WOMEN WHO CODE TORONTO

Building a Personalized Recommender Using Neo4j Graph Database

Grace Tenorio @datatheque

Wifi & Username: TBD Password: TBD

Thank you

@WomenWhoCodeTO

Thank you

@WomenWhoCodeTO

intelliware.ca

software development

@WomenWhoCodeTO

PriceMetrix®

REALIZE YOUR VALUE

@WomenWhoCodeTO

@WomenWhoCodeTO

HAVE YOU JOINED SLACK?

Get updates first! Special giveaways on Slack

bit.ly/WWCTOSlack

@WomenWhoCodeTO

NEW ON SLACK

Let’s celebrate wins!

#applaud-her

COMMUNITY UPDATES

JUN. 9

RxWorkshop

Learn from Ben Lesh, dev lead of RxJS and Senior Software Engineer at Google. Get started in functional reactive programming. Enter & mention us for a chance to win:

bit.ly/RxWorkshop-WWCTO

www.rxworkshop.com

@WomenWhoCodeTO

WWC Toronto Team

Stephanie

Betty

Stella

Rebecca

Maddie

🎉 Aileen

@WomenWhoCodeTO

WWC Toronto Team

🎉 Nahrin

@WomenWhoCodeTO

OUR SPEAKER

Grace Tenorio

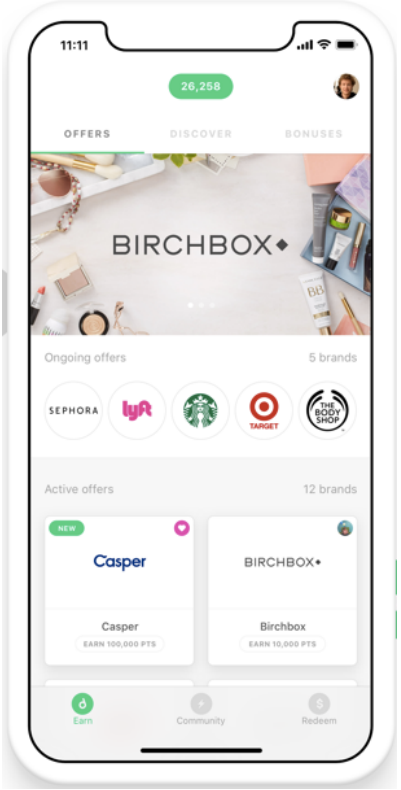
@datatheque

@WomenWhoCodeTO

Building a Personalized Recommender with Neo4j

Who Am I?

- Data Scientist at drop
- Data Science Blogger
- Budding Graphista



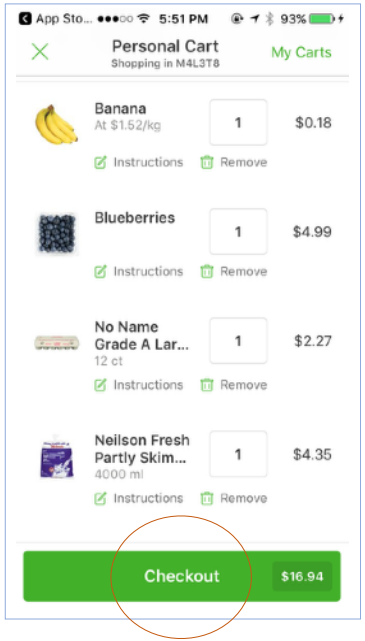
Agenda

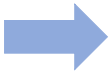
- Use Case Context
- Recommender Systems
- Neo4j Graph Database
- Live Demo

