



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David



DBAAS AND NOSQL

Week:
5-Dec-2023

DBaaS and NoSQL: Database Systems (DBS)
Dr Suresh Shanmugasundaram

So far...



1. Transactions
2. Concurrency Control
3. Recovery
4. Indexing
5. Optimization
6. Security
7. Distributed Databases

Learning Objectives



1. DBaaS
2. Big Data
3. NoSQL Databases
4. Types of NoSQL Databases
5. Graph & Document Databases

DBaaS - Database as a Service

- ❑ Cloud database run on a cloud computing platform.
- ❑ Access to a Database is provided as a service.
- ❑ Users don't have to install and maintain the DB's on their systems.
- ❑ The providers of this service will install, maintain, update, and upgrade the DB's and the users will be charged for this service.
- ❑ These DB's are run on VM (Virtual Machines) and if the VM's are provided by the service provider then it will be optimized for DB installations.
- ❑ The users can also upload their own VM.
- ❑ Both Relational and NoSQL DB's are available as a service.

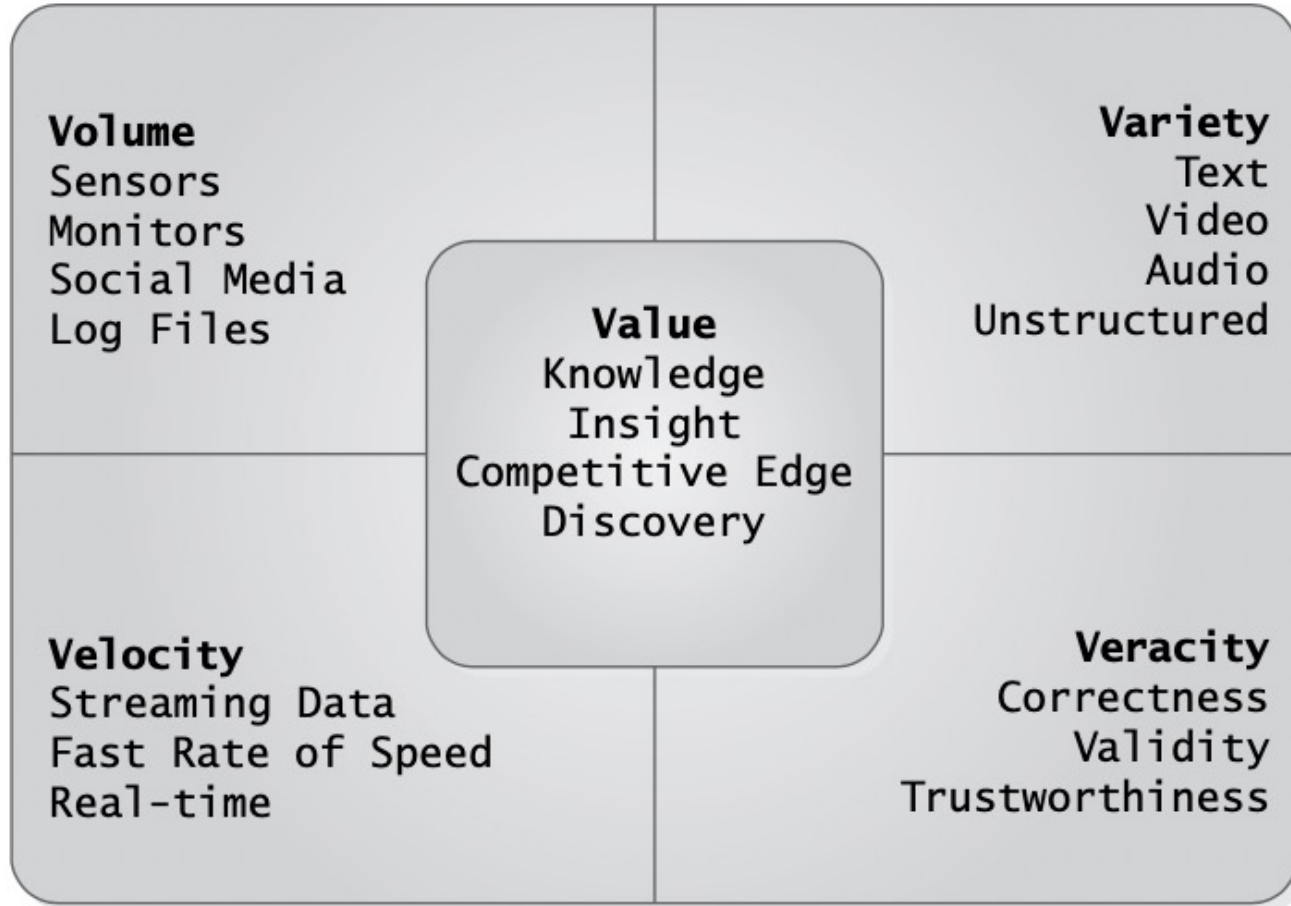
Cloud DB's

- ❑ **SQL DB's:** Examples: Oracle Database Cloud Service, Google Cloud SQL, Microsoft Azure SQL DB, Amazon Relational DB service etc. They scale vertically very easily. Horizontal Scalability is tricky which some of these have started to address.
- ❑ **NoSQL DB's:** Examples: Amazon DynamoDB, Google Cloud BigTable/Datastore, Amazon DocumentDB, Oracle NoSQL DB Cloud Service, Azure Cosmos DB etc. They can deal with heavy read/write operations and scale horizontally and vertically very well.

Internet Age and Big Data Challenges

- Data comes in large volumes.
- Data comes in at faster rate.
- Performance decreases exponentially.
 - Organizations forced to invest in high end servers
 - Increase network capacity
- However, there was a limit beyond which vertical scaling did not help... too expensive!

5 Vs of Big Data



Optimization Attempt

- Standard techniques
 - ▣ Indexes half of the databases
 - Made those queries a little faster
 - ▣ Materialized views for complex joins
 - Nightmare to maintain, get stale, etc...
 - ▣ De-normalizes
 - Any thing but a smooth transition!
 - Redundancy
 - ▣ Introduce caching
 - Data too stale
 - More redundancy

Top Three Drivers for NoSQL

