CSCU9YQ - NoSQL Databases

Lecture 10b: Graph Databases

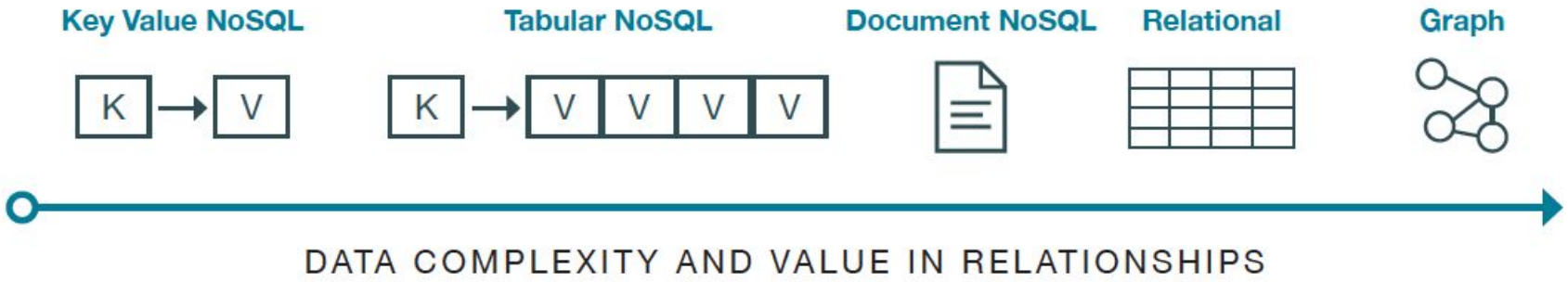
Prof Gabriela Ochoa
http://www.cs.stir.ac.uk/~goc/
University of Stirling

Resources

- DataStax Enterprise
  - Documentation: https://www.datastax.com/resources/
  - Academy: https://academy.datastax.com/resources/ds330-datastax-enterprise-graph
  - Graph Examples: https://www.youtube.com/watch?v=u7nHyzFHqMA
  - White Paper: Why Graph? A Look at How Cloud Application Benefit From Graph Technology
- Neo4J
  - Website: https://neo4j.com/

The Data Model Continuum

RDBMS and Graph (NoSQL) databases are at the high end of the data model continuum where the relationships between data are concerned and are somewhat similar in their base characteristics



Graph Databases

- Used for storing, managing and querying data that is complex and highly connected.
- Focus on data relationships
- Has explicit graph structure, each node knows its adjacent nodes
- As the number of nodes increases, the cost of a local step (or hop) remains the same
- Examples:
  - DataStax Graph, Neo4j, OrientDB, InfiniteGraph, AllegroGraph

Comparing Graph with Relational Databases

- They are similar in that they involve data that contains connections or relationships between data elements
- Differences
  - The underlying engine used to store and access data
  - Graphs DBs allows properties to be assigned to edges, which is not allowed in Relational DBs
  - Graphs DBs prioritise relationships, where Relational DB uses foreign keys to connect entities in a secondary fashion

	RDBMS	GRAPH DB
An identifiable “something” or object to keep track of	Entity	Vertex
A connection or reference between two objects	Relationship	Edge
A characteristic of an object	Attribute	Property

