28-29-30//NOVEMBRE 2017

OverNet EDUCATION

La più importante **conferenza italiana** sulle tecnologie **Microsoft**

WPC 2017 MILANO

28-29-30/NOVEMBRE

[Graph DB] in SQL Server 2017, Azure SQLDB e Azure Cosmos DB : da 0 a 100 in 60 minuti

IGOR PAGLIAI

CLOUD AND DATA ARCHITECT

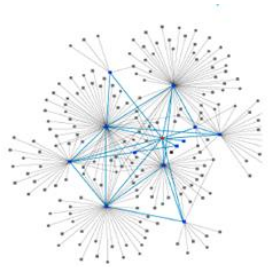
CSE INDUSTRY TEAM

MICROSOFT CORP.

@igorpag

Agenda

- Introduzione ai “Grafi” - 101
  - Quali tecnologie
  - Modello relazionale vs. grafi
  - Scenari e concetti di base
- SQL Server 2017 & Azure SQLDB
  - Supporto e funzionalità
  - Limitazioni e «roadmap»
- Azure Cosmos DB
  - Caratteristiche e funzionalità
  - Supporto specifico ai grafi
  - Gremlin!
- Comparazione
  - PRO e CONTRO di ogni soluzione
- Q&A



Introduzione ai Grafi

Eulero, Il problema dei ponti di Königsberg

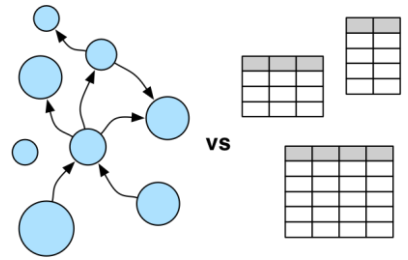
Rise of graph databases

- Rising in popularity: why?
- Gartner: Leveraging 5 graphs provides *sustained competitive advantage*
  - social, intent, consumption, interest, and mobile
- Major database vendors
  - Neo4j Inc.
  - Azure Cosmos DB (Microsoft ☺)
  - OrientDB
  - Titan + Cassandra
  - Giraph (Facebook)
- Industry leaning towards multi-model

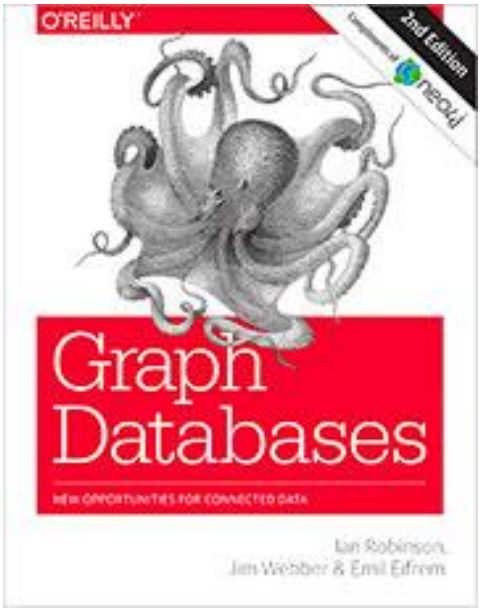
27 systems in ranking, November 2017

Rank			DBMS	Database Model	Score		
Nov 2017	Oct 2017	Nov 2016			Nov 2017	Oct 2017	Nov 2016
1.	1.	1.	Neo4j	Graph DBMS	38.45	+0.50	+1.70
2.	2.	↑ 4.	Microsoft Azure Cosmos DB	Multi-model	13.03	+0.40	+9.78
3.	3.	↓ 2.	OrientDB	Multi-model	6.10	-0.03	+0.03
4.	4.	↓ 3.	Titan	Graph DBMS	5.68	+0.21	+0.22
5.	5.	↑ 6.	ArangoDB	Multi-model	3.15	-0.01	+0.87
6.	6.	↓ 5.	Virtuoso	Multi-model	1.88	+0.03	-0.67
7.	7.	7.	Giraph	Graph DBMS	1.02	-0.01	+0.05
8.	8.	↑ 12.	GraphDB	Multi-model	0.74	+0.10	+0.56
9.	9.	9.	AllegroGraph	Multi-model	0.60	-0.02	+0.12
10.	10.	↓ 8.	Stardog	Multi-model	0.54	+0.01	+0.01

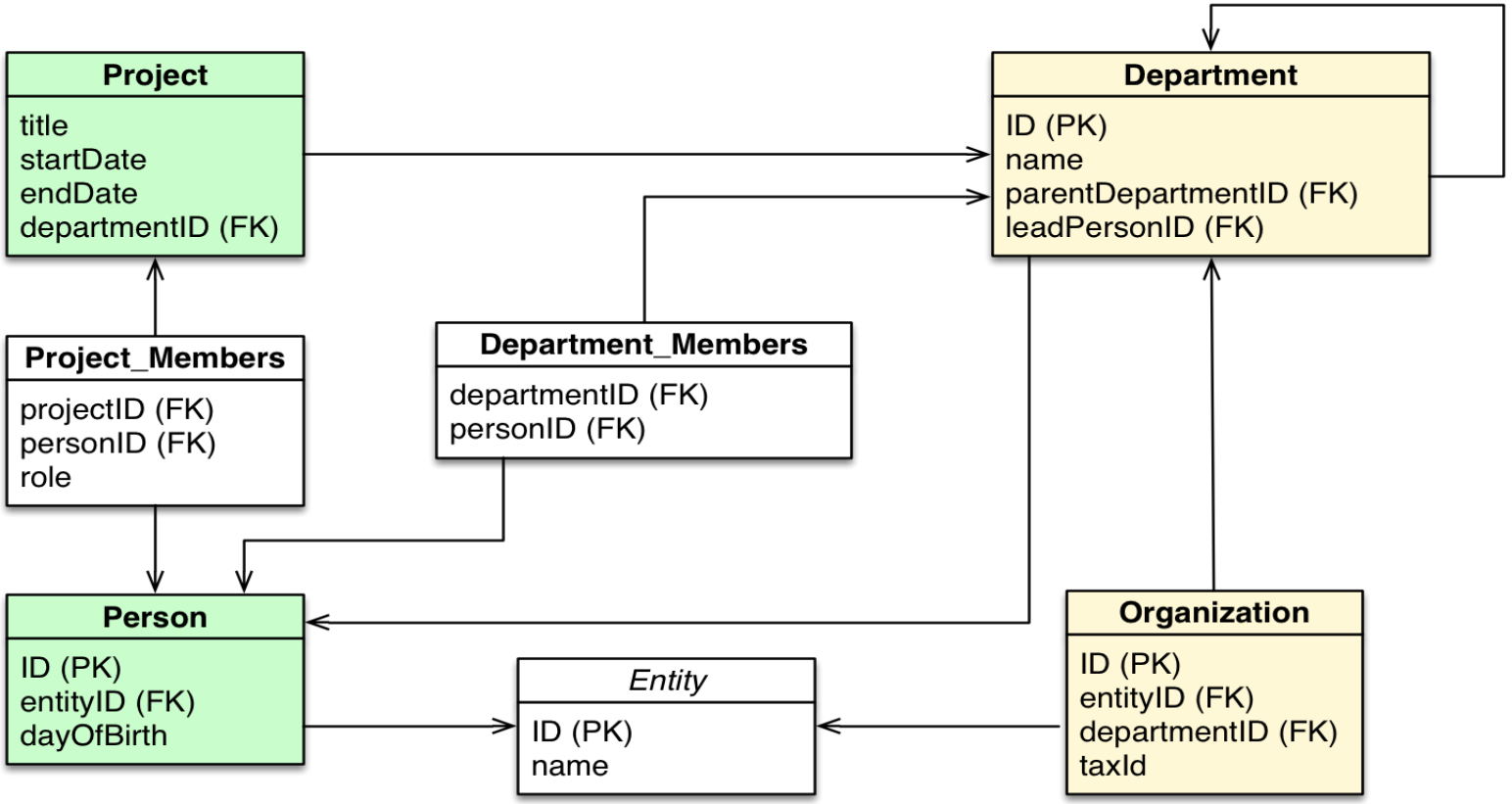
Neo4j



- Most famous and widely adopted
  - Graph model only, no multi-model, OSS, Java
- Cypher primary language, Gremlin also supported
- Interesting quote from author: "*Relational models **lacks** relations*!"
  - Relational model is not natural and force logical remapping
  - M:M relationships, Normal Forms, etc.
- Native "Compute" and "Storage" layers, no hybrid
  - "Index-free Adjacency"
- Native algorithm support
  - Transitive closure, arbitrary length traversal, etc.
- Super efficient storage layer
  - Fixed length records: offset calculation by ObjectID
  - Separate files for Edge and Nodes
  - External storage for blobs/binary
- No PaaS or SaaS (so far)
- No global distribution (so far)
- No horizontal partitioning/sharding (so far)



