JAIN DEEMED-TO-BE UNIVERSITY | SCHOOL OF COMPUTER SCIENCE AND IT

Master of Computer Applications

18MCA301 – **NoSQL Databases**

Module -5

Graph Based Databases

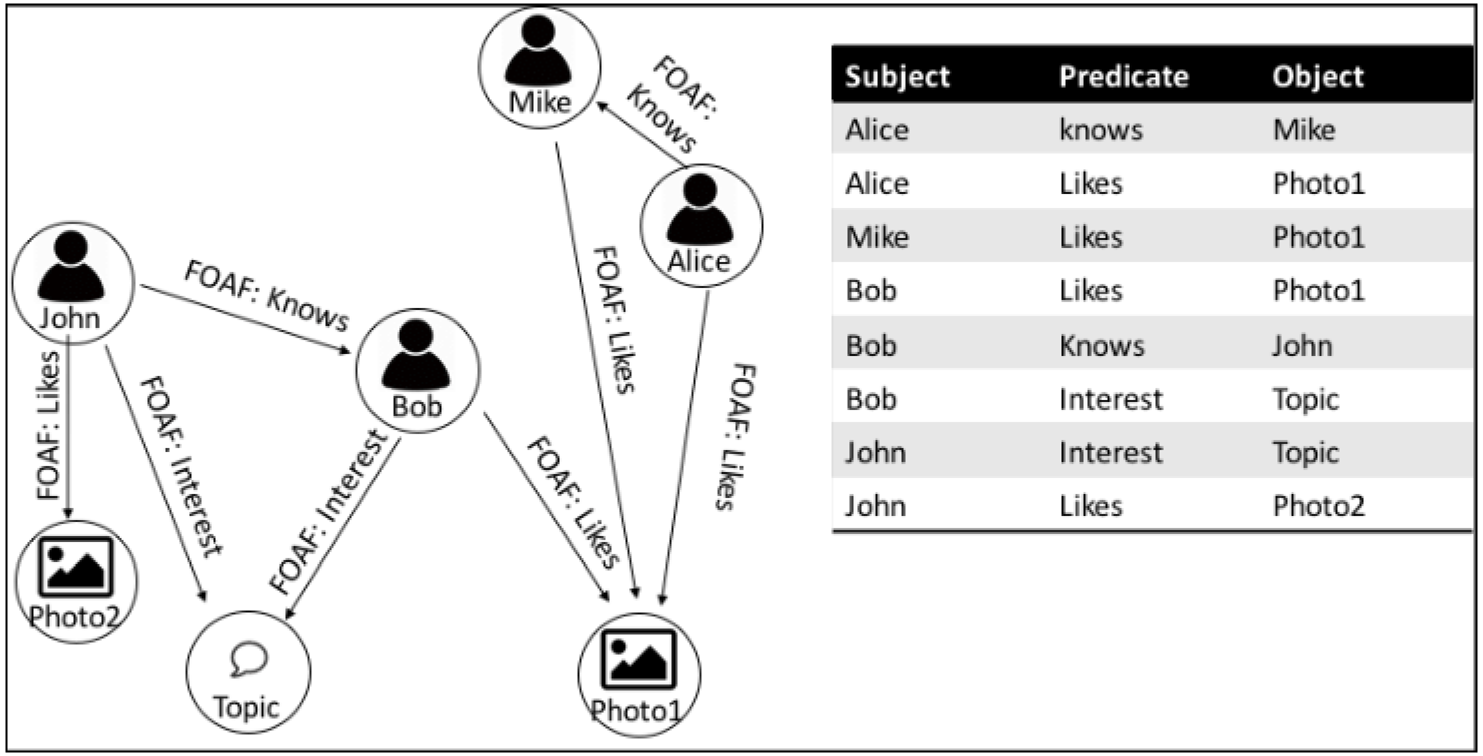
III General

Dr Gangothri

October 2020

Graph and Triple Store

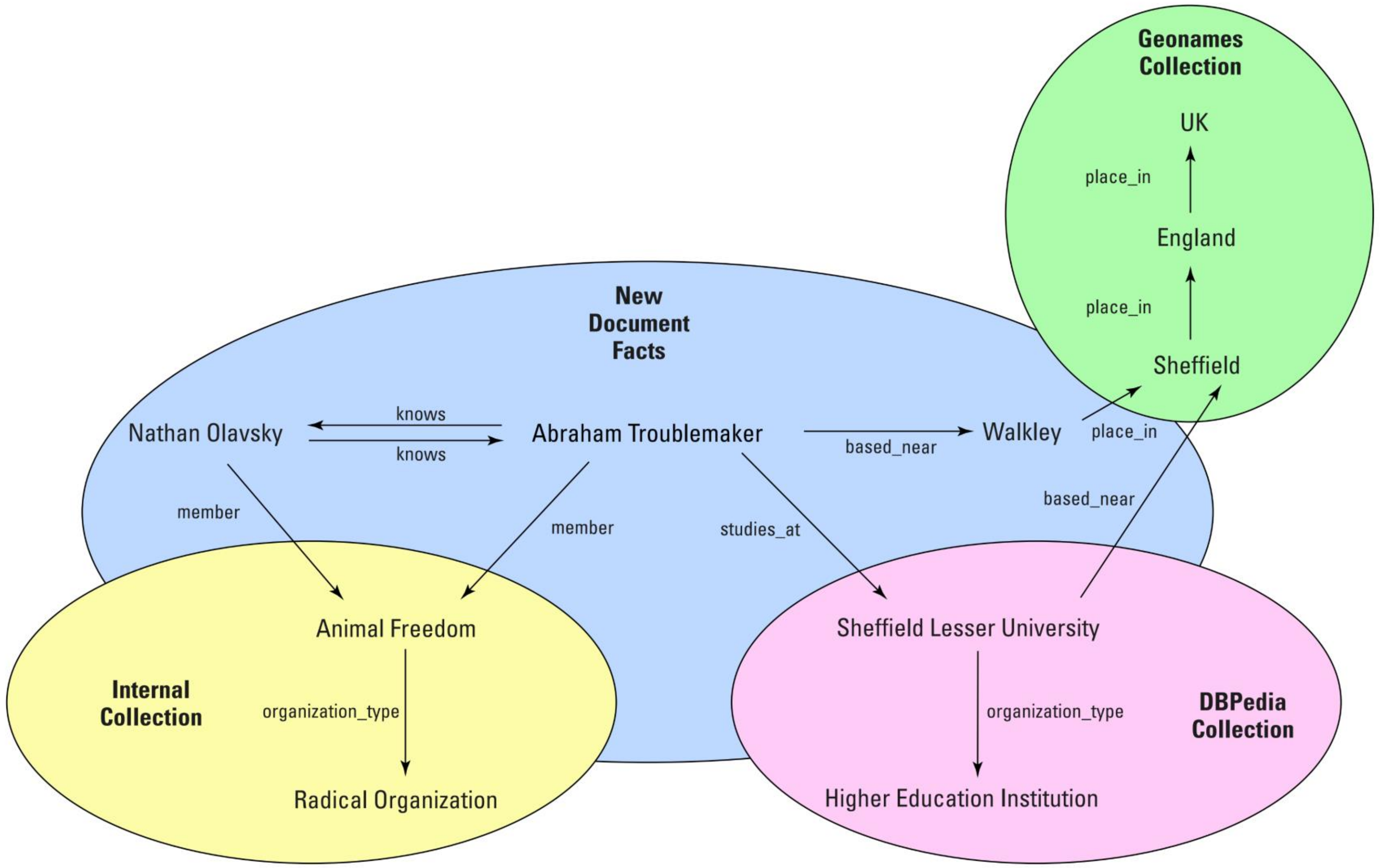
- Do you really need a web of interconnected data, or can you simply tag your data and infer relationships according to the records that share the same tags?
- If you do have a complex set of interconnected data, then you need to decide what query functionality you need to support your application.
- In order to get the facts, or assertions, that you require, are you manually adding them, importing them from another system, or determining them through logical rules, called inferencing?