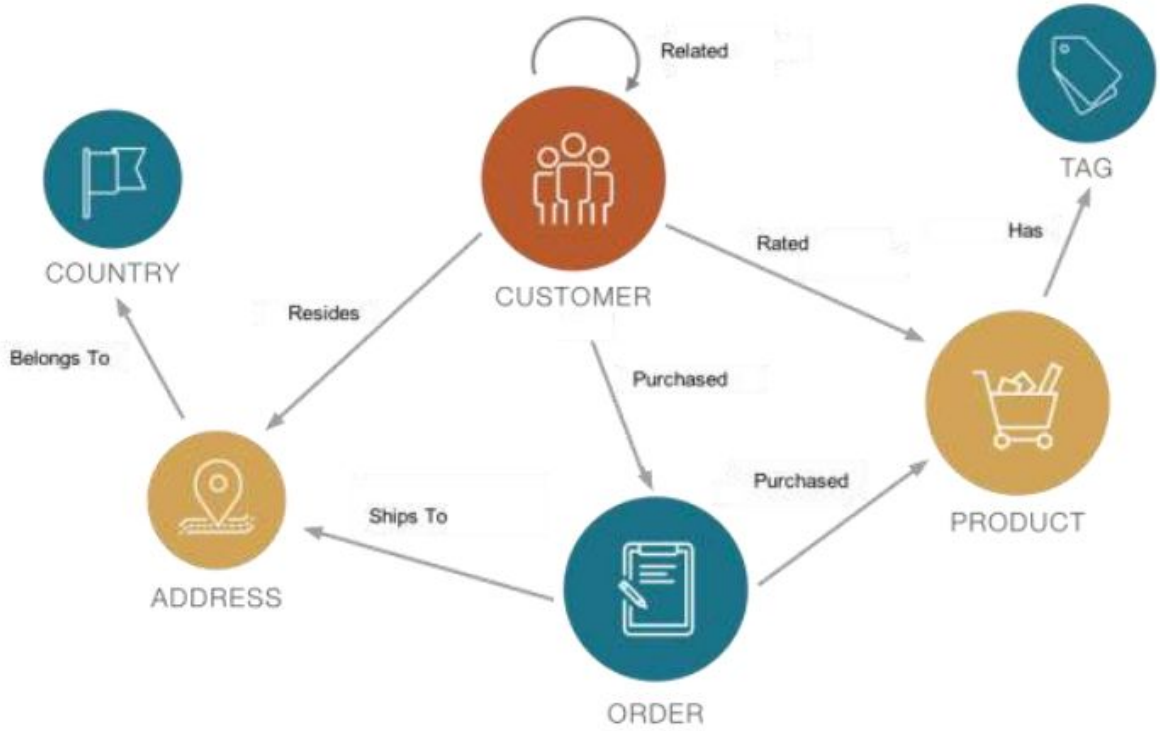
Data Model

Relationships are explicitly embedded in a graph data model.

Graphs are foundationally designed to express relatedness.

More concerned with the relationships (edges) than with the entities (vertexes)