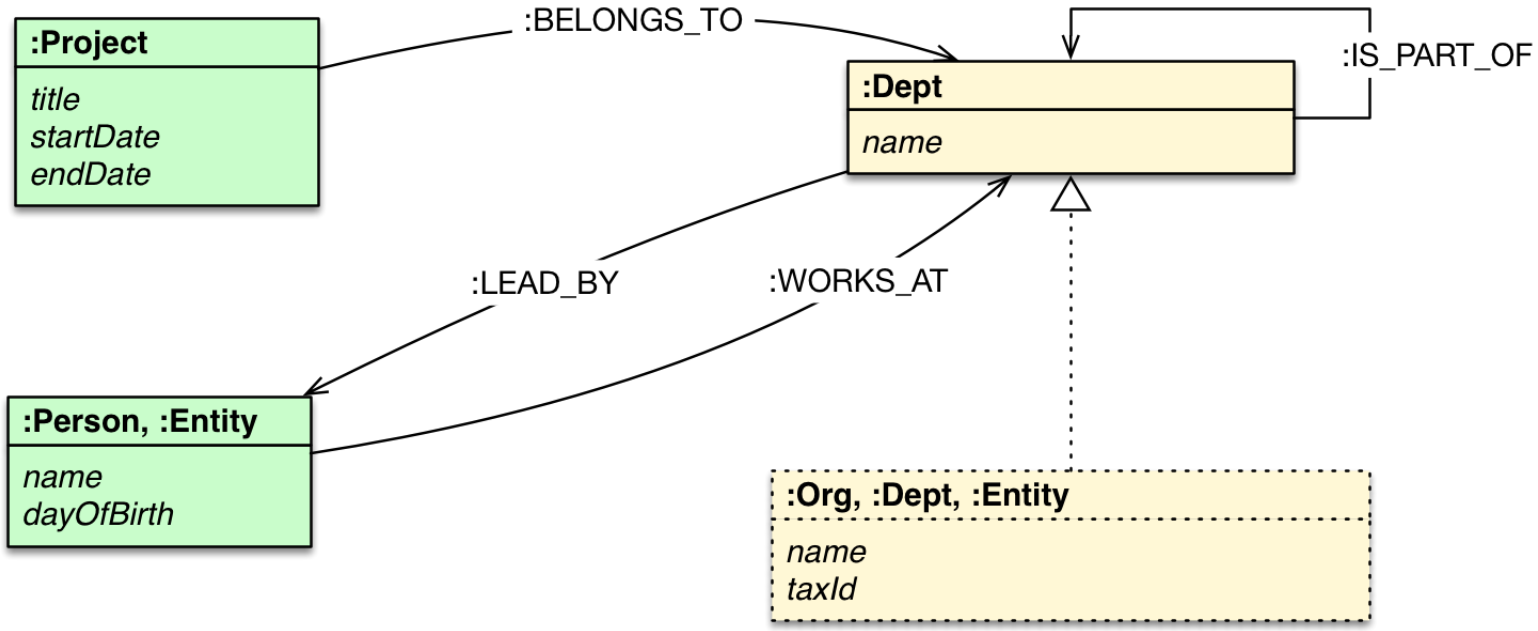
Relational vs. Graph: Models



- 1NF – First normal form
- 2NF – Second normal form
- 3NF – Third normal form
- EKNF – Elementary key normal form
- BCNF – Boyce–Codd normal form
- 4NF – Fourth normal form
- ETNF – Essential tuple normal form
- 5NF – Fifth normal form
- 6NF – Sixth normal form
- DKNF – Domain/key normal form

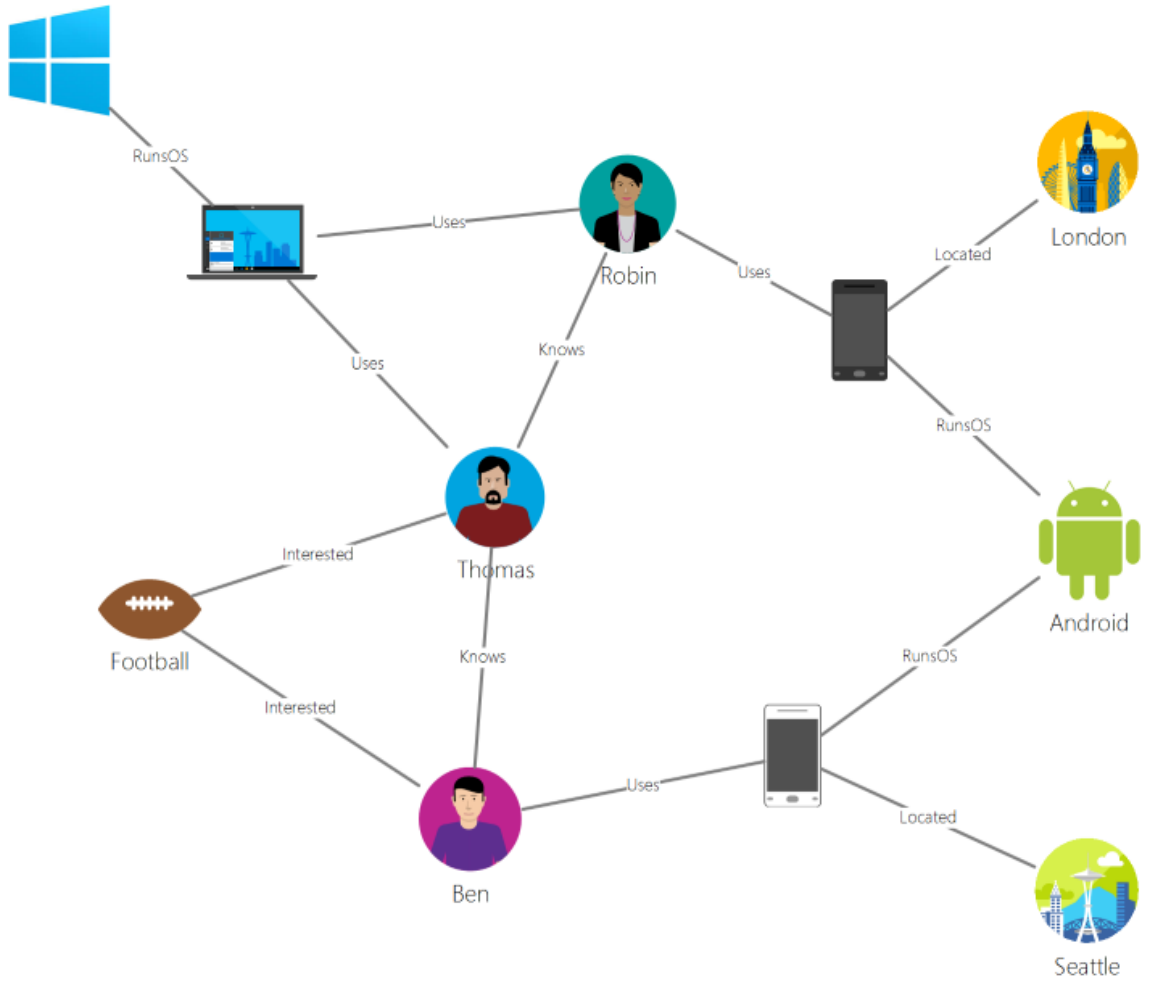
Source: https://neo4j.com/developer/graph-db-vs-rdbms

Relational vs. Graph: Models



Why use graph databases?

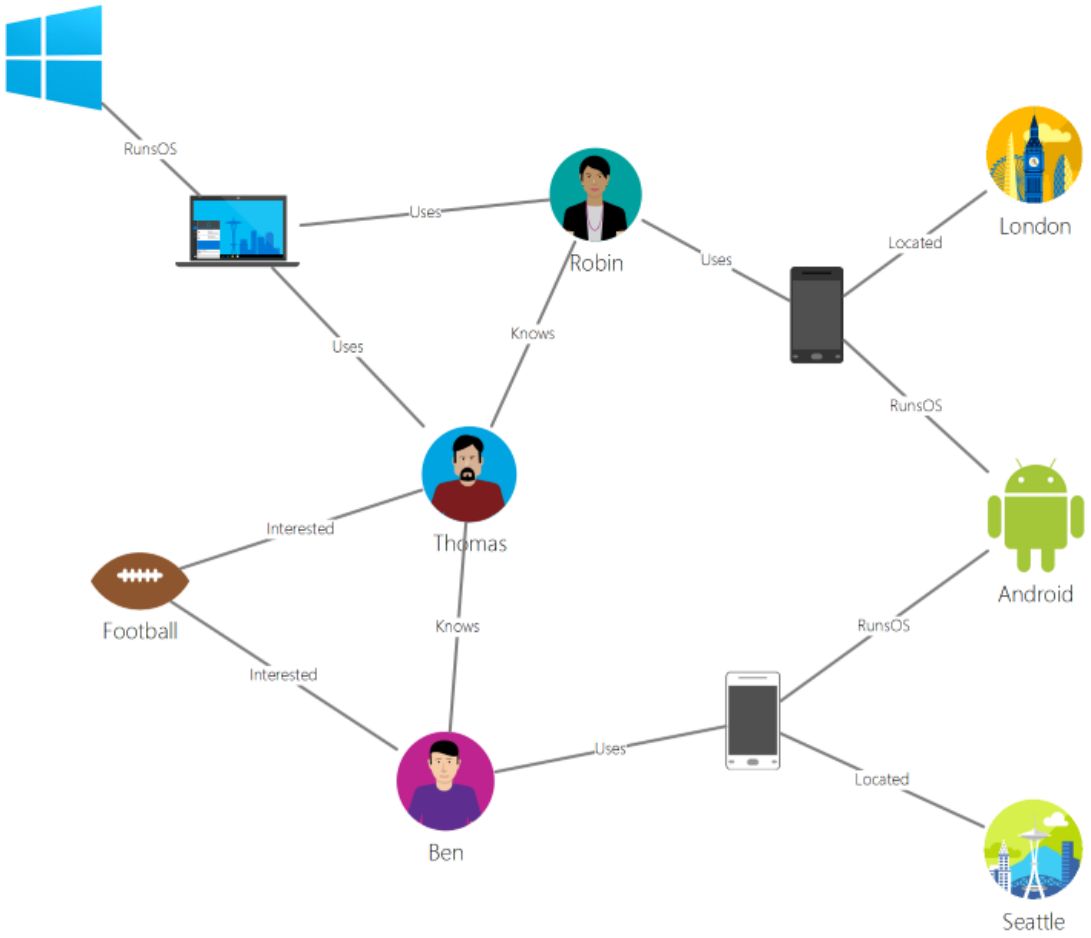
- Natural and intuitive way to model the real world
- Graph traversal/navigation
- Relationships are first-class concepts
- No Joins!
- Built for fast graph computations
- Graph databases are traditionally flexible, No Schema (NoSQL family)



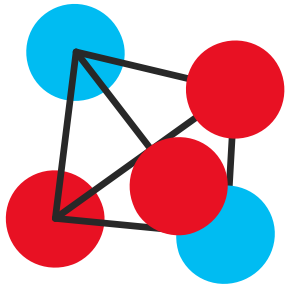
Graph database concepts

Vertices, Edges and Properties

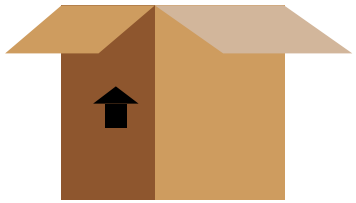
- Vertex (or Node) = any entity
- Edge = relationship, connects two vertices
- Vertex and edge have “Labels” and “Properties” (Key-Value)
- Graph = G(V,E) = Collection of Vertexes and Edges
- Traversals = query over graphs



Graph database scenarios

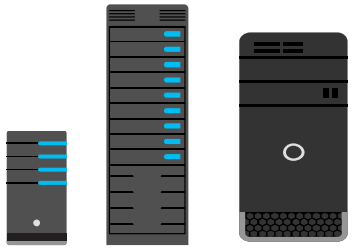


Social Networking



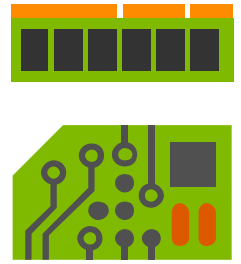
Recommendations

Security &
Fraud Detection



Network & Operations

Customer 360



IoT (Logistics)