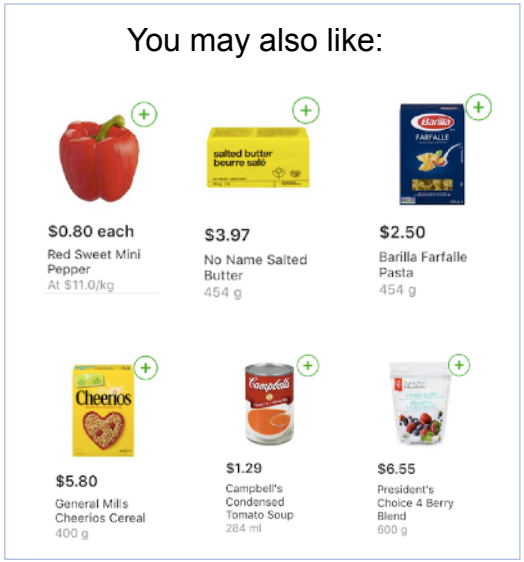
Use Case Context

What We Are Building

Recommender system for an online grocery





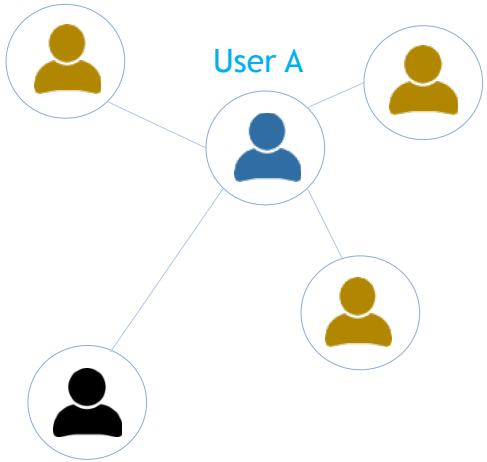


instacart Grocery Data

Instacart has open-sourced a subset of their transaction data

https://www.instacart.com/datasets/grocery-shopping-2017

Our Recommendation Strategy



Find users similar to A based on purchase history

Identify products they have purchased that A has not yet purchased

Select top 10 products to recommend to A

Tools for Implementing Our Recommender

- open-source programming language
- general purpose

- open-source web application
- combines code, output, text & visualization

- open-source graph database
- Cypher query language

Recommender Systems

What Are Recommender Systems?

“Tools to help identify worthwhile stuff.”
- Joseph Konstan, Intro to Recommender Systems

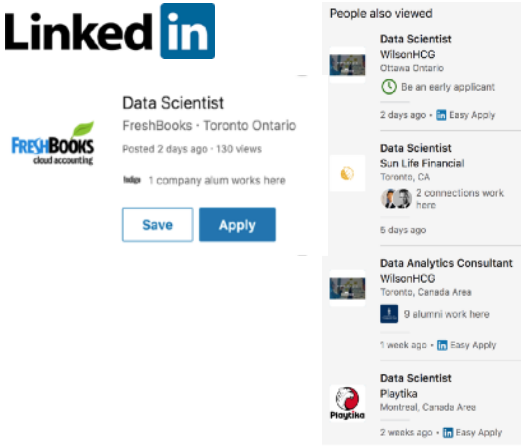
Dr. Joseph Konstan
Professor, Computer Science and Engineering
University of Minnesota

Recommender Systems Are Everywhere!