- ▣ **Rigid schemas**

- Not suitable for semi structured, unstructured data

- ▣ Designed **to scale up** not scale out across clusters

- Need high end machines calling for heavy investment in the infrastructure

- ▣ **Performance** degrades with larger volumes of data

Basic reason for the NoSQL were the challenges of handling exponential growth in volume of data and types of data.

Introduction to NoSQL – Contd...

- New category of databases. It is another file management system as were Flat file systems, RDBMS, OLAP.
- Concept of NoSQL grew with internet giants such as the Google, Yahoo and Amazon who deal with gigantic volumes of data.
 - RDBMS do not scale with massive volumes of data.
 - There is a limit to vertical growth of the underlying servers.
 - Distribution of RDBMS on multiple hosts was not possible on scales required by the volume of data (CAP theorem).
- NoSQL main characteristics is it's non-adherence to relational database concepts of CODD.
- NoSQL means, not only SQL-Have their own query language.

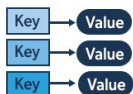
Introduction to NoSQL – Contd...(2)

- NoSQL scale out much better than relational databases.
- They do not follow strict schema.
- ACID properties are not always guaranteed.
- Main focus of NoSQL databases include:
 - Scalability (handle large amounts of data without performance degradation)
 - Performance (consistently perform within expected values irrespective of volume)
 - High availability (Should be fault tolerant)
 - ▣ Scalability and performance of NoSQL come at cost of functionalities
 - Limited functionalities compared to RDBMS
 - Do not support joins, no complex transactions, no support for constraints (legal values for a column)
 - Transactions and constraints have to be handled by the applications

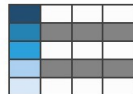
Applicability of NoSQL

- Need to store and retrieve huge volumes of data and retrieve large sets of data for processing (usually in petabytes and above).
- Data does not have consistent structure, is semi-structured or unstructured such as image data, video files etc.
- No need to maintain relation between structures. Relations are addressed at application level.
- Availability of data is more important than data consistency.