OverNet EDUCATION

SQL Server 2017 & Azure SQLDB

GRAPH PROCESSING WITH SQL SERVER 2017& AZURE SQL DATABASE

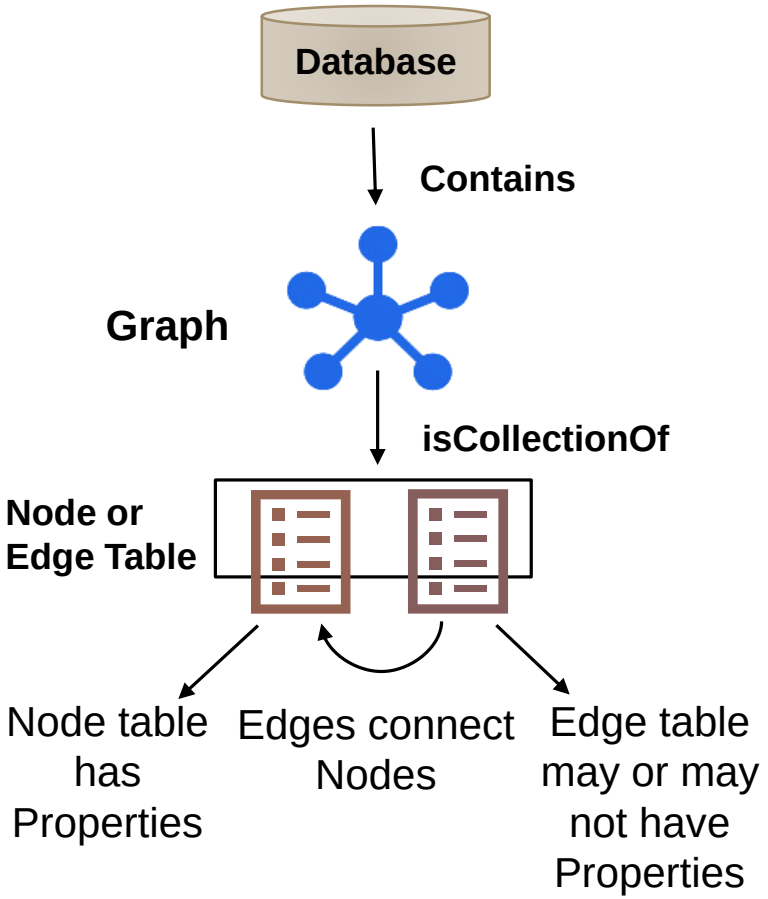
Introducing SQL Graph

A collection of Node and Edge *tables* in the database

Language Extensions

- DDL Extensions – create node/edge tables
- Query Language Extensions – New built-in: **MATCH**, to support pattern matching and traversals

Tooling and Eco-system

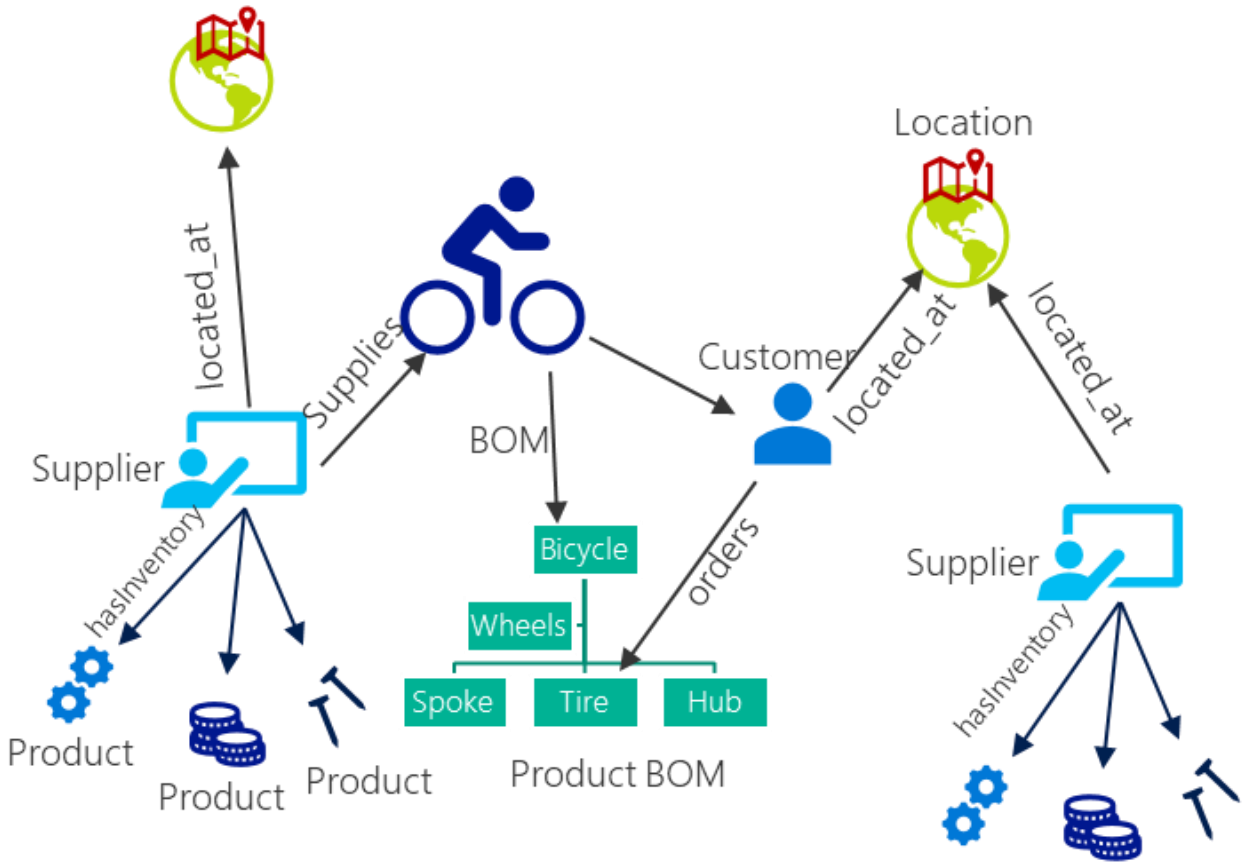


DDL Extensions

Create Nodes and Edges

Properties associated with Nodes and Edges

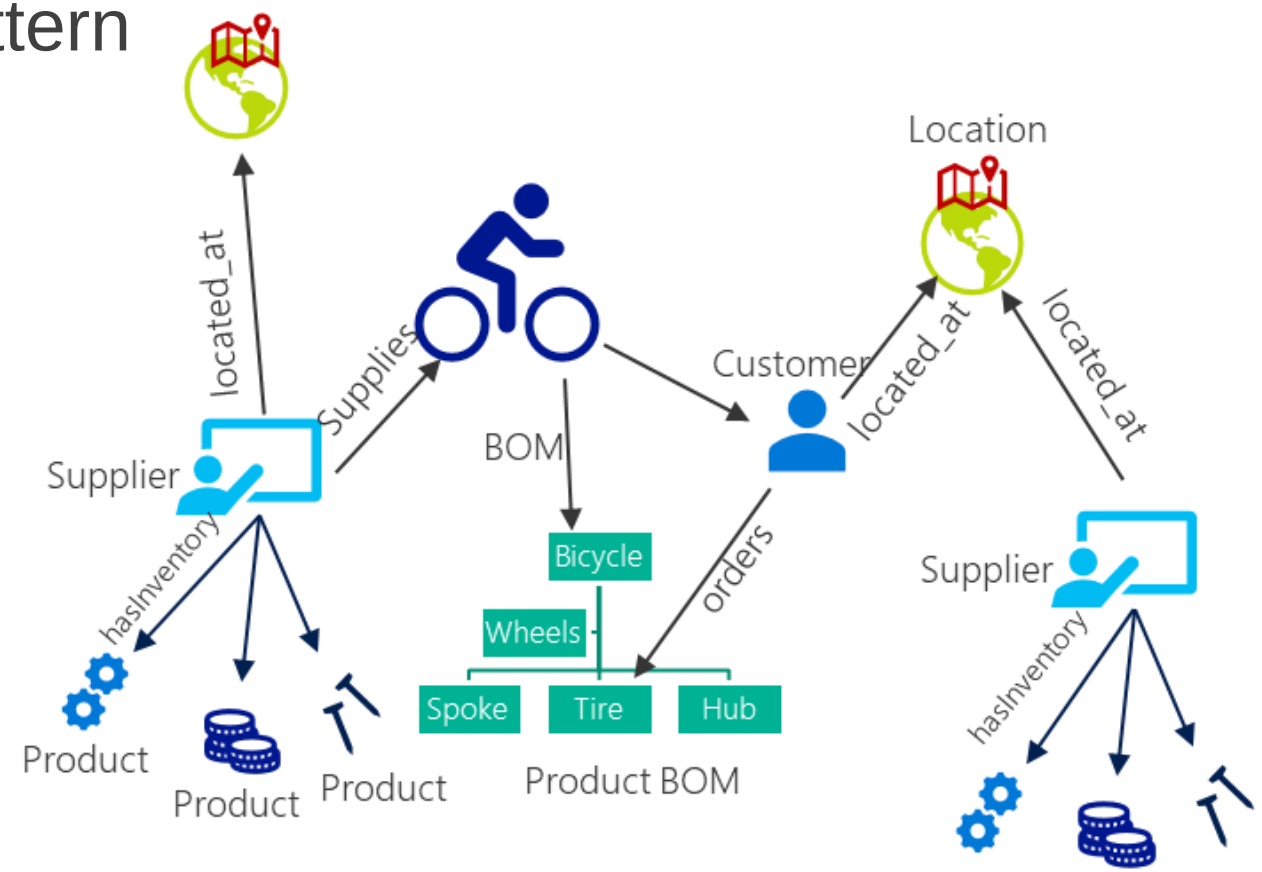
```
CREATE TABLE Product (ID INTEGER
PRIMARY KEY, name VARCHAR(100)) AS
NODE;

CREATE TABLE Supplier (ID INTEGER
PRIMARY KEY, name VARCHAR(100)) AS
NODE;

CREATE TABLE hasInventory AS EDGE;

CREATE   TABLE   located_at(address
varchar(100)) AS EDGE;
```