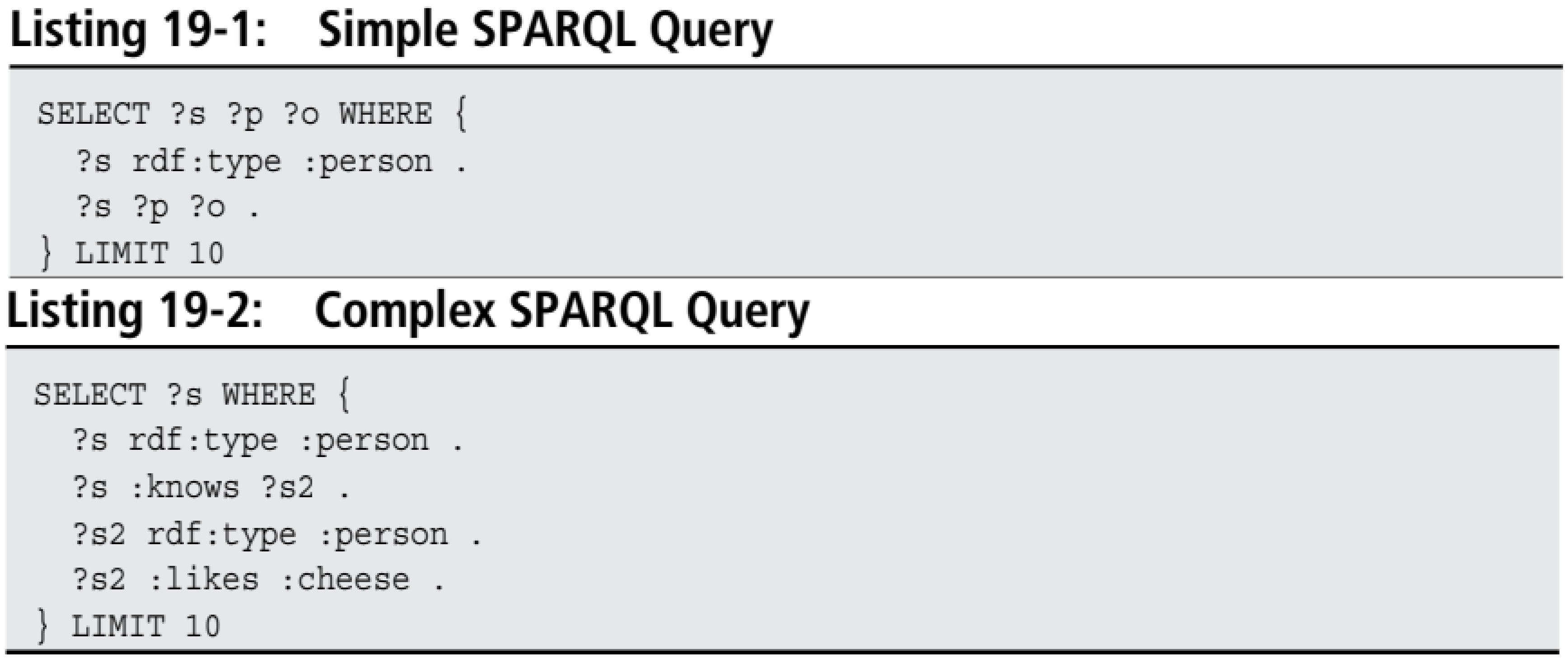
Subject	Predicate	Object
Alice	knows	Mike
Alice	Likes	Photo1
Mike	Likes	Photo1
Bob	Likes	Photo1
Bob	Knows	John
Bob	Interest	Topic
John	Interest	Topic
John	Likes	Photo2