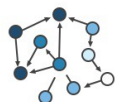
Key-Value



Column-Family



Graph



Document

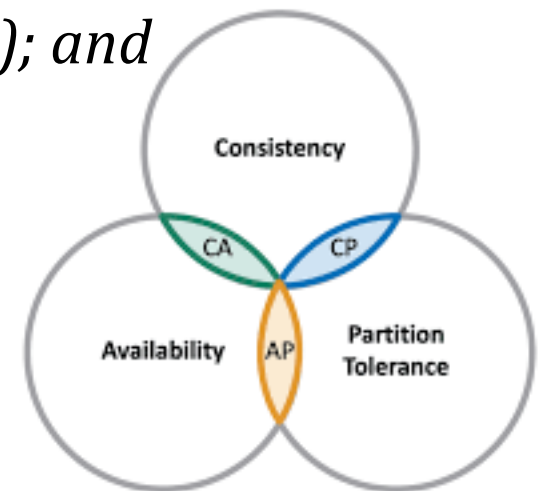


Seeds of NoSQL

- Three major papers were the “seeds” of the NOSQL movement:
 - CAP Theorem
 - BigTable (Google)
 - DynamoDB (Amazon)

The CAP Theorem

- Eric Brewer on distributed systems:
- *The CAP theorem states that any networked shared-data*
- *system can have at most **two of three** desirable properties:*
 - **consistency** (*C*) equivalent to having a single up-to-date copy of the data;
 - **high availability** (*A*) of that data (for updates); and
 - **tolerance to network partitions** (*P*).



What CAP Means

- Consistency means that data is the same across the cluster, so you can read or write to/from any node and get the same data.
- Availability means the ability to access the cluster even if a node in the cluster goes down.
- Partition Tolerance means that the cluster continues to function even if there is a "partition" (communications break) between two nodes (both nodes are up, but can't communicate).

Eventual Consistency

- This is not so bad
 - ▣ It means that we just settled for a lesser degree of Consistency
 - ▣ Business entities must deal with the challenges of inconsistencies with appropriate business policies
- Eventual Consistency as Opposed to Immediate Consistency.

The Compromise

- We settle for weaker consistency model
 - ▣ BASE
 - Basically Available
 - Soft state
 - Eventual Consistency
- ACID on the individual node BASE on the cluster.

What is NOSQL?

- ❑ Not only SQL instead of No SQL.
- ❑ Key features (advantages):
 - ▣ non-relational
 - ▣ don't require schema
 - ▣ data are replicated to multiple nodes (so, identical & fault-tolerant) and can be partitioned:
 - down nodes easily replaced
 - no single point of failure
 - ▣ horizontal scalable
 - ▣ cheap, easy to implement (open-source)
 - ▣ massive write performance
 - ▣ fast key-value access



NOSQL categories

1. **Key-value**

- Example: DynamoDB, Voldermort, Scalaris

2. **Document-based**

- Example: MongoDB, CouchDB

3. **Column-based**

- Example: BigTable, Cassandra, Hbase

4. **Graph-based**

- Example: Neo4J, InfoGrid

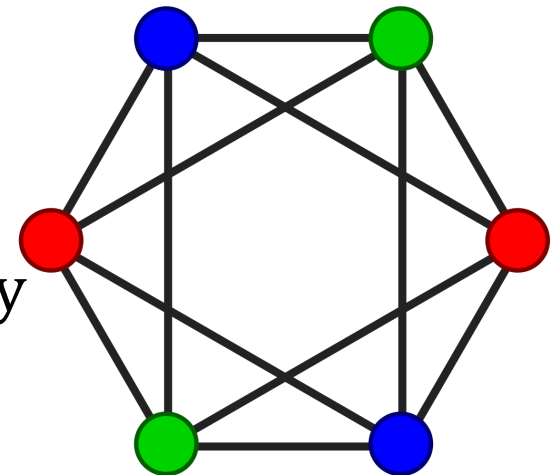
- “No-schema” is a common characteristics of most NOSQL storage systems.
- Provide “flexible” data types.