Query Language Extensions

Multi-hop navigation and join-free pattern matching using **MATCH** predicate:

```
SELECT Prod.name as ProductName,
   Sup.name as SupplierName
FROM Product Prod, Supplier Sup,
  hasInventory hasIn,
  located_at supp_loc,
  Customer Cus,
  located_at cust_loc,
  orders, location loc
WHERE
MATCH(
  cus-(orders)->Prod<-(hasIn)-Sup
)
```



Integrated into the SQL Engine

Existing tools all work out of the box, for example SSMS, backup and restore, import / export, etc

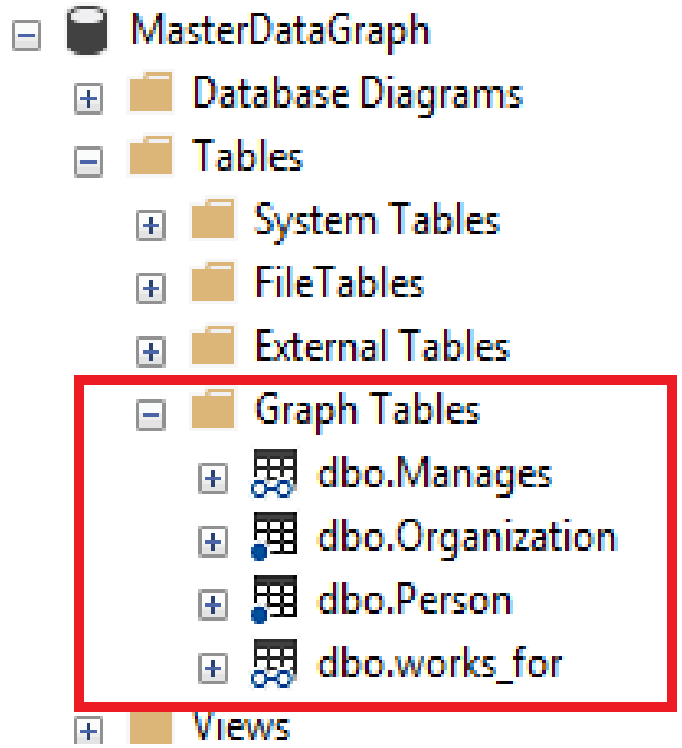
Queries can join existing tables and graph node / edge tables

Columnstore Indexes, ML (R / Python), HA etc.

Security and Compliance: Dynamic Data Masking, Row Level Security

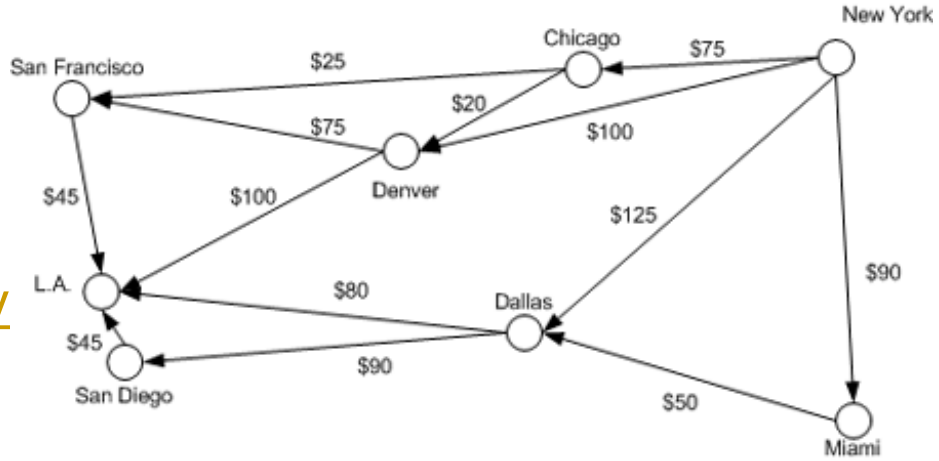
Available on SQL Server on Linux as well!

NEW tool: SQL Operations Studio (SQLOpsStudio) for Windows, Linux and MacOS



Relational vs. Graph

- Graph and relational designs can answer the same questions
  - But if traversal of relationships define the primary application requirements, Graph can solve this more intuitively and with less code
- Shortest Paths
  - Find all paths from a given node to another node
  - Find shortest paths to all other (reachable) nodes
  - Sample T-SQL implementation in https://github.com/arvindshmicrosoft/YelpDatasetSQLServer/3_GraphAlgorithms.sql
  - Gremlin Example:




```
g.V('robin')
 .repeat(out())
 .until(has('id', 'ben'))
 .path()
```

Why SQL Graph over 100s of other Graph DBs?

SQL Server and SQL DB as the main Data Hub

1. Bring computation closer to data: Machine Learning & AI, Blobs Hadoop, Graph
2. Handle all data formats : relational, FileStream, XML, JSON
3. Deployment flexibility: Linux, Windows, Docker & Containers, Azure, On-Premises
4. Enterprise grade features: security, audit, encryption, HA&DR, performances, etc.
5. Familiar: continue using existing skills – no surprises!

Current limitations

- Future Improvements
  - Shortest Path native implementation
  - Arbitrary length traversals:
    Find friends 1-5 hops away

```
MATCH (Person-(isFriendOf*1:5)->Person)
```

  - Heterogeneous associations
    - Find Sessions **or** Speakers that one likes
  - Language enhancements – MERGE DML
    - Create if not exist
  - Edge Constraints
    - Edges can connect only to specific Nodes

DEMO

Using Graph DB support in SQL Server 2017 and Azure SQLDB

Azure Cosmos DB

Linked Session: DEV006 - CosmosDB - La nuova frontiera del BigData e NoSql
(Today in Sala Verde at 5:00pm)

Bring your data (and API) as it is

Apache Cassandra

Table API

SQL

MongoDB API

Key-value

Column-family

Document

Graph

Guaranteed low latency at the 99th percentile

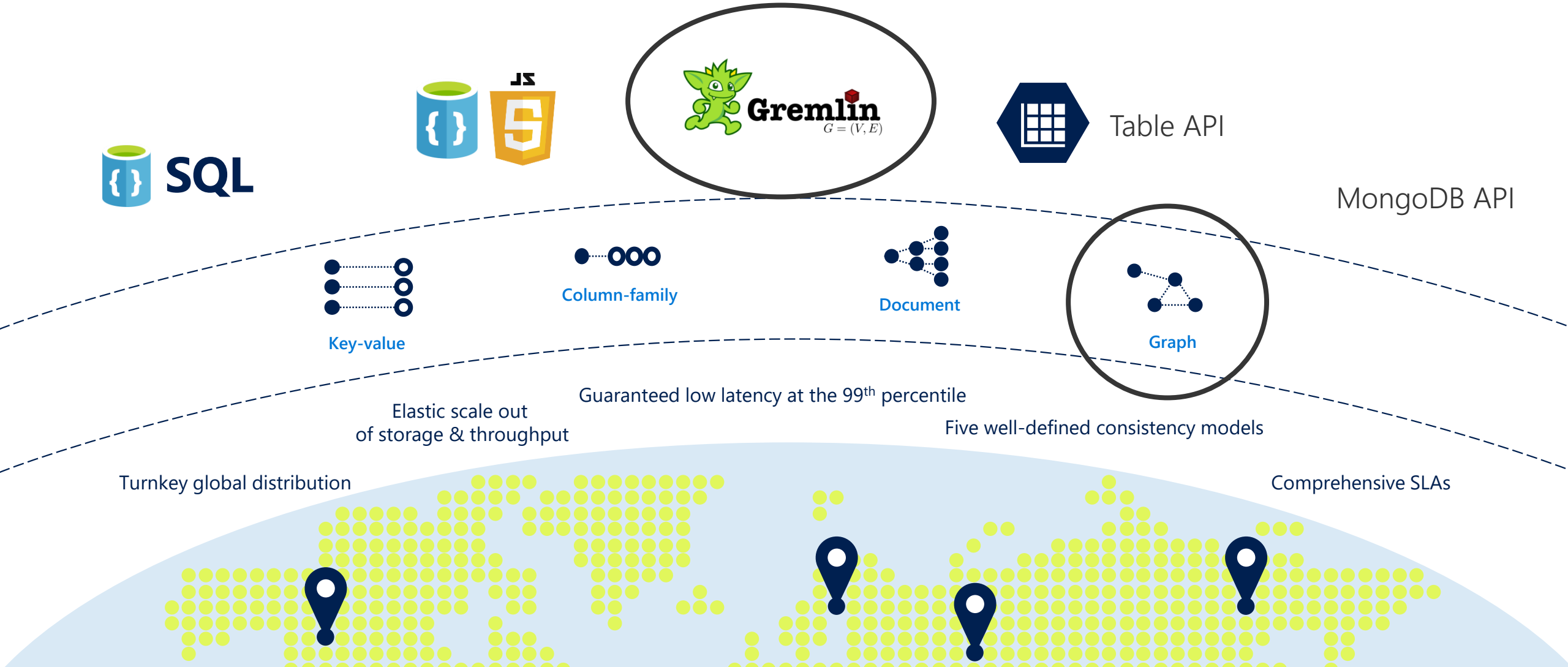
Elastic scale out
of storage & throughput

Five well-defined consistency models

Turnkey global distribution

Comprehensive SLAs

Azure Cosmos DB Graph API



Latency & Consistency Models

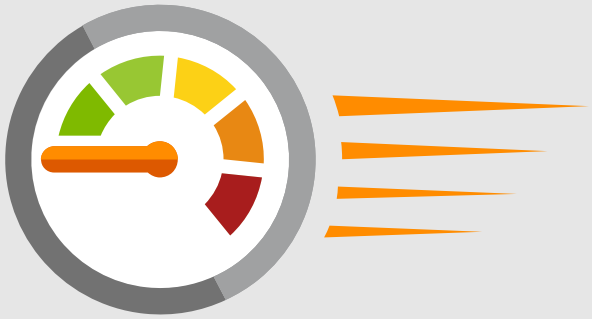
Guaranteed low latency at P99 (99th percentile)