Triple queries

- A triple store managed individual assertions.
- For most use cases you can simply think of an assertion as a "fact."
- These assertions describe subjects' properties and the relationships between subjects.
- The data model consists of many simple subject – predicate – object triples.
- This subject – predicate – object triple allows complex webs of assertions, called graphs, to be built up.
- One triple could describe the type of the subject, another an integer property belonging to it, and another a relationship to another subject. each relationship and type is described using a particular vocabulary.
- Each vocabulary is called an ontology.
- These ontologies are described using the same triple data model in documents composed of Resource Description Framework (RDF) statements.
- In graph theory these subjects are called vertices, and each relationship is called an edge.
- In a graph, both vertices and edges can have properties describing them.


Geonames Collection
UK
place_in
England
place_in
Sheffield
New Document Facts
Nathan Olavsky
knows
knows
Abraham Troublemaker
based_near
Walkley
place_in
based_near
member
member
studies_at
Animal Freedom
Internal Collection
organization_type
Radical Organization
Sheffield Lesser University
organization_type
DBPedia Collection
Higher Education Institution

- Every graph store is a triple store because both share the same concepts. However, not every triple store is a graph store because of the queries that each can process. A triple store typically answers queries for facts.



Listing 19-1: Simple SPARQL Query

```
SELECT ?s ?p ?o WHERE {
  ?s rdf:type :person .
  ?s ?p ?o .
} LIMIT 10
```

Listing 19-2: Complex SPARQL Query

```
SELECT ?s WHERE {
  ?s rdf:type :person .
  ?s :knows ?s2 .
  ?s2 rdf:type :person .
  ?s2 :likes :cheese .
} LIMIT 10
```

- These examples SPARQL queries share one thing in common: They return a list of triples as the result of the operation. They are queries for data itself, not queries about the state of the relationships between subjects, the size of a graph, or the degree of separation between subjects in a graph.

Graph queries

- A graph store provides the ability to discover information about the relationships or a network of relationships.
- Graph stores can respond to queries for data, too, and are also concerned with the mathematical relationships between vertices.
- ***Shortest path***: Finds the minimum number of hops between two vertices and the route taken.
- ***All paths***: Finds all routes between two vertices.
- ***Closeness:*** Given a set of vertices, returns how closely they match within a graph.
- ***Betweenness:*** Given a set of vertices, returns how far apart they are within a graph.
- Subgraph: Either finds a part of the graph that satisfies the given constraints or returns whether a named graph contains a specified partial graph.
- These algorithms are mathematically much harder queries to satisfy than simply returning a set of facts.
- This is because these algorithms could traverse the graph to an unpredictable depth of search from the first object in the database.
- Triple queries, on the other hand, are always bounded by a depth within their queries and operate on a known set of vertices as specified in the queries themselves.

Deciding on Triples or Quads

- The subject – predicate – object data model is a very flexible one.
- It allows you to describe individual assertions.
- There are situations though when the subject – predicate – object model is too simple, typically because your assertion makes sense only in a particular context.
- Triple stores have the concept of a named graph.
- Rather than simply add all your assertions globally, you add them to a named part of the graph.
- You can use this graph name to restrict queries to a particular subset of the information.
- In this way, you don’t need to change the underlying ontology or the data model used in order to support the concept of context.
- Note that each triple can be stored in only a single named graph.
- This means you must to carefully select what you use as your context and graph name.
- If you don’t, then you may find yourself in a situation where you need to use two contexts for a single set of triples.

Storing RDF

- The first standard you need to become familiar with when dealing with triples is the Resource Description Framework (RDF).
- This standard describes the components of the RDF data model, which includes subjects, predicates, objects, and how they are described.
- Key RDF concepts:

1. ***URI:*** The unique identifier of a subject or a predicate.
2. ***Namespace:*** A namespace allows packaging up of objects in an ontology. Namespaces allow you to mix internal RDF constructs and third-party ontologies.
3. ***RDF type:*** Used to assert that a subject is an instantiation of a particular type. Not equivalent to a class. RDF supports inheritance between RDF types, including across ontologies.
4. ***Subject:*** The entity you're describing — for example, physical (person, place) or conceptual (meeting, event). Takes the form of a URI.
5. ***Predicate:*** The edge, or relationship, between the subject and the object. Takes the form of a URI.
6. ***Object:*** Either another subject URI when describing relationships between subjects, or an intrinsic property value like an integer age or string name.