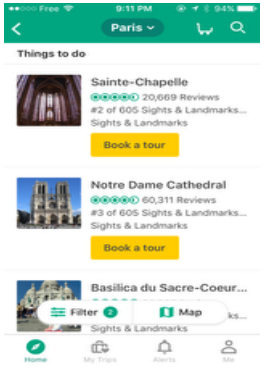
amazon

Frequently bought together

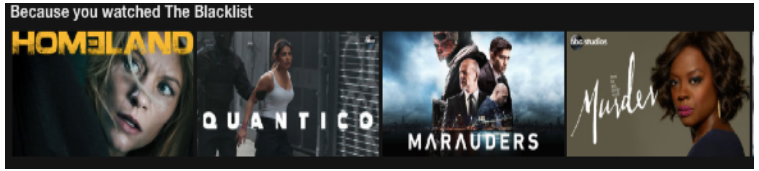
LinkedIn



tripadvisor



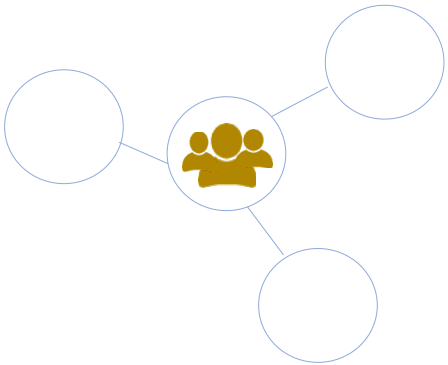
NETFLIX



Types of Recommenders

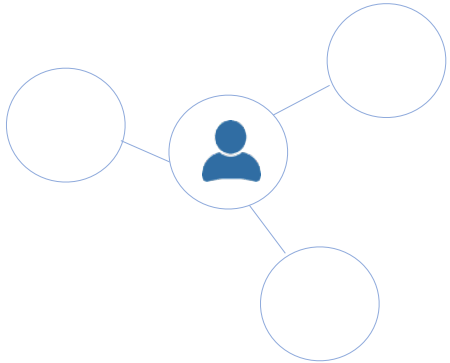
Non-Personalized

Recommendations **not** tailored to a specific user



Personalized

Recommendations tailored to a specific user



Non-Personalized Recommender Algorithms

Find items based on popularity

- averages
- up/down votes
- likes

Aggregated Measures

Find items likely to occur together

- apriori algorithm to get frequent item sets
- support, confidence & lift to quantify relationship strength

Association Rules Mining

Personalized Recommender Algorithms

Find items using similar users

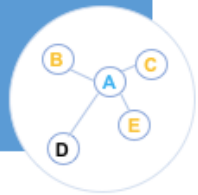
- past agreement implies future agreement
- compute similarity between users to identify user neighbourhood
- recommend items using neighbours' preferences

User-User Collaborative Filtering

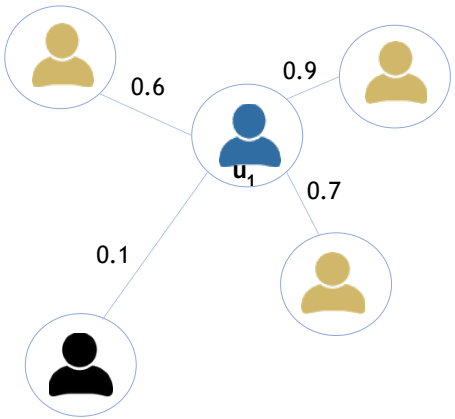
Find items using similar items

- relationships between items stable over time
- compute similarity between items to identify like items
- recommend items similar to those previously liked by user

Item-Item Collaborative Filtering



Our Implementation of User-User Collaborative Filtering

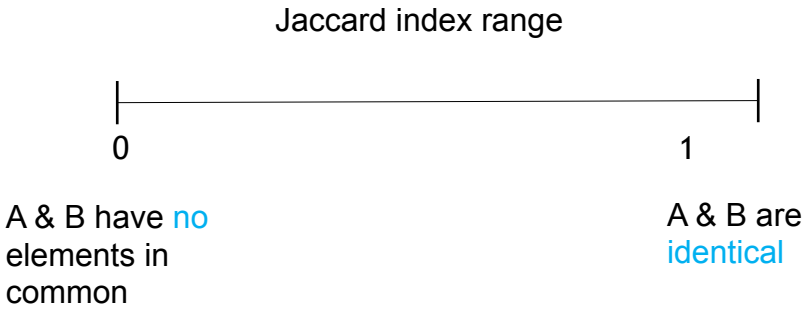


1. Compute similarity between user pairs
2. For each user u_1, select top k neighbours based on similarity
3. Identify products purchased by top k neighbours that u_1 has not yet purchased
4. Rank products by number of purchasing neighbours
5. Recommend top n products to u_1

Jaccard Index

Jaccard index measures similarity between two sets A & B

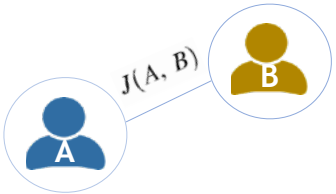
$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$