Requests are served from local region

Single-digit millisecond latency worldwide

Write optimized, latch-free database engine designed for SSD

Synchronous automatic indexing at sustained ingestion rates



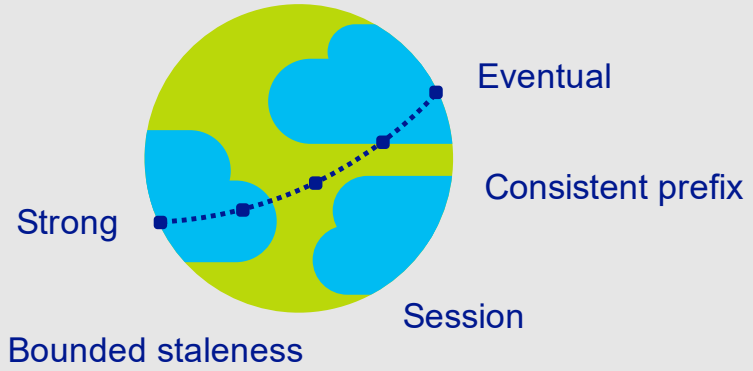
Multiple, well-defined consistency choices

Global distribution forces us to navigate the CAP theorem

Writing correct distributed applications is hard

Five well-defined consistency levels

Programmatically change at anytime, can be overridden on a per-request basis



Scalability & Indexing

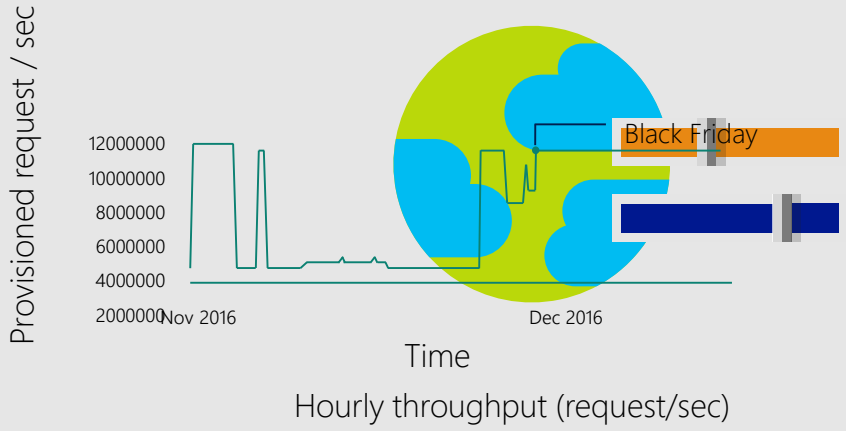
Elastically scalable storage and throughput (RU)

Single machine is never a bottle neck, transparent server-side partition management

Elastically scale storage (GB to PB) and throughput (100 to 100M req/sec) across many machines and multiple regions

Automatic expiration via policy based TTL

Pay by the hour, change throughput at any time for only what you need



Schema-agnostic, automatic indexing

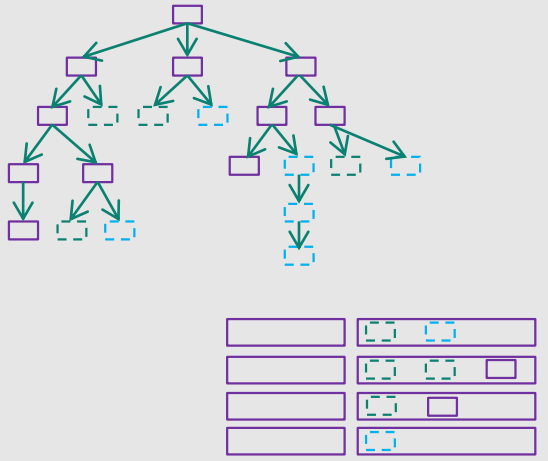
At global scale, schema/index management is painful

Automatic and synchronous indexing

Hash, range, and geospatial

Works across every data model

Highly write-optimized database engine



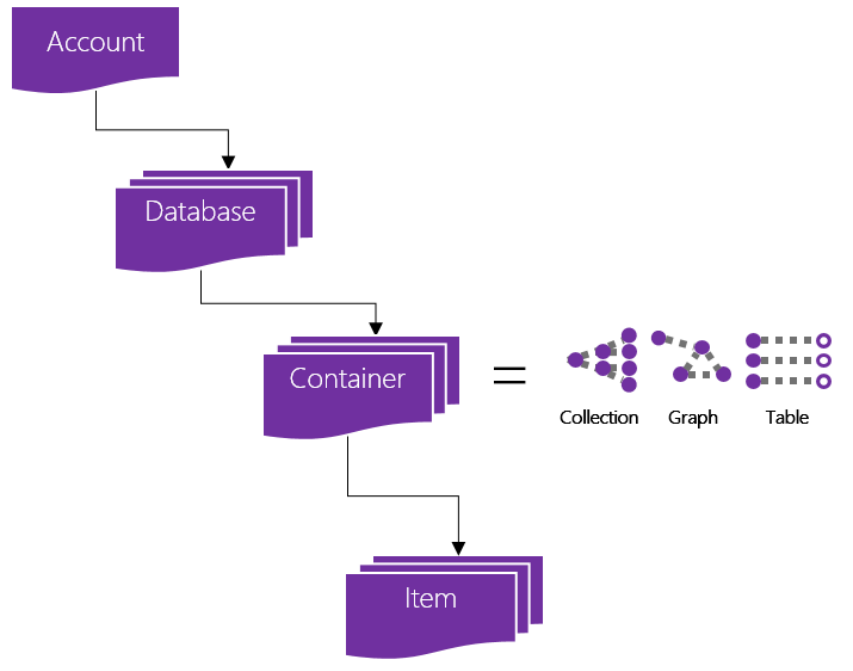
Partitioning

Resource Model

Resources identified by their logical and stable URI

Hierarchical overlay over horizontally partitioned entities; spanning machines, clusters and regions

Extensible custom projections based on specific type of API interface



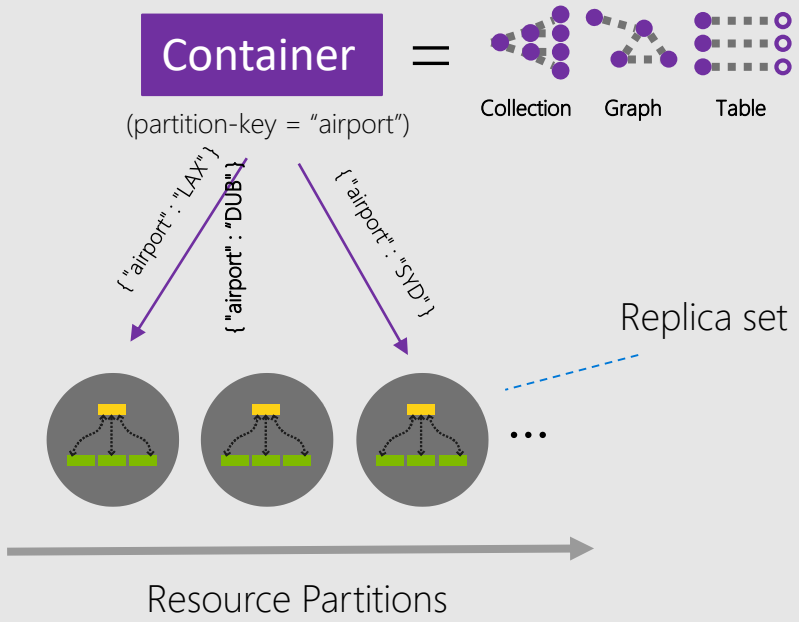
Horizontal Partitioning

Containers are horizontally partitioned

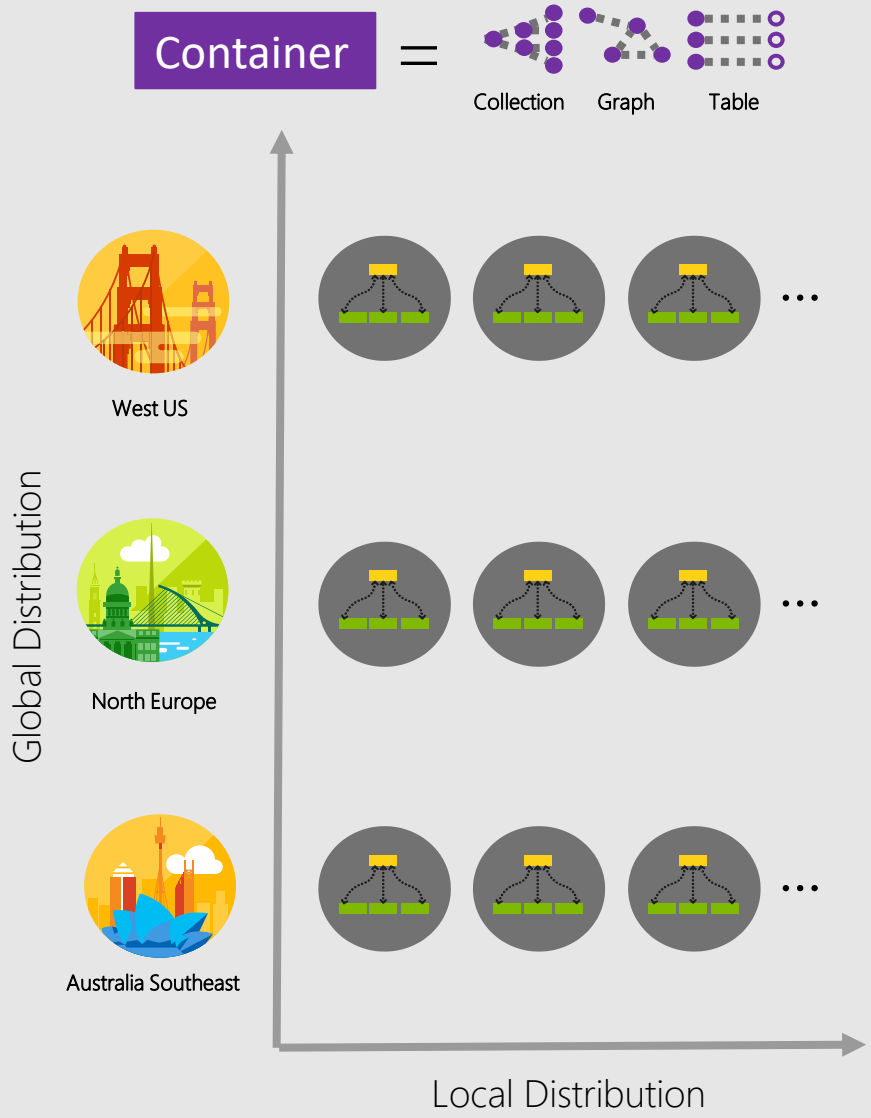
Each partition made highly available via a replica set

Partition management is transparent and highly responsive

Partitioning scheme is dictated by a “partition-key”