- A key difference between RDF and other specifications is that there are multiple expression formats for RDF data, not just a single language.
- Common languages are N-Triples, Turtle, and RDF/XML.
- In addition to RDF, there are other related standards in the ecosystem. Here are the standards you need to become familiar with:

1. ***SPARQL:*** Semantic querying language. Triple store equivalent of SQL. Able also to construct new triples and return as a result set.
2. ***RDF Schema*** (known as RDFS): RDFS helps define the valid statements allowed for a particular schema.
3. ***OWL (the Web Ontology Language):*** Sometimes referred to as RDFS+. A subset of OWL is commonly used to supplement RDF Schema definitions.
4. ***SKOS (Simple Knowledge Organization System):*** W3C standard recommendation that describes using RDF to manage controlled vocabularies, thesauri, taxonomies, and folksonomies.

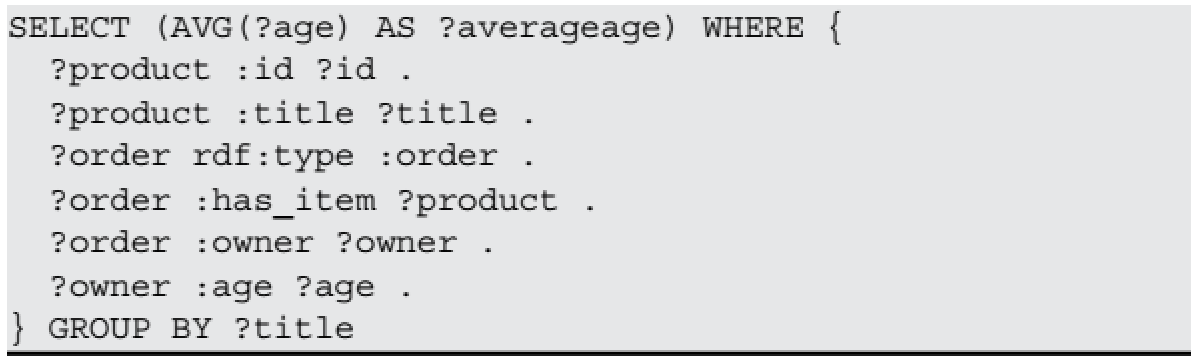
- These specifications allow you to define not only the data in your database but also the structure within that data and how it's organized.
- You can use a triple store by utilizing a single RDF serialization like N-Triples, and SPARQL to query the information the database contains.

Querying with SPARQL

- SPARQL is a recursive acronym that stands for SPARQL Protocol and RDF Query Language.
- SPARQL uses a variant of the Turtle language to provide a query mechanism on databases that store RDF information. SPARQL provides several modes of operation:

1. Select: Returns a set of matching triples from the triple store
2. Ask: Returns whether a query matches a set of triples
3. Construct: Creates new triples based on data in the triple store

- These operations can be restricted to portions of the database using a Where clause.
- ***Using SPARQL 1.1:*** Version 1.1 provides a “group by” structuring mechanism and allows aggregation functions to be performed over triples.
- SPARQL 1.1 also provides a HAVING keyword that acts like a filter clause, except it operates over the result of an aggregation specified in the SELECT clause, rather than a bound variable within the WHERE clause.



```
SELECT (AVG(?age) AS ?averageage) WHERE {
  ?product :id ?id .
  ?product :title ?title .
  ?order rdf:type :order .
  ?order :has_item ?product .
  ?order :owner ?owner .
  ?owner :age ?age .
} GROUP BY ?title
```