Graph Databases

Graph Theory Applied

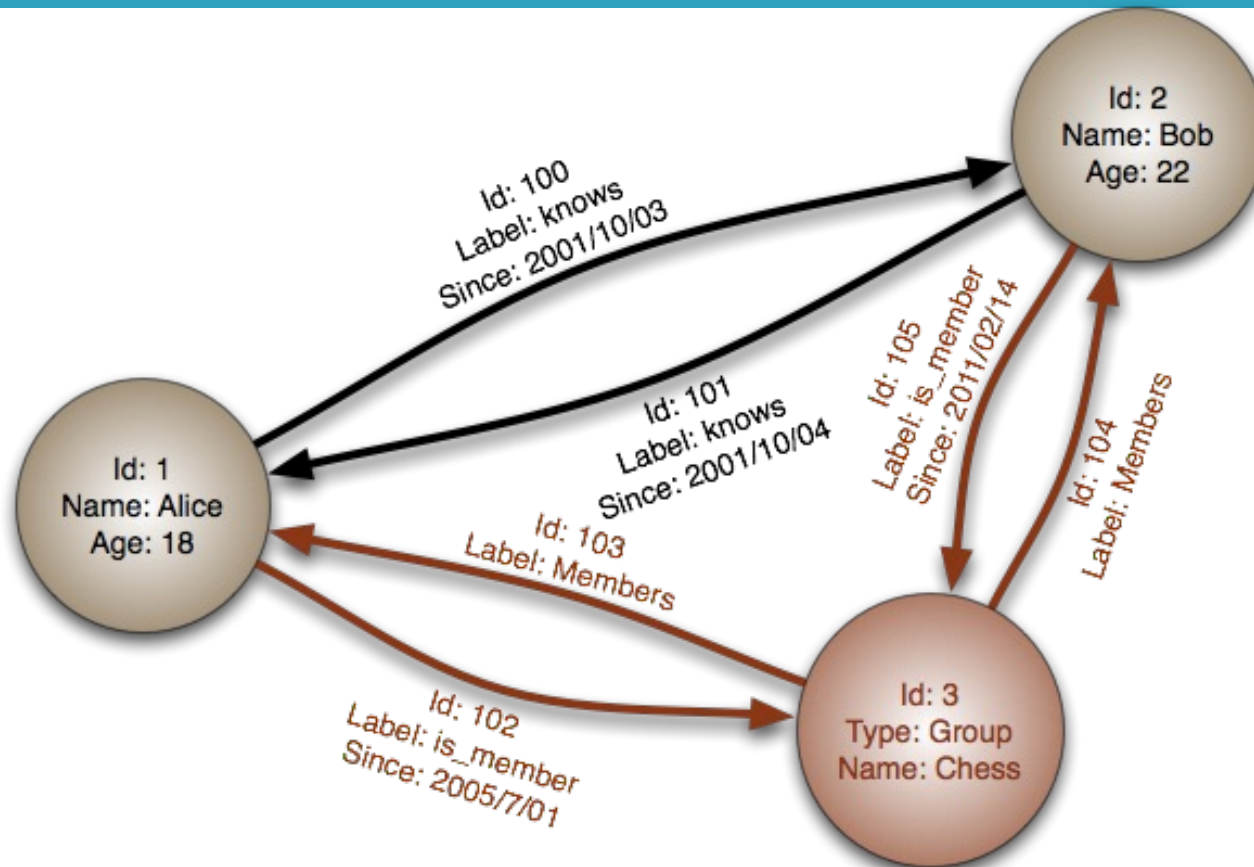
- Graph Theory: a branch of Mathematics used to model the connections between objects.
- Became popular with the wide use of Internet as data in Web 2.0 is highly interconnected.
- Also used in Biology (DNA), Physics, Chemistry, and Economics to model interconnections.
- With BigData, too many many-to-many relationships needed to be modelled; Graph Theory comes to the rescue in the form of Graph Database.