Global Distribution



Global Distribution

All resources are horizontally partitioned and vertically distributed

Distribution can be within a cluster, x-cluster, x-DC or x-region

Replication topology is dynamic based on consistency level and network conditions

The service handles routing query requests to the right partition using the partition key

Best Practices: Partitioning

All items with the same partition key will be stored in the same partition

Multiple partition keys may share the same partition using hash-based partitioning

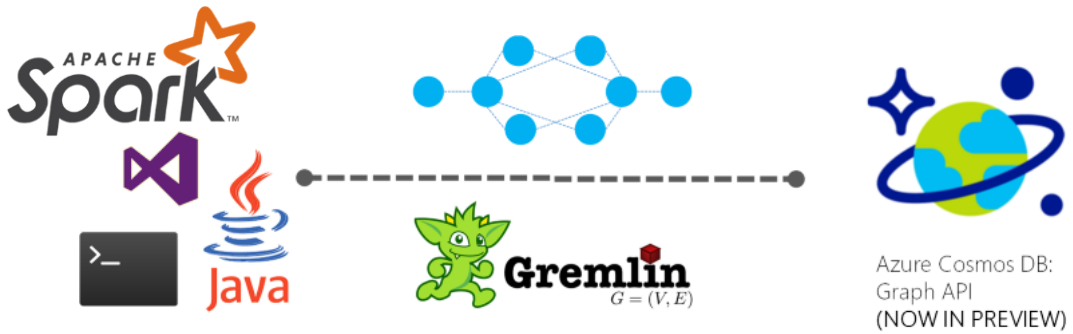
Select a partition key which provides even distribution of storage and throughput (req/sec) at any given time to avoid storage and performance bottlenecks

Partition key should be represented in the bulk of queries for read heavy scenarios to avoid excessive fan-out.

Partition key is the boundary for cross item transactions. Select a partition key which can be a transaction scope.

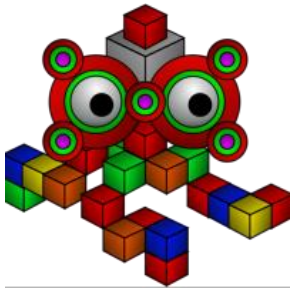
APIs and programmability: Gremlin!

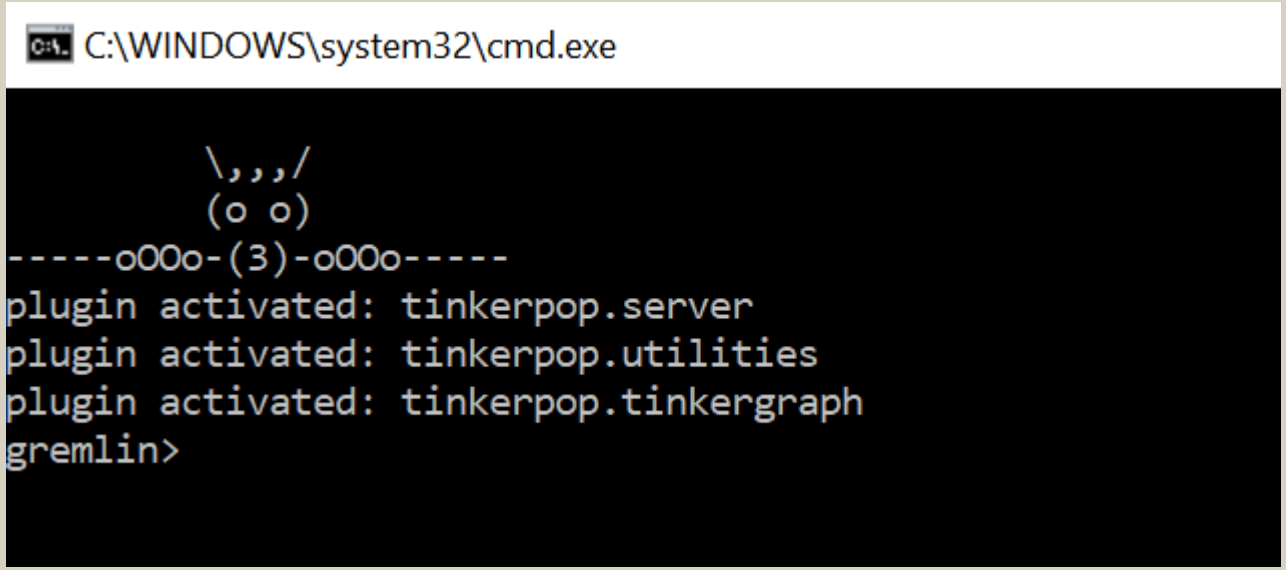
- Part of **TinkerPop** Apache Project
  - Graph traversal language & API
  - Console Client and Server
- Work with any **TinkerPop** client driver
  - Java, Python, Node.js, etc.
  - GraphSON and Gryo wire formats
- First-party .NET client SDK
- Direct connectivity to CosmosDB
  - Also supported in Titan, DSE Graph, Neo4j
- Gremlin ecosystem just works
  - Work with Apache Spark on HDInsight for graph analytics



Developing with Gremlin 101(Demo)

- Using any Gremlin driver
- Add a vertex
- Add an edge
- Filter by a property value
- Traverse the graph (join)
- More complicated examples
- From SQL to Gremlin





DEMO

Using GREMLIN API to access Azure Cosmos DB for Graph traversal

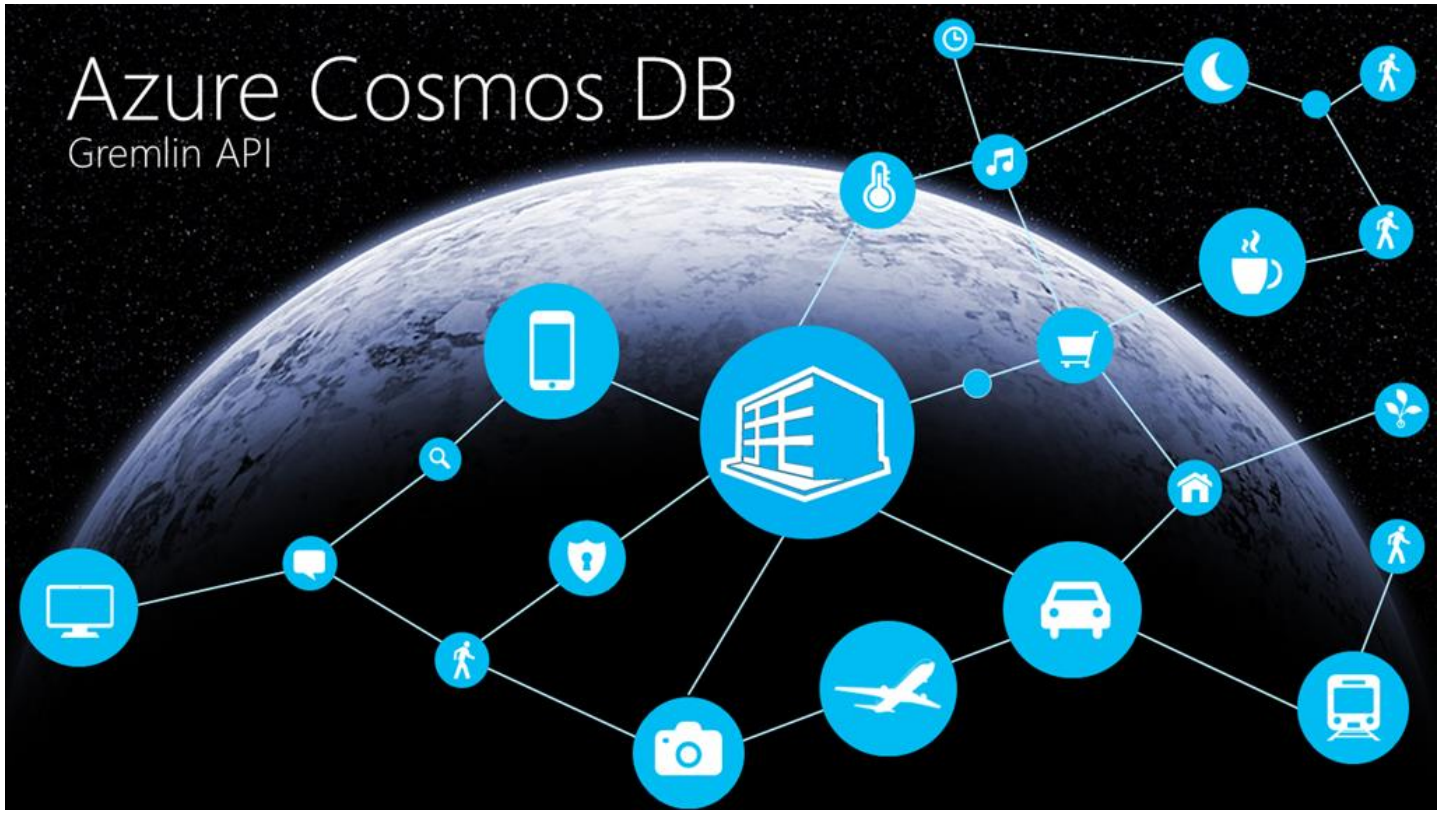
When? Very soon….