Triple Store Structures

- Triple stores provide great flexibility by allowing different systems to use the same data model to describe things.
- RDF Schema (RDFS), OWL, and SKOS allow developers to use the familiar RDF mechanism to describe how these structures interrelate to each other and the existing relations and values.
- ***Describing your ontology:*** An ontology is a semantic model in place within an RDF store.
  - A single store can contain information across many ontologies.
  - Indeed, you can use two ontologies to describe different aspects of the same subject.
  - The main tool used to describe the structures in an RDF ontology is the RDF Schema Language (RDFS).
  - Relationships within triples are directional, thus the semantic web industry's frequent references to directed graphs. The relationship is from one subject to one object.

- ***Enhancing your vocabulary with SKOS:*** A common requirement in using a triple store is to define concepts and how objects fit within those concepts.
  - A single store can contain information across SKOS is used to define vocabularies to describe the preceding scenarios' data modeling needs. A concept is the core SKOS type.
  - Concepts can have preferred labels and alternative labels.
  - Labels provide human readable descriptions.
  - A concept can have a variety of other properties, too, including a note on the scope of the concept.
  - This provides clarification to a user of the ontology as to how a concept should be used.
  - A concept can also have relationships to narrower or broader concepts.
  - A concept can also describe relationships as close matches or exact matches.
  - SKOS provides a web linkable mechanism for describing thesauri, taxonomies, folksonomies, and controlled vocabularies. This can provide a very valuable data modeling technique.
  - In particular, SKOS provides a great way to power drop-down lists and hierarchical navigation user-interface components.
  - So, consider SKOS for times when you need a general-purpose, cross-platform way to define a shared vocabulary, especially if the resulting data ends up in a triple store.

- ***Describing data provenance:*** In larger systems, or systems used over time, you can end up with very complicated interconnected pieces of information.
- How do you prove that the chain of information and events you received, assessments you made, and decisions taken were reasonable and justified for this action to take place?
- Similarly, records, especially documents held in document-orientated NoSQL databases, are changed by people who are often in the same organization.
- This is even more the case when you're dealing with distributed systems like a Wiki.
- How do you describe the changes that content goes through over time, who changed it, and why? This kind of documentation is known as data provenance.
- PROV Ontology (PROV-O) provides a way to describe documents, versions of those documents, changes, the people responsible, and even the software or mechanism used to make the change!
- PROV-O describes some core classes:

1. prov:Entity: The subject being created, changed, or used as input
2. prov:Activity: The process by which an entity is modified
3. prov:Agent: The person or process carrying out the activity on an entity

- These three core classes can be used to describe a range of actions and changes to content.
- They can form the basis for systems to help prove governance is being followed within a data update or action chain.
- PROV-O comprises many properties and relationships.

1. *wasGenertedBy:* Indicates which agent generated a particular entity
2. *wasDerivedFrom:* Shows versioning chains or where data was amalgamated
3. *startedAtTime, endedAtTime:* Provide information on how the activity was performed
4. *actedOnBehalfOf:* Allows a process agent to indicate, for example, which human agent it was running for; also, used to determine when one person performs an operation at the request of another

- PROV-O is a group of standards that includes validator services that can be created and run against a triple store using PROV-O data.

Data Integrity

- Most enterprises expect a database to preserve their data and to protect it from corruption during normal operations.
- This can be achieved through server side features, or settings in a client driver.
- Whatever the approach, the ultimate aim is to store multiple copies of the latest data, ensuring that even if one copy is lost the data stays safe and accessible.
- ***Enabling ACID compliance:***
  - ACID compliance means that a database must guarantee the following:
  - Atomicity: Each change, or set of changes, happens as a single unit. In a transaction, either all the changes are applied or all are abandoned
  - Consistency: The database moves from one consistent state to another. Once data is added, a following request will receive view of the data.
  - Isolation: Each transaction is independent of other transactions running at the same time so that, ideally, each transaction can be played one after the other with the same results. This arrangement is called being fully serializable.
  - Durability: Once data is confirmed as being saved, you're guaranteed it won't be lost.
  - These properties are important features for mission-critical systems where you need absolute guarantees for data safety and consistency.