Quantifying User Similarity Using Jaccard Index

Given items purchased by users A and B, compute Jaccard index

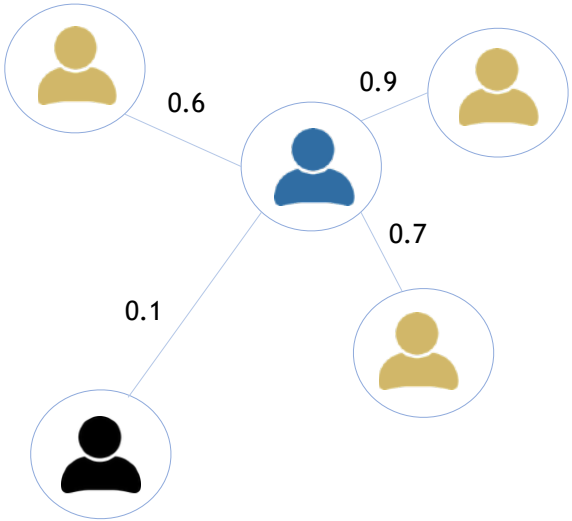
item	purchased by A?	purchased by B?
banana	1	1
apple	0	0
orange	0	1
lemon	1	0
celery	0	1



$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = 1/4 = 0.25$$

Selecting A User's Neighbourhood

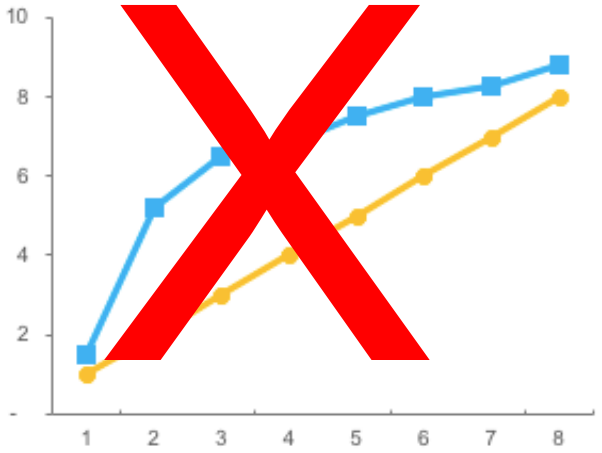
Select top k users with highest similarity to target user

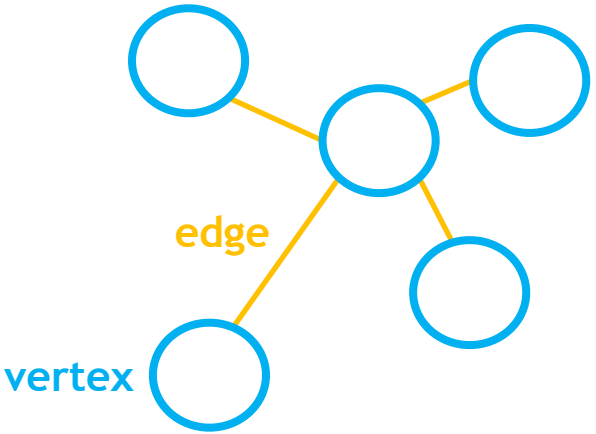


k = neighbourhood size = 3

Graph Databases

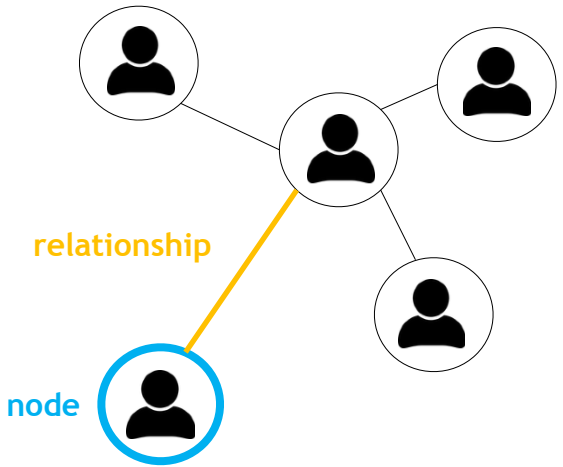
What is a Graph?