Graph Database – Introduction

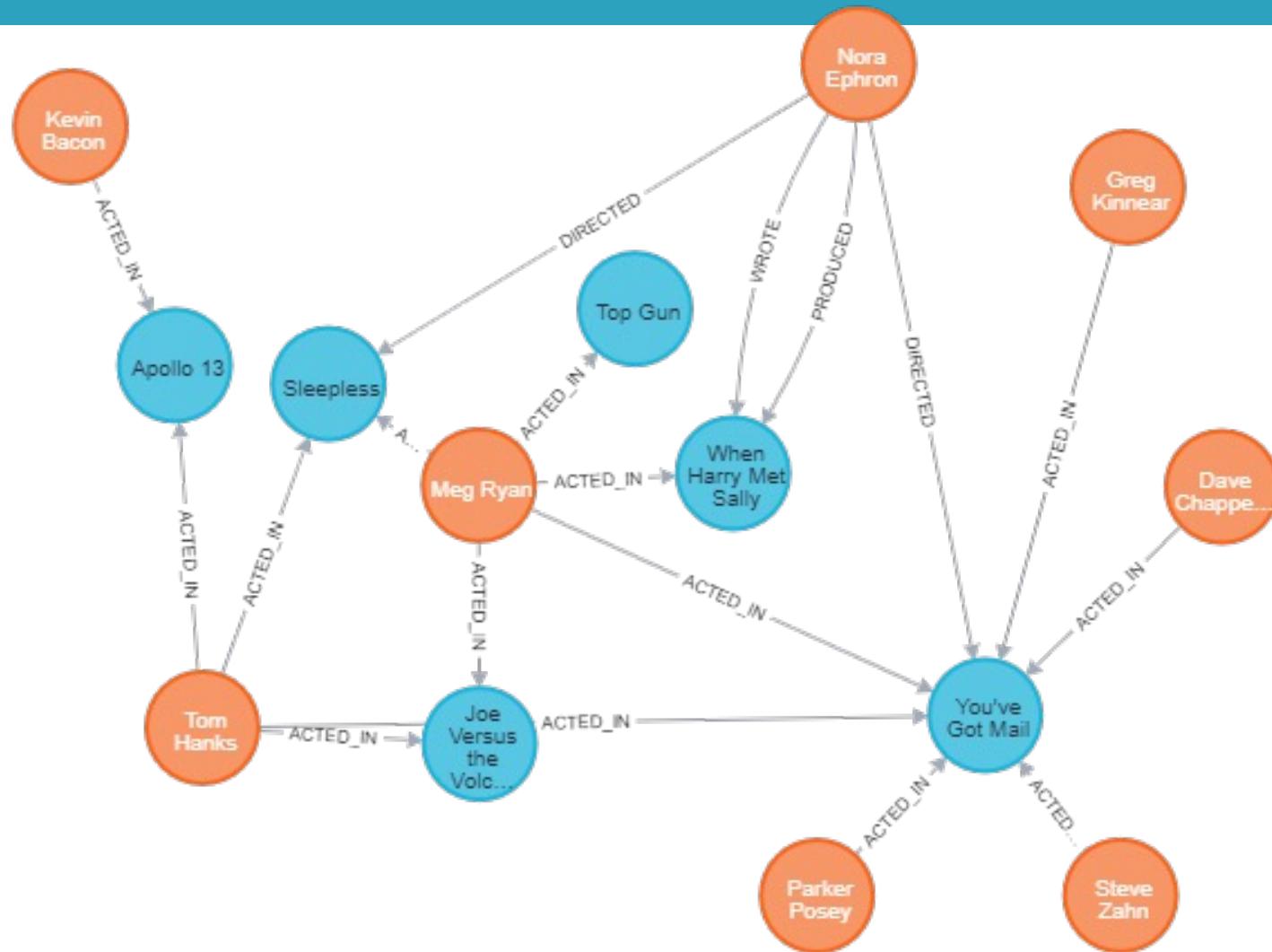
- ❑ Graph Data Model consists of **Nodes** and **Edges**, where the *nodes* represent an *entity* and the *edges* represent the *relationship* between two entities.
- ❑ Graph Databases built on **Graph Data Model**.
- ❑ Efficiently manages **many-to-many, property-laden relationships** that co-exist in a highly dynamic environment.
- ❑ ideal way to represent and analyse **complex nonlinear networks** e.g. Twitter/Facebook data and connections.
- ❑ Drawback suggested: inefficient with end-to-end transaction processing and rapid associative summarization.