The Azure Cosmos DB Graph API (Gremlin) service will move out of preview and into general availability **by the end of the year 2017**

Try out Gremlin API today for free today with our Try Cosmos DB for free experience

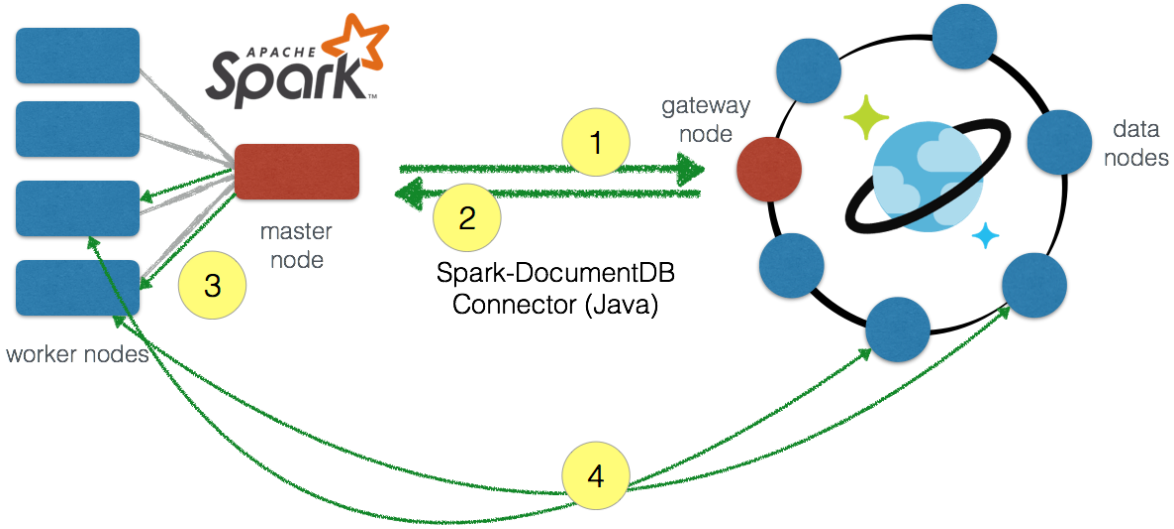
feedback on what you want to see next

Developer Forums on Stack Overflow



Comparison Recap

- **Neo4j**
  - Embedded into applications
  - Strong and deep Java dependencies
  - Graph traversal
  - Single node
- **SQL Server 2017** and **SQLDB**
  - Most of data is relational
  - Easy integration with other RDBMS features
    - The Machine Learning example
  - Different data workloads integration
- **Azure Cosmos DB**
  - Planet scale!
  - Multi-master and relaxed consistency models
  - Low latency and strong SLA
  - Stressing over Graph traversal
  - Integration: Apache Spark to Azure Cosmos DB Connector
  - Gremlin compat: compatibility with the open-source frameworks recommended by Apache TinkerPop



Q&A

Domande e risposte

Contatti

ROZZANO (MI) – BOLOGNA
ROMA – NAPOLI – GENOVA
TORINO

OverNet Education

Info@OverNetEducation.it

www.OverNetEducation.it

Rozzano (MI) +39 02 365738

Bologna +39 051 269911

www.wpc-overneteducation.it

APPENDIX

Session resources

SQL Server/DB Graph documentation: https://docs.microsoft.com/en-us/sql/relational-databases/graphs/sql-graph-overview

SQL Server 2017 Graph blog post: https://blogs.technet.microsoft.com/dataplatforminsider/2017/04/20/graph-data-processing-with-sql-server-2017/

Welcome to Azure Cosmos DB: https://docs.microsoft.com/en-us/azure/cosmos-db/introduction

DB-Engines Ranking of Graph DBMS: https://db-engines.com/en/ranking/graph+dbms

Azure Cosmos DB: Perform graph analytics by using Spark and Apache TinkerPop Gremlin: https://docs.microsoft.com/en-us/azure/cosmos-db/spark-connector-graph

Free Book Download O'Reilly's Graph Databases: https://neo4j.com/graph-databases-book/?ref=home

SAMPLE: Recommendation System 2 million nodes, 48.5 million edges)

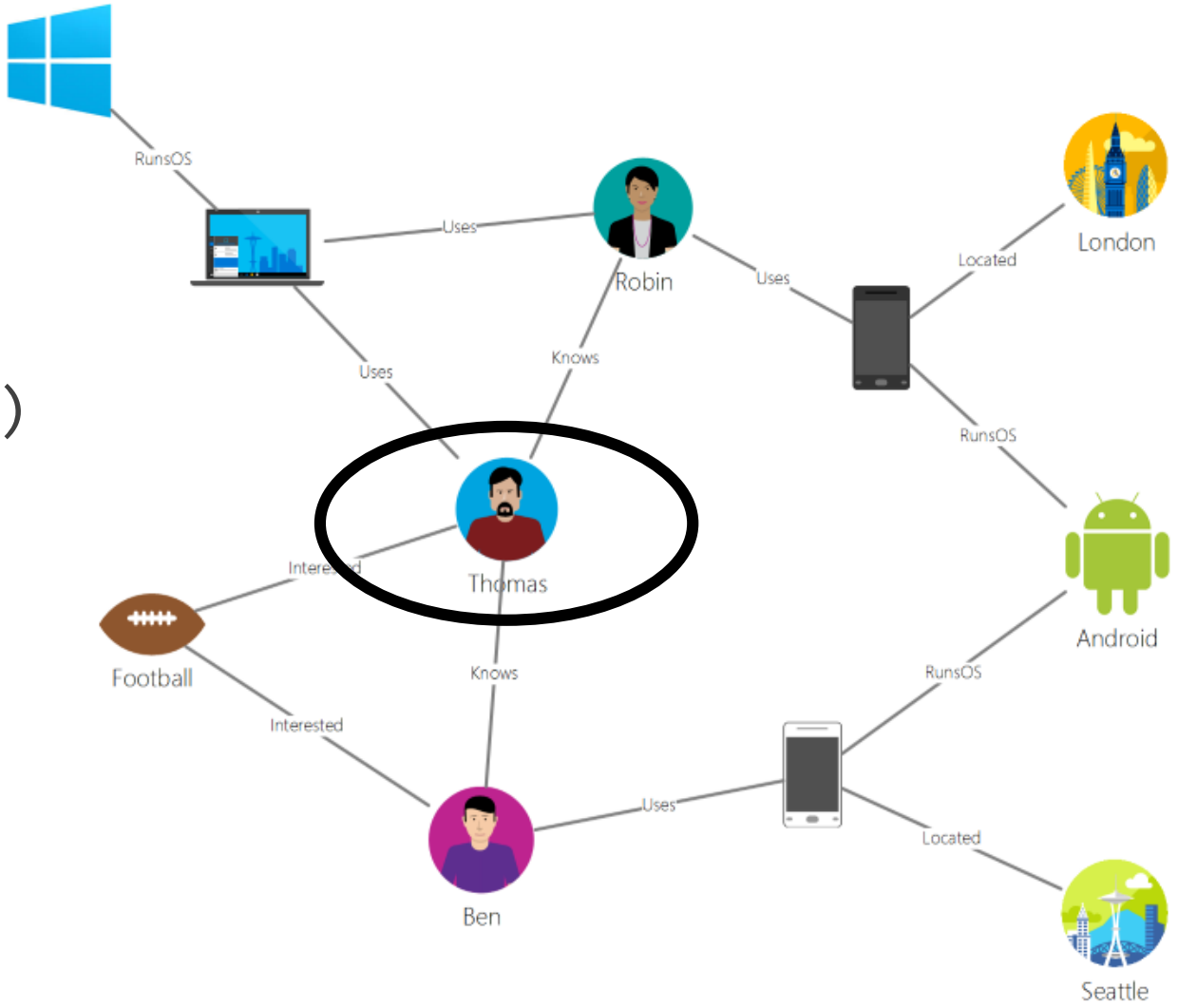
- https://blogs.msdn.microsoft.com/sqlcat/2017/04/21/build-a-recommendation-system-with-the-support-for-graph-data-in-sql-server-2017-and-azure-sql-db/
- https://github.com/arvindshmicrosoft/MillionSongDatasetinSQLServer

SAMPLE: A more comprehensive end-to-end scenario which features Graph Data as one of the building blocks

- https://github.com/Microsoft/sql-server-samples/blob/master/samples/applications/iot-connected-car

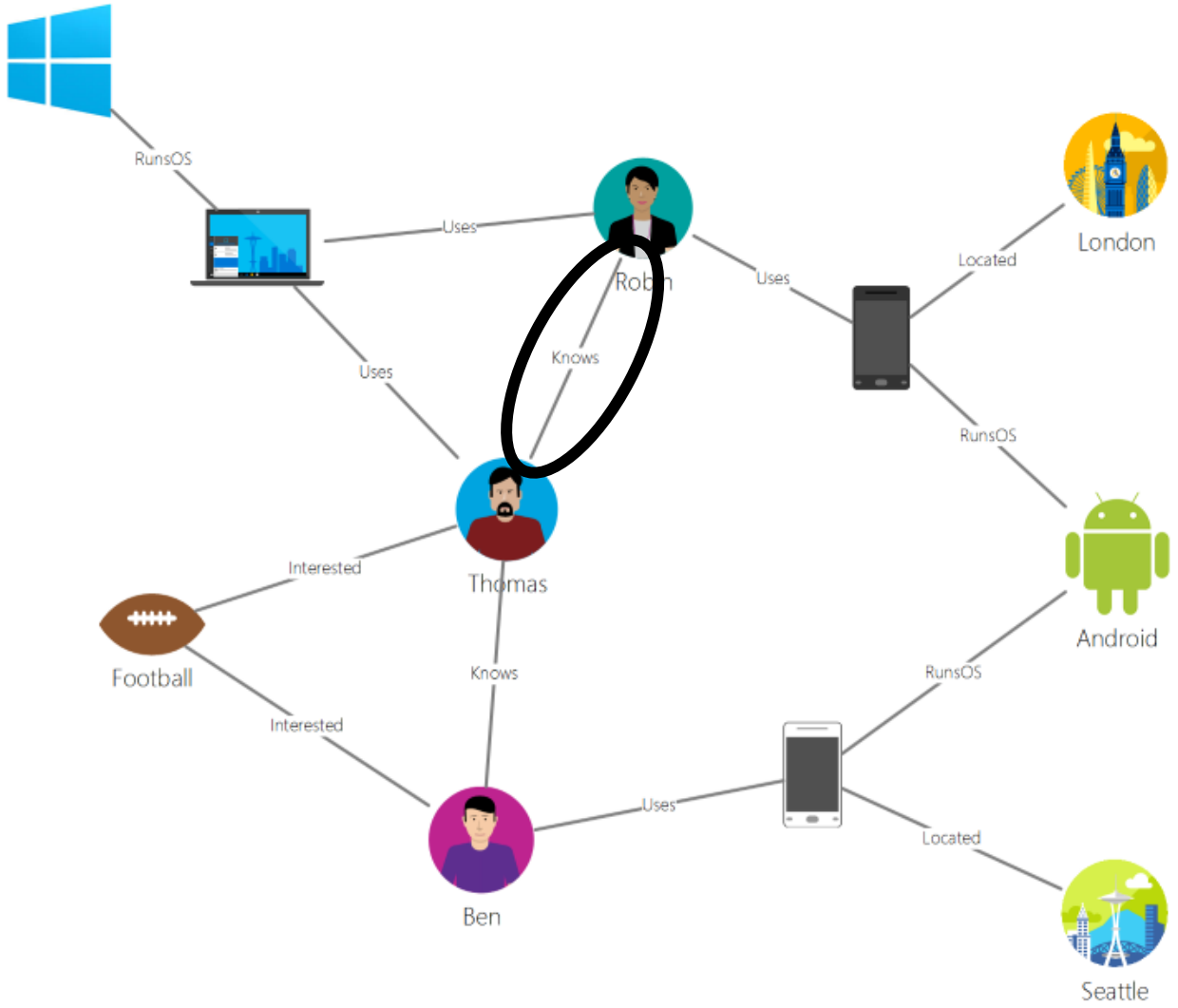
Add vertex

```
g.addV('person')
   .property('id', 'Thomas')
   .property('firstName', 'Thomas')
   .property('lastName', 'Andersen')
   .property('age', 44)
   .property('userid', 1)
```