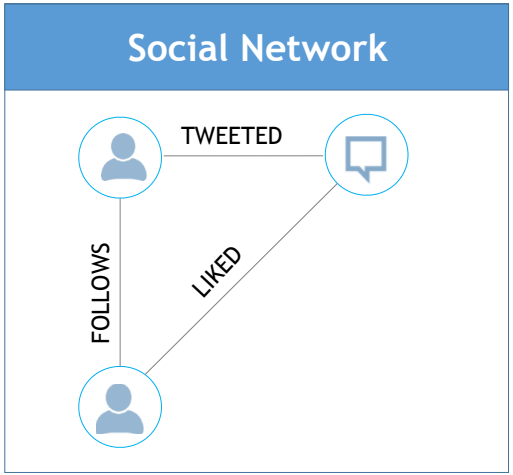
What is a Graph Database?

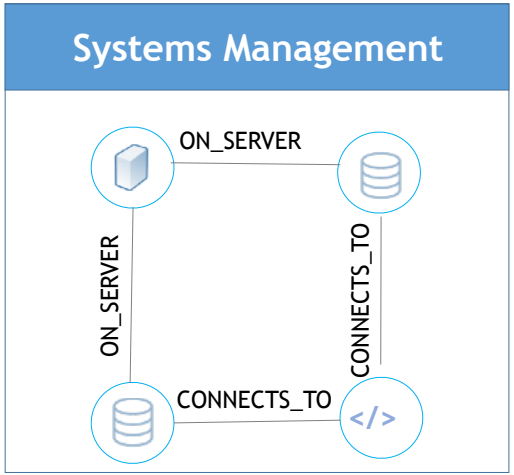
Database designed to efficiently store, process and retrieve connected data

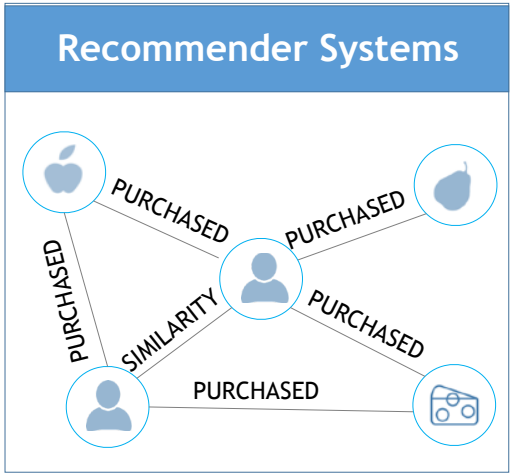


Why Use a Graph Database?

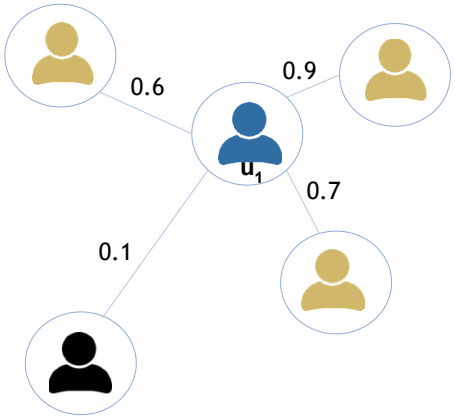
Intuitive way of representing real-world domains







Our Recommender System is a Graph!