Nodes: Entities
Edges: relationships between entities

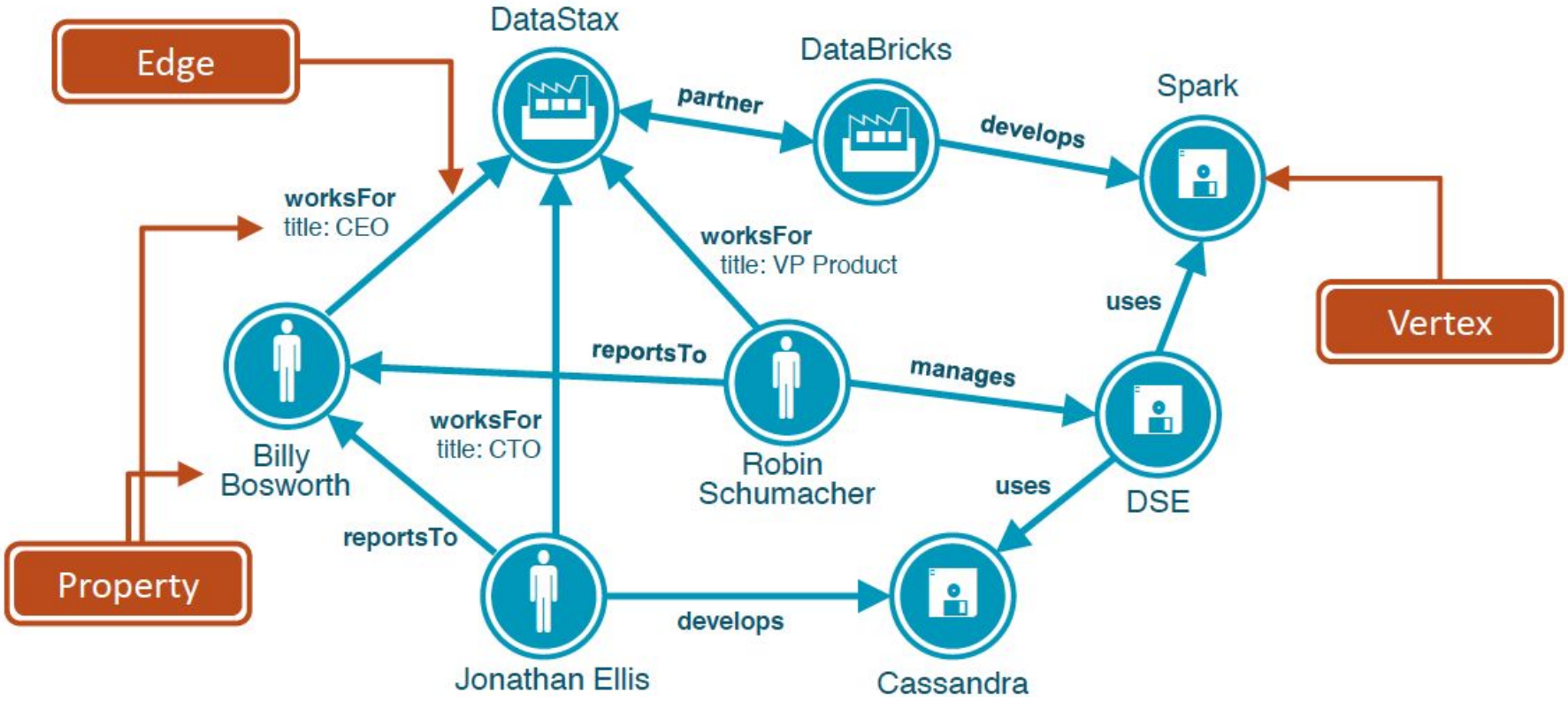


Characteristics of DataStax Enterprise Graph

- Built with Apache TinkerPop
  - An open source graph computing framework
  - Gremlin traversal language to interact with databases
  - Supporting tools: Gremlin Server, data bulk loaders, graph algorithms, visualisation
- Integrated Deeply with Apache Cassandra, Inherits its benefits
  - Constant uptime,
  - write/read/active-everywhere functionality,
  - linear scalability,
  - predictable low latency response times, and operational maturity.
- Inspired by the Open Source Titan Database
  - Titan is used by many well-known enterprises such as Amazon
  - Titan is a scale-out DB with pluggable storage back end options
  - DSE goes further than Titan’s basic scale-out capabilities by integrating with cassandra

Introduction to DataStax Enterprise Graph

Property Graph



RDBMS	DSE GRAPH
Simple to moderate data complexity	Heavy data complexity
Hundreds of potential relationships	Hundreds of thousands to millions or billions of potential relationships
Moderate JOIN operations with good performance	Heavy to extreme JOIN operations required
Infrequent to no data model changes	Constantly changing and evolving data model
Static to semi-static data changes	Dynamic and constantly changing data
Primarily structured data	Structured and unstructured data
Nested or complex transactions	Simple transactions
Always strongly consistent	Tunable consistency (eventual to strong)
Moderate incoming data velocity	High incoming data velocity (e.g. IoT)
High availability (handled with failover)	Continuous availability (no downtime)

Factors in Decision process

Relational vs. Graph

Use Cases

- Customer 360
  - Understand and analyse customer relationships consolidated across business units for a holistic view (for Business Intelligence and others)
  - A graph model is an excellent way to consolidate disparate data

- Personalisation and Recommendation
  - Enterprises need Influence customers to buy their products
  - Offer relevant products and services
  - Example A web retailer needs to recommend products to customers
    - Based on both their profile and previous interactions and purchases
    - Recent interactions on the site (browsing sessions) can be stored into a graph

Use Cases

- Security and Fraud Detection
  - Determine which entity, transaction or interaction is fraudulent, poses a security risk, or is a compliance concern
  - A graph model is useful because most transactions/interactions cannot be deemed fraudulent in isolation. Instead, they can be made in a larger context (eg. previous transactions, past customer behaviour)
- IoT and Networking
  - IoT uses cases involve devices or machines that generate time-series information such as event and status data.
  - A graph model allows to perform analysis on relationships among data elements. Understand how they relate to each other over time.