Graph Database – Introduction



By Originally uploaded by Ahzf (Transferred by Obersachse) - Originally uploaded on en.wikipedia, CC0, <https://commons.wikimedia.org/w/index.php?curid=19279472>

Graph Database – Movie Database (Neo4j)



Graph Database – Key Aspects

- ❑ **Relationships** are the key to the Graph Data Modelling.
- ❑ **Connected Data** is as important as the individual data points.
- ❑ Uses Graph Theory's **traversal algorithms** to retrieve data using the GraphQL (Graph Query Language).
- ❑ Various types of Graph Data Models:
 - a. **RDF** (Resource Description Framework) **Triples**
 - b. **Property Graphs** (used in Neo4j)
 - c. **Hypergraphs**

RDF Triples



Figure 1: An RDF Graph Describing Eric Miller

<https://www.w3.org/TR/2004/REC-rdf-primer-20040210>

- RDF is a collection of triples (**subject, predicate, and object**).
- A set of such triples is called **RDF Graph**.
- **Subject** and **Object** represented by **nodes**.
- **Predicate** represented by **edges**.

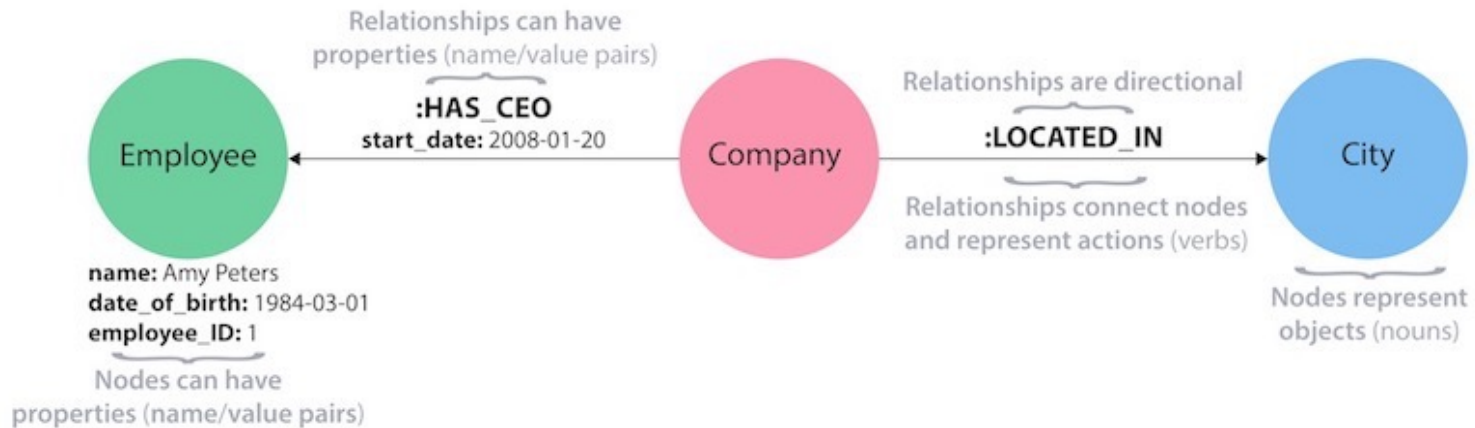
RDF Triples – More ...