  - For highly interconnected systems where data is dependent on other data, or where complex relationships are formed, ACID compliance is a necessity.
  - This is because of the unpredictable interdependency of all the subjects held in a triple store.
  - Graph stores tend to guarantee ACID properties only on their primary master server.
  - Because of the complex math involved, graph stores tend not to be sharded — that is, they don't have part of their data residing on different servers.
- ***Sharding and replication for high availability:*** The simplest way to provide high availability is to replicate the data saved on one server to another server.
  - Rather than have these replica servers sit idle, each one is also a master for different parts of the entire triple store.
  - So, if one server goes down, another server can take over the first server's shards (partitions), and the service can continue uninterrupted.
  - This is called a highly available service and is the approach that MarkLogic Server and OrientDB take.
  - An alternative and easier implementation is to make each replica eventually consistent with respect to its view of the data from the master. If a master goes down, you may lose access to a small frame of data, but the service as a whole remains highly available.
  - If you can handle this level of inconsistency, then ArangoDB may be a good open-source alternative to MarkLogic Server and OrientDB

- ***Replication for disaster recovery:***
  - Secondary clusters of MarkLogic Server, OrientDB, and ArangoDB are eventually consistent with their primary clusters.
  - This tradeoff is common across all types of databases that are distributed globally.
  - Primary clusters of Neo4j and AllegroGraph also employ this method between servers in the same site.
  - Their master servers hold the entire database, and replica servers on the same site are updated with changes regularly, but asynchronously.
  - In addition to replicating the master to local replicas, consider replicating the data to a remote replica, too, in case the primary data center is taken offline by a network or power interruption.

Storing Documents with Triples

- Document NoSQL databases typically provide:

1. The ability to store many elements/properties within a single document
2. Concept of a collection for a group of documents
3. Ability to create specialized indexes over structures within their content
4. Ability to join different query terms together to satisfy a request to match a document

- You can map these properties onto their equivalent triple store functionality.
- Specialized indexes are used to ensure that the triple store-specific query functionality is fast.
- Also document NoSQL databases don’t support relationships among documents.
- By applying triple store technology to these databases, you can represent a document as a subject, either explicitly or implicitly, and use triples to describe them, their metadata, relationships, and origin.
- This functionality is provided in two ways, depending on the approach you take with the triple store:

✓Use a document as a representation of a subject/vertex and a relationship/edge (ArangoDB).

✓Use a document as a container for many triples (OrientDB, MarkLogic Server)

- Describing documents:
  - OrientDB and ArangoDB, don’t support storage of metadata about documents outside of the documents themselves.
  - By creating a subject type for a document, you can graft document metadata functionality into these databases.
  - This subject type can hold the ID field of the document and have a predicate and object for every piece of metadata required.
- Combining queries:
  - With a combined query, you query both the document and the triple store in order to answer a question related to all the information in your database.
  - A combined query could be a document provenance query where you want to return all documents in a particular collection
  - If this document changes, the semantic data will change also.
  - In this case, you want to be able to replace the set of information extracted from the document and stored in the triple store.

- There are two mechanisms for doing so:
- ***Use a named graph***: Use the document ID, or a variation of it, for the name of a graph and store all extracted triples in that graph, which makes it easy to update the extracted metadata as a whole.
- This process works for all triple stores.
- The advantage of a named graph is that it works across triple store implementations.
- The downside is that you have to manually create server-side code to execute one query against the triple store and another against the document store in order to resolve your complex document provenance query.
- ***Store the triples in the document they were extracted from***: If your document structure supports embedding different namespaces of information, like MarkLogic Server, you can store an XML representation of the triples in an element inside the document.
- This approach offers the advantage of linking all the required indexes in the same document ID (MarkLogic Server calls this a URI).
- MarkLogic Server has a built-in search engine that includes support for full text, range (less than, greater than) queries, as well as semantic (SPARQL) queries.