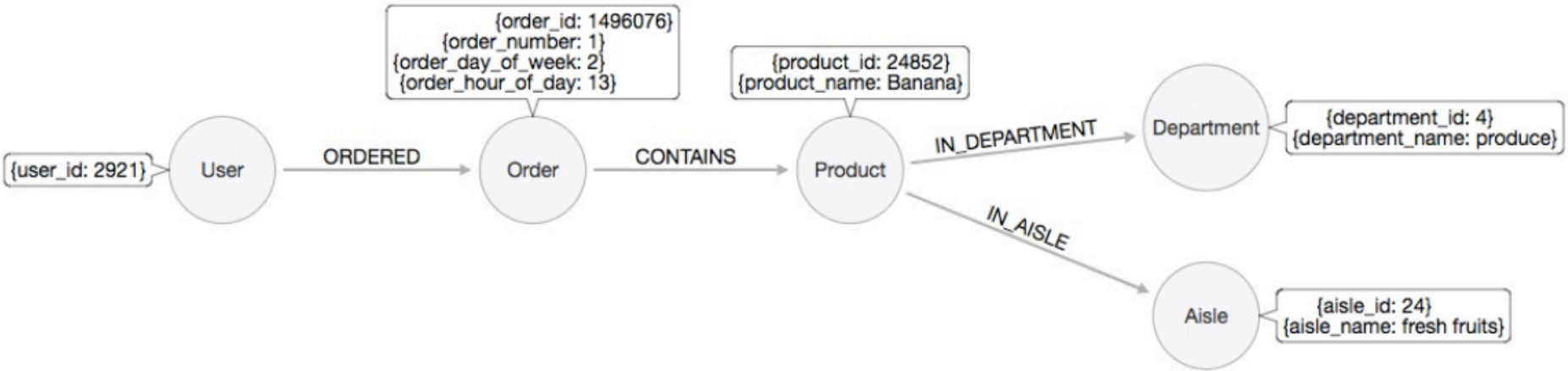


Each node represents a user

Each relationship represents similarity between two users

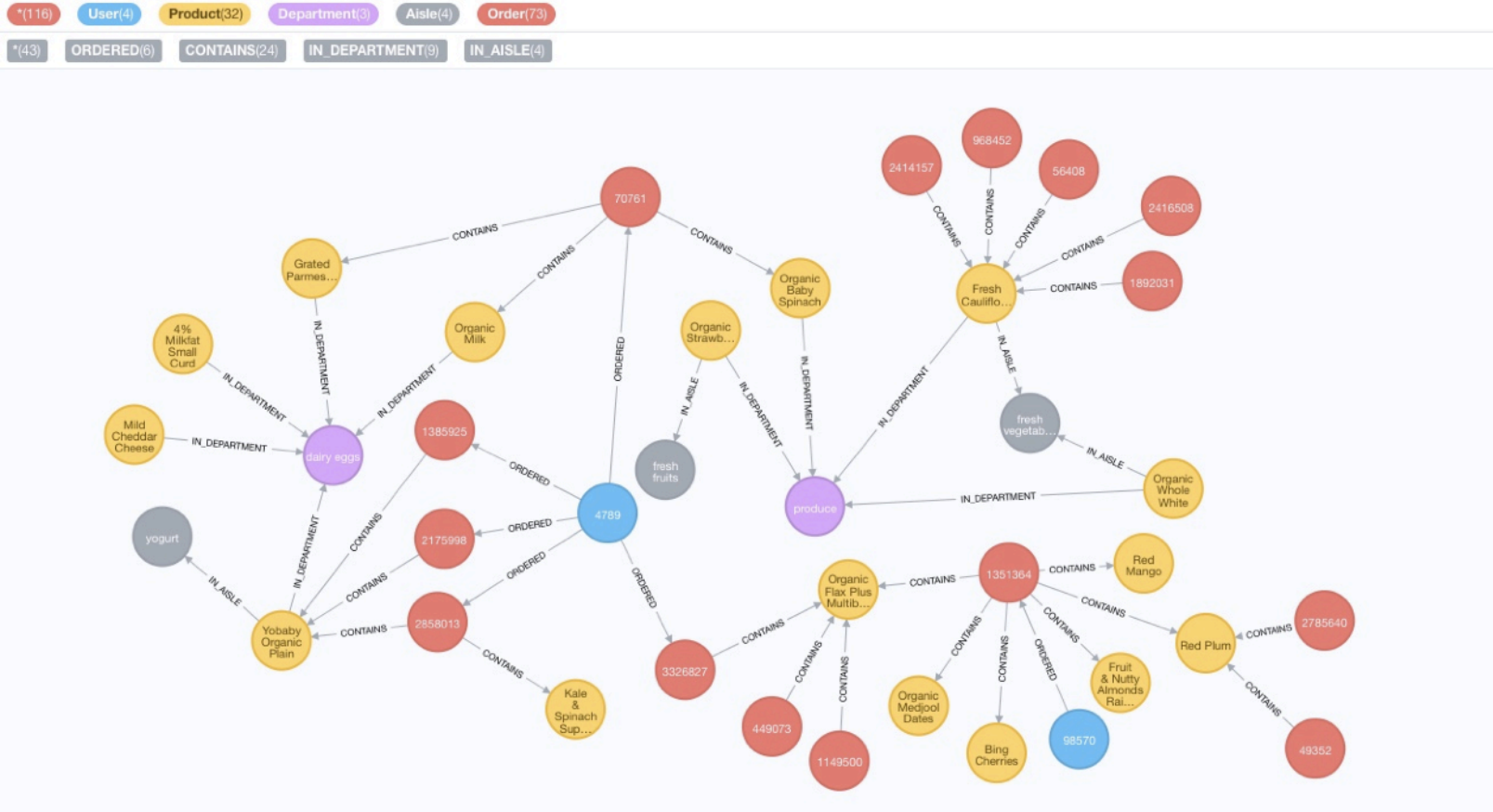
How to Model Our Data in a Graph

Nodes represent entities and connections between them represent relationships



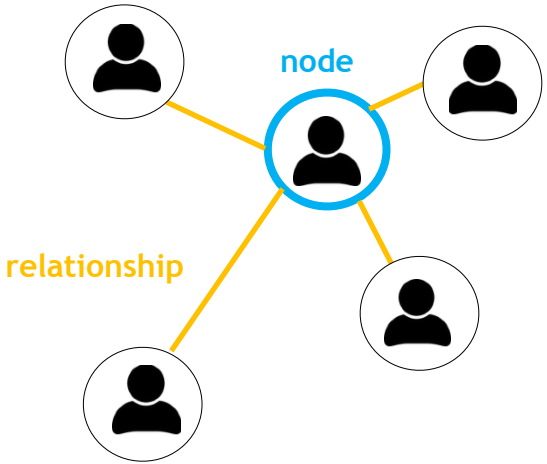
http://www.apcjones.com/arrows/#

What Our Graph Looks Like in Neo4j



Why Use Neo4j?

Each node is stored directly to its adjacent nodes and relationships



Graph traversals are faster!

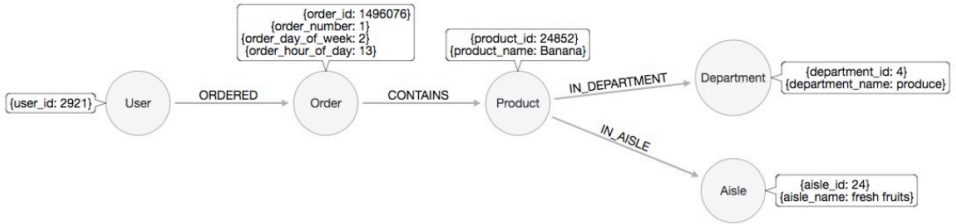
Some Real-World Use Cases of Neo4j

Working with Our Graph: Cypher Query Language

Declarative language used to query and manipulate data in a graph

```
(Order) -[:CONTAINS]-> (Product)
```

Cypher is often referred to as SQL for Graphs!



Sample Cypher Queries

Equivalent to querying one table in SQL

SELECT

```
MATCH   (p:Product)
RETURN p.product_name
```

WHERE

```
MATCH   (p:Product)
WHERE p.product_id = 24852
RETURN p.product_name
```

ORDER BY

```
MATCH   (p:Product)
RETURN p.product_name
ORDER BY p.product_name DESCENDING
```

LIMIT

```
MATCH   (p:Product)
RETURN p.product_name
ORDER BY p.product_name DESCENDING
LIMIT 10
```

