Introduction to Graph Databases

Swaminathan Muthuveerappan

Agenda

- Why Graph Databases?
- Neo4j's Cypher Query Language (CQL)
- Optimization: Indexing
- Optimization: Procedures
- Optimization: Streaming
- Optimization: Queryable Relationships

What is a Database?

- Organize data for easy access
- Fast searching of documents

Types of Databases

- SQL Databases (MySQL, PostgreSQL, Google CloudSQL etc...)
- NoSQL Databases (MongoDb, Google Datastore etc...)
- Graph Databases (Neo4j, ArangoDB)

Let's design a database for an ecommerce application (promotions domain)

Specifications

1. Business should be able to create Promotions on their inventory
2. Business should be able to know the list of applicable SKUs/Variants for a given Promotion

SKU - Stock Keeping Unit

Inventory Specification

- Category
  - Mobile Phones, Electronics etc...
- Product
  - iPhone, LG OLED TV etc...
- Variant
  - iPhone 128GB Space Grey, LG Smart OLED 4K 65B9PUA etc...

(image)

Will SQL Databases work well?

How about NoSQL Databases?

Finally…

Introduction to Graph Databases :)

Why would Graph Databases work?

- Closely represent whiteboard models
- Relationships are first-class citizens
- Cheap Traversals

(image)

The Cypher Query Language

Let's create a simple graph using Cypher queries

CREATE Clause

```
CREATE (appleBrand:Brand{id:'apple_123', name:'Apple'}) RETURN appleBrand;
```

Relationships

```
MATCH (appleBrand:Brand{id:'apple_123'})
MATCH (iPhoneVariant:Variant{id:'variant_123'})
CREATE (appleBrand)-[:has_variant]->(iPhoneVariant);
```

Querying

Get all the variants of a given promotion

(image)

Optimizations

1. Indexing to improve search performance

2. Extending Neo4j with User-Defined procedures

3. Streaming for large data

4. Improving performance with proper Relationship names

Thank you for your time

:)

Questions?