- ❑ Triple stores are modelled around RDF, maintained by W3C.
- ❑ RDF data model is used for data interchange on the Web (Semantic Web).
- ❑ Query Languages used is **SPARQL**.
- ❑ Used in P3P and **Creative Commons**.
- ❑ Useful in building **knowledge graphs**.
- ❑ Allows for effective integration of data from different sources.

Property Graphs

- Contains Nodes (data entities) and Edges (relationships)
- **Nodes and Edges can have properties** (key-value pairs).
- Nodes can have 1 or more labels.
- Relationships can have names and directions.
- Relationships can have a start and end node.
- **Relationships can also have properties.**
- E.g. Neo4j

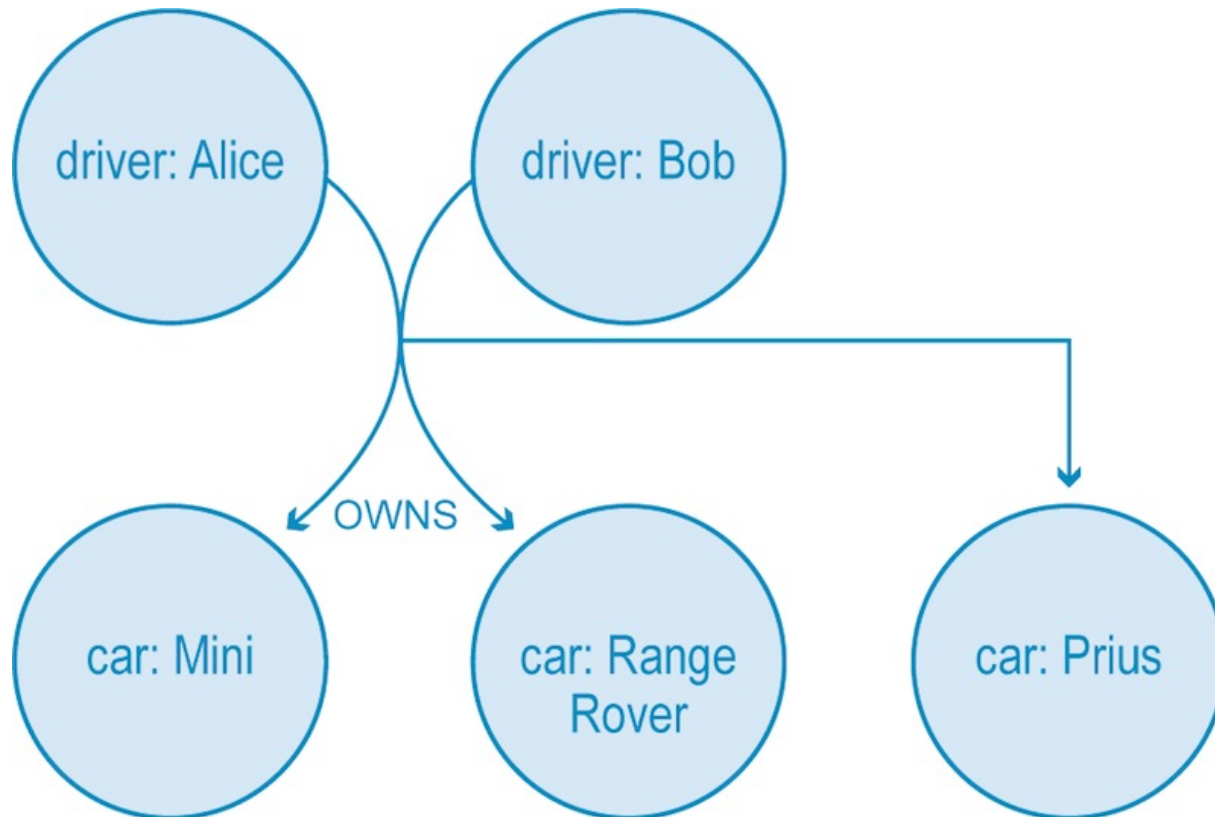
Property Graph - Example



Hypergraphs

- A Graph in which an edge can join any number of vertices (as opposed to only two in an ordinary graph).
- Labelled hypergraphs are used to represent data on stateful machines.
- Highly useful to represent a large number of many-to-many relationships.
- They are multidimensional and more generalized.
- Theoretically they are said to produce accurate, information-rich data models.
- Each node (known as a **hypernode**) can consist of one or more nodes or hypergraphs and each edge (known as a **hyperedge**) or link can consist of one or more edges.