Semantic Facts

- Semantic facts are properly called assertions. For most use cases, using the term facts is an easier mental model to picture, so I use the term facts throughout. An assertion is a single triple of subject – predicate – and object.
- ***Extracting context with subjects:***
  - Natural Language Processing (NLP) is the process of looking at words, phrases, and sentences and identifying not only things (people, places, organizations, and so on) but also their relationships.
  - You can store this information as subjects, properties, and relationships — as you do in triple stores. Several pieces of software are available to help you do so. Here are the most popular ones:
  - OpenCalais: An open-source semantic extraction tool
  - Smartlogic Semaphore Server: A commercial entity extraction and enrichment tool that can also generate triples
  - TEMIS Luxid: Another common commercial entity extraction and enrichment tool
  - All of these tools provide entity extraction (identify things) with entity enrichment (add extra information about those things).
  - They also have the ability to generate an output report. This report can be in a list of semantic assertions (OpenCalais) in RDF or XML format.

- ***Forward inferencing:***
  - Inferencing is the ability to take a set of assertions and extrapolate them to other assertions. You can perform backward inferencing and forward inferencing
  - Forward inferencing is typically done at ingestion time, which does lead to slower performance, but it also means that you don’t have to do this work at query time.
  - In this respect, forward inferencing is similar in effect to the denormalization approach of other NoSQL databases.
  - Forward inferencing does allow you to build quite sophisticated webs of stored and inferred facts, which users can employ as helpful shortcuts.

Web of Facts

- You can create a triple store that answers questions by using other peoples' data instead of just your own.
- The Resource Description Framework (RDF) standard was created to allow this web of interconnected data to be built.
- This approach is generally referred to as the Semantic Web.
- There's also a growing movement of open-data publishing.
- This movement has led governments, public sector organizations, research institutions, and some commercial organizations to open up their information and license it to be reused.
- Combining the concepts of open-data publishing and RDF and SPARQL allows you to create Linked Open Data (LOD), which is published data under an open license that refers to other sources of linked open data on other websites.

1. Taking advantage of open data
2. Incorporating data from GeoNames: GeoNames is a database of geographic information that covers a vast range of information
3. Incorporating data from DBpedia: DBpedia contains a semantic extraction of information available within Wikipedia.
4. Linked open-data publishing
5. Migrating RDBMS data

Managing the Social Graph

- Social relationships between people provide a classic example of how a relationship graph is used.
- Social graphs are important in several instances, including the following:

1. Social networks: Include the likes of Facebook, Twitter, and LinkedIn.
2. Professional organizations: To identify individuals or groups in large organizations with required expertise. Also can be used for organizational charts.
3. Organized crime detection: Police and security organizations perform multi-year million-dollar investigations of organized crime networks with many points of contact and relationships.
4. Family tree: Includes many parent, grandparent, uncle, and other relationships. Some assertions from such research may or may not be accurate.

- ***Storing social information***: You can find ontologies to model social data.
  - The FOAF RDF model gives individuals the ability to describe themselves, their interests, their contact details, as well as a list of people they know.
  - This model includes Linked Open Data links to their contacts’ FOAF descriptions, when it’s known.
  - When describing family trees, most people use the Genealogical Data Communication (GEDCOM) file format.

- ***Performing social graph queries***: Triple and graph stores can be used to perform queries on a social graph in several ways.
- Suggest a product to users based on similar items purchased by other people.
- Suggest that users add new friends based on their being friends of your existing friends.
- Suggest groups that users can be a member of based on existing group memberships, friends with similar group memberships, or others with similar likes and dislikes who aren’t in their network.
- You can do the preceding by constructing SPARQL to enable users to do the following:

1. Find all customers who share the same or who have similar product purchases.
2. Order customers by descending value of the number of matches.
3. Find all products those people have ordered that the user hasn’t.
4. Order the list of products by descending value according to the total number of orders for each item across all similar customers.