Add edge

```
g.V('Thomas')
   .addE('knows')
   .property('since', '2017')
   .to(g.V('Robin'))
```

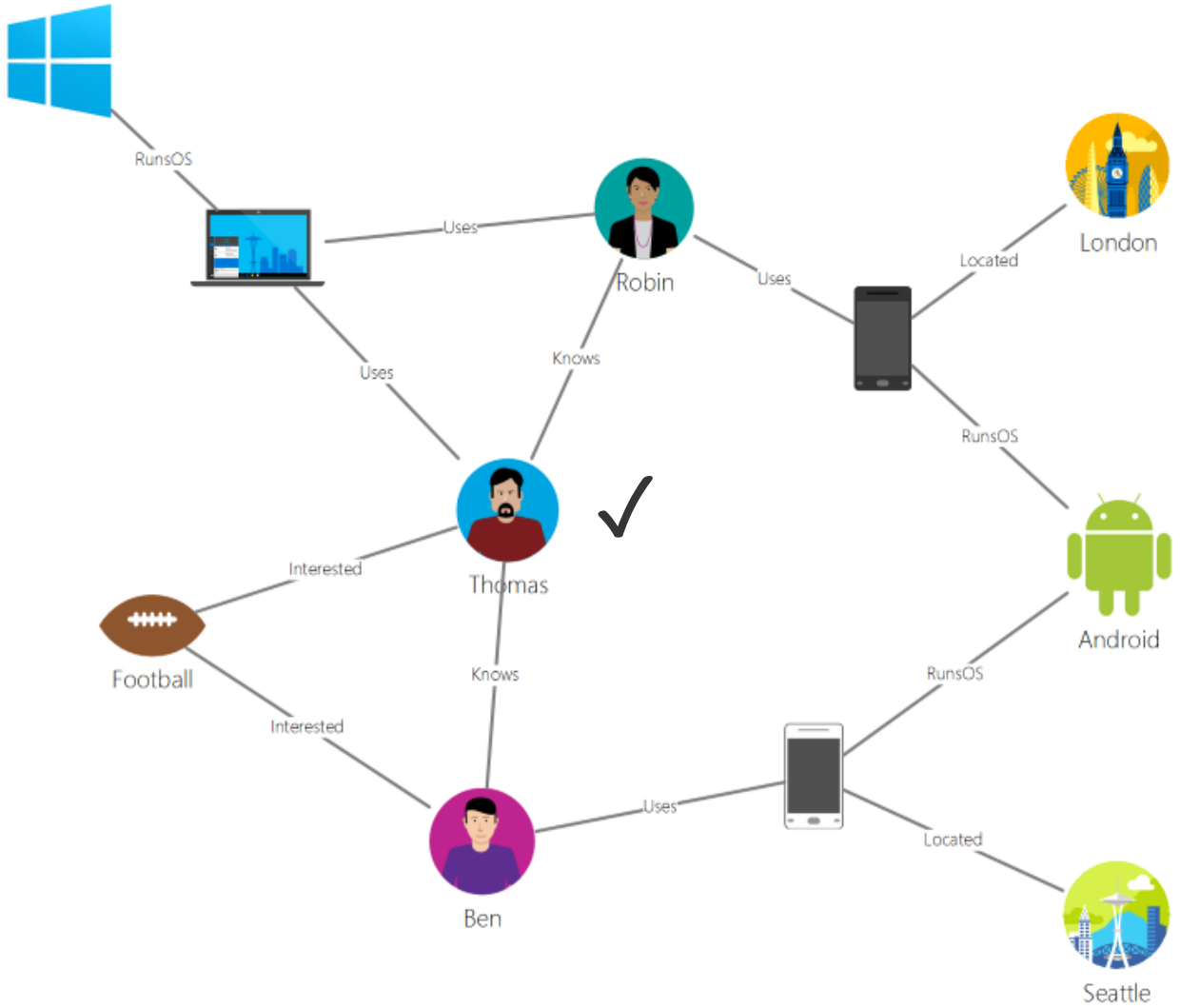


Gremlin
$G = (V, E)$

Filter

```
g.V()
   .hasLabel('person')
   .has('age', gt(40))
```

✓ Automatic index on /label/person
✓ Automatic index on /age/[>40]

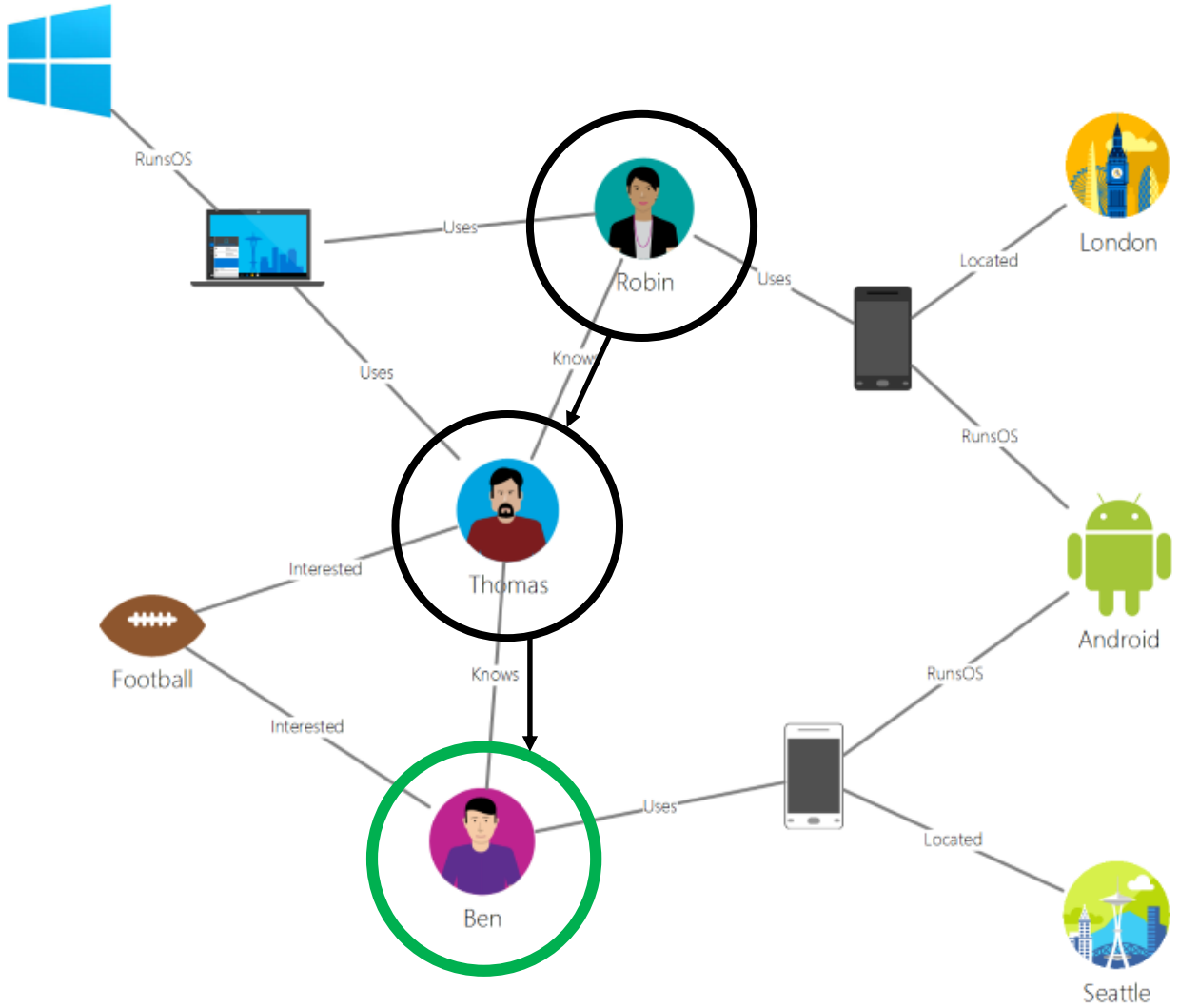


Traverse & Closure

```
g.V('robin')
   .out('knows')
   .hasLabel('person')
```

```
g.V('robin')
   .out('knows')
   .hasLabel('person')
   .out('knows')
   .hasLabel('person')
```

```
g.V('robin')
   .repeat(out())
   .until(has('id', 'ben'))
   .path()
```



From SQL to Gremlin

Projection

```
g.V().hasLabel('person')
  .values('firstName')
```

Sort

```
g.V().hasLabel('person')
  .order().by('firstName', decr)
```

Filter

```
g.V().hasLabel('person')
  .has('age', gt(40))
```

Aggregate

```
g.V().hasLabel('person').count()
```

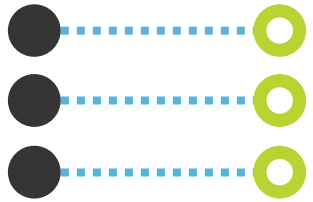
Join

```
g.V('robin').out('knows').hasLabel('person')
  .out('knows').hasLabel('person')
```

Graph data format (multi-model)

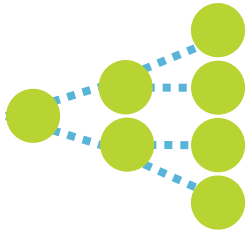
Who wants to have 4 different databases?

- Database engine operates on atom-record-sequence (ARS) based type system
- All data models are efficiently translated to ARS
- API and wire protocols are supported via extensible modules
- Instance of a given data model can be materialized as trees

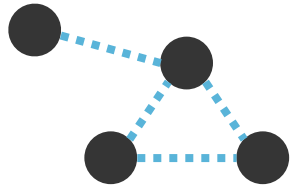


KEY-VALUE

COLUMN-FAMILY



DOCUMENT



GRAPH

Graph compute tier

- Two-tier architecture
- Independently scalable graph compute tier for graph
- Data tier with low latency, automatic indexing, global distribution, etc.