Hypergraphs – Example



□ <https://neo4j.com/blog/other-graph-database-technologies/?ref=blog>

Hypergraphs – More...

- ❑ Groups relevant pieces of information in a relational style.
- ❑ Uniform handling of all n-ary relationships.
- ❑ Ability to express higher-order information like relationship between the relations, nesting of information etc.
- ❑ Because of the effective group of information, data retrieval can be planned and executed optimally.

Property Graphs – More...

- Possible to uniquely identify instances of a relationship unlike in RDF.
- Quantify instances in a relationship by giving them attributes.
- Easy to navigate.
- Uses **Cypher** as QL.
- Implements **index-free adjacency** between nodes in disks.
- Works very well in highly dynamic environments and transactional use cases where data integrity is key.

Graph Databases – Applications



- Social networks
- Transportation networks
- Biological pathways
- Citation networks

A horizontal bar at the top of the slide, divided into a red section on the left and a blue section on the right.

Document Databases

Document Databases

- ❑ Also called as **document store** or **document oriented database**.
- ❑ Flexible, semi-structured, and can easily evolve as and when the applications change.
- ❑ Used in content management applications (blogs, video streaming platforms etc.), storing catalog information, etc.
- ❑ Flexible indexing (associates keys with documents), analytics over document collections, and efficient ad hoc queries.

Document Databases – Key Aspects

- ❑ **Collections** similar to tables in RDBMS.
- ❑ **Documents** similar to rows.
- ❑ Documents are usually JSON/XML objects (MongoDB uses a binary encoded version of JSON called BSON).
- ❑ Each document has properties and values (which can be scalars, arrays, links to documents in other databases or sub-documents).
- ❑ Documents can have attachments as well.
- ❑ Each document is identified by a key which is used to create an index for quick retrieval.
- ❑ **Popular document databases are Amazon DocumentDB, MongoDB, CouchDB etc.**

Document Database – Example (MongoDB)

Relational

ID	first_name	last_name	cell	city	year_of_birth	location_x	location_y
1	'Mary'	'Jones'	'516-555-2048'	'Long Island'	1986	'-73.9876'	'40.7574'

ID	user_id	profession
10	1	'Developer'
11	1	'Engineer'

ID	user_id	name	version
20	1	'MyApp'	1.0.4
21	1	'DocFinder'	2.5.7

ID	user_id	make	year
30	1	'Bentley'	1973
31	1	'Rolls Royce'	1965

MongoDB

```
{
  first_name: "Mary",
  last_name: "Jones",
  cell: "516-555-2048",
  city: "Long Island",
  year_of_birth: 1986,
  location: {
    type: "Point",
    coordinates: [-73.9876, 40.7574]
  },
  profession: ["Developer", "Engineer"],
  apps: [
    { name: "MyApp",
      version: 1.0.4 },
    { name: "DocFinder",
      version: 2.5.7 }
  ],
  cars: [
    { make: "Bentley",
      year: 1973 },
    { make: "Rolls Royce",
      year: 1965 }
  ]
}
```

Document Databases – More...

- ❑ Easy to scale horizontally to accommodate large data volume.
- ❑ Scales very well with both structured and semi-structured data.
- ❑ Schemas can be modified easily without any downtime and promoting high availability.
- ❑ Not ideal for applications with complex search queries or with applications requiring complex multiple operation transactions.



End of Story!

Thank you for your engagement !
All the best for your End
Assessments!