Sample Cypher Queries

Equivalent to querying two tables in SQL

COUNT

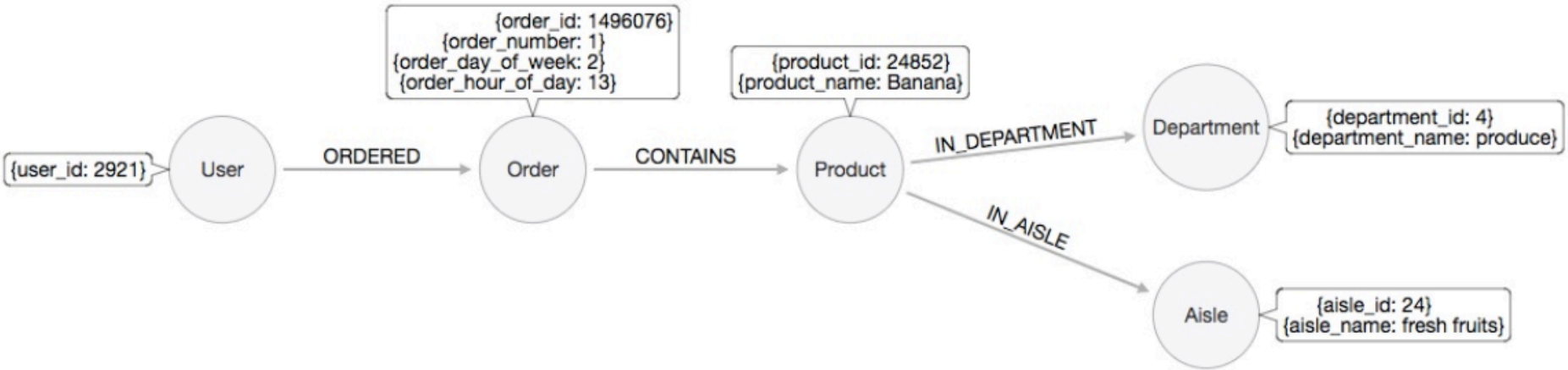
```
MATCH (p:Product)-[:IN_DEPARTMENT]->(d:Department)
WHERE d.department_name = 'produce'
RETURN COUNT(p.product_name)
```

GROUP BY

```
MATCH (p:Product)-[:IN_DEPARTMENT]->(d:Department)
RETURN d.department_name, COUNT(p.product_name)
```

Live Demo

Reminder: Our Graph Data Model



http://www.apcjones.com/arrows/#

Data to Load into Neo4j

products.csv

product_id	product_name	aisle_id	department_id
1	Chocolate Sandwich Cookies	61	19
34	Peanut Butter Cereal	121	14
45	European Cucumber	83	4
99	Local Living Butter Lettuce	83	4
116	English Muffins	93	3

3,960 records

aisles.csv

aisle_id	aisle
1	prepared soups salads
2	specialty cheeses
3	energy granola bars
4	instant foods
5	marinades meat preparation

135 records

departments.csv

department_id	department
1	frozen
2	other
3	bakery
4	produce
5	alcohol

22 records

More Data…

order_product.csv

order_id	product_id
209	39409
209	20842
209	16965
209	8021
209	23001

18,841 records

order_user.csv

order_id	user_id	order_number	order_dow	order_hour_of_day
2808127	701	1	2	14
2677145	701	2	3	11
740361	701	3	1	13
2866491	701	4	3	12
1676999	701	5	4	11

1,902 records

Loading Data Using Cypher

```
LOAD CSV WITH HEADERS FROM 'file:///neo4j_products.csv' AS line FIELDTERMINATOR '|'
WITH line

    // Create Product nodes
    CREATE (product:Product {product_id: toInteger(line.product_id), product_name: line.product_name})


    // Create Aisle and Department nodes
    MERGE  (aisle:Aisle {aisle_id: toInteger(line.aisle_id)})
    MERGE  (department:Department {department_id: toInteger(line.department_id)})


    // Create relationships between products and aisles & products and departments
    CREATE (product)-[:IN_AISLE]->(aisle)
    CREATE (product)-[:IN_DEPARTMENT]->(department);
```

products.csv

```
product_id|product_name|aisle_id|department_id
1|Chocolate Sandwich Cookies|61|19
34|Peanut Butter Cereal|121|14
45|European Cucumber|83|4
99|Local Living Butter Lettuce|83|4
116|English Muffins|93|3
122|Pomegranate Molasses|29|13
123|Sherry Reserve Vinegar|19|13
130|Vanilla Milk Chocolate Almond Ice Cream Bars Multi-Pack|37|1
133|Purifying Daily Detox Scrub|109|11
141|Restaurant Style Organic Chia & Quinoa Tortilla Chips|107|19
```

THANK YOU

Grace Tenorio

@datatheque

@WomenWhoCodeTO

SAVE THE DATE

MONDAY, MAY 28

@WomenWhoCodeTO

Thank you

@WomenWhoCodeTO

Thank you

@WomenWhoCodeTO

WOMEN WHO CODE® TORONTO

See you next time!

@WomenWhoCodeTO