Kuva 1. Internet of Things. Lähde: Huffington Post

SQL

```
SELECT TOP (5) [t14].[ProductName]
    FROM (SELECT COUNT(*) AS [value],
                 [t13].[ProductName]
            FROM [customers] AS [t0]
     CROSS APPLY (SELECT [t9].[ProductName]
                   FROM [orders] AS [t1]
              CROSS JOIN [order details] AS [t2]
              INNER JOIN [products] AS [t3]
                      ON [t3].[ProductID] = [t2].[ProductID]
              CROSS JOIN [order details] AS [t4]
              INNER JOIN [orders] AS [t5]
                      ON [t5].[OrderID] = [t4].[OrderID]
               LEFT JOIN [customers] AS [t6]
                      ON [t6].[CustomerID] = [t5].[CustomerID]
              CROSS JOIN ([orders] AS [t7]
                          CROSS JOIN [order details] AS [t8]
                          INNER JOIN [products] AS [t9]
                                  ON [t9].[ProductID] = [t8].[ProductID])
                  WHERE NOT EXISTS(SELECT NULL AS [EMPTY]
                                     FROM [orders] AS [t10]
                                CROSS JOIN [order details] AS [t11]
                                INNER JOIN [products] AS [t12]
                                        ON [t12].[ProductID] = [t11].[ProductID]
                                     WHERE [t9].[ProductID] = [t12].[ProductID]
                                       AND [t10].[CustomerID] = [t0].[CustomerID]
                                       AND [t11].[OrderID] = [t10].[OrderID])
                    AND [t6].[CustomerID] <> [t0].[CustomerID]
                    AND [t1].[CustomerID] = [t0].[CustomerID]
                    AND [t2].[OrderID] = [t1].[OrderID]
                    AND [t4].[ProductID] = [t3].[ProductID]
                    AND [t7].[CustomerID] = [t6].[CustomerID]
                    AND [t8].[OrderID] = [t7].[OrderID]) AS [t13]
          WHERE [t0].[CustomerID] = N'ALFKI'
       GROUP BY [t13].[ProductName]) AS [t14]
ORDER BY [t14].[value] DESC
```

SQL vs Gremlin for a Recommendation Query

SQL vs Gremlin for a Recommendation Query

Gremlin

```
g.V().has("customerId","ALFKI").as("customer").
  out("ordered").out("contains").out("is").aggregate("products").
    in("is").in("contains").in("ordered").where(neq("customer")).
  out("ordered").out("contains").out("is").where(not(within("products"))).
  groupCount().by("name").
    order(local).by(values,decr).
    select(keys).limit(5)
```

	DSE GRAPH	NEO4J	TITAN (using backend like Amazon DynamoDB, HBase, etc.)	TITAN (using Cassandra backend)
Open Development Language	Yes	Yes	Yes	Yes
Architecture	Masterless	Master-Slave	Master-Slave	Masterless
Consistency Model	Tunable	Tunable	Not tunable	Tunable
Scaling Method	Scale out read/writes	Scale up	Scale out for reads	Scale out reads/ writes
Write, Read, Active-Anywhere	Yes	No	No	Yes
Auto Multi-Data Center and Cloud Zone Support	Yes	No	No	Yes
Advanced Security Features	Yes	No	No	No
Integrated Analytics	Yes	No	No	No
Integrated Search	Yes	No	No	No

Comparison with other Graph Databases

Summary

- Used for storing, managing and querying data that is complex and highly connected. Focus on data relationships
- Data Model: Property Graph
- Compared Graph with Relational Databases
- Features of DataStax Enterprise Graph
- Use Cases