
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;
```

Cypher

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


Source: https://neo4j.com/developer/cypher-query-language/

Summary

- Graph databases store relationships and connections as first-class entities
- Graph databases have good performance when dealing with connected data
- Cypher is a declarative pattern-matching graph query language
- Querying connected data is easier with Cypher

Neo4j Resources

1. Neo4j Tutorial:
   https://www.tutorialspoint.com/neo4j/index.htm
2. Video Tutorials:
   https://neo4j.com/blog/neo4j-video-tutorials/?_ga=2.57983406.580712586.1555337212-902296776.1553382068
3. GraphGists are teaching tools which allow you to explore how data in a particular domain would be modeled as a graph and see some example queries of that graph data

   - https://neo4j.com/graphgists/

4. Awesome user-defined procedures:
   https://github.com/neo4j-contrib/neo4j-apoc-procedures