CS520: KNOWLEDGE GRAPHS
Data Models, Knowledge Acquisition, Inference, Applications

Lectures and Invited Guests
Spring 2021, Tu/Thu 4:30-5:50, cs520.Stanford.edu

Learn about the basic concepts, latest research & applications
What are some Knowledge Graph Data Models?
Outline

• Two Popular Knowledge Graph Data Models
  • Resource Description Framework (RDF) (Query language: SPARQL)
  • Property Graphs (Query language: Cypher)

• Comparison of RDF and Property Graphs

• Comparison of Graph Models with Relational Model

• Limitations of Graph Data Models

• Summary
Resource Description Framework

• Designed to represent information on the web
• Standardized by World Wide Web (W3C) Consortium
RDF Data Model

• Triple is the basic unit of representation
  • Consists of subject, predicate, and object

```
<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>art</td>
<td>knows</td>
<td>bob</td>
</tr>
</tbody>
</table>
```
RDF Data Model

• The nodes can be of three types
  • Internationalized Resource Identifiers (IRI)
    • Uniquely identifies resources on the web
  • Literals
    • A value of certain type (integer, string, etc.)
  • Blank nodes
    • A node with no identifier (anonymous)
Internationalized Resource Identifiers

URL: http://www.wikipedia.org
URI: www.wikipedia.org
IRI: https://hi.wikipedia.org/हिन्दी_विकिपीडिया
Internationalized Resource Identifiers

• Generalization of Uniform Resource Identifiers
  • URIs sequence of characters chosen from a limited subset of the repertoire of US-ASCII
    • Uniform Resource Locator (URL) is a URI that also specifies the method of access
  • IRIs use characters chosen from Universal Character Set (UCS)

Examples:
URL: http://www.wikipedia.org
URI: www.wikipedia.org
IRI: https://hi.wikipedia.org/हिन्दी_विकिपीडिया
Internationalized Resource Identifiers

<http://example.org/art> <http://xmlns.com/foaf/0.1/knows> <http://example.org/bob>
Internationalized Resource Identifiers

We can define prefixes

@prefix foaf: <http://xmlns.com/foaf/0.1/>
@prefix ex: <http://example.org/>

ex:art foaf:knows ex:bob
Literal

• A value of certain type

Examples:

ex:bea foaf:age 23

"1"^^xsd:integer
"01"^^xsd:integer
Blank Nodes

• Used for representing structured information

exstaff:85740  externs:address  "1501 Grant Avenue, Bedford, Massachusetts 01730" .

exstaff:85740  externs:address  _:art_address
  _:art_address  externs:street  "1501 Grant Avenue"
  _:art_address  externs:city  "Bedford"
  _:art_address  externs:state  "Massachusetts"
  _:art_address  externs:zip  "01730"
RDF Vocabulary

- A set of IRIs to be used in describing the data
- RDF graphs are static
  - By providing suitable vocabulary extension dynamics of data may be captured
RDF Dataset

• A collection of RDF graphs with
  • Exactly one default graph
  • One or more named graphs
    • Name can be a blank node or an IRI
Query Language: SPARQL

• Simple Protocol and Query Language (pronounced “sparkl”)
• Queries can go across multiple sources
  • Show me on a map the birthplace of people who died in Winterthour
• Full-featured query language
  • Required/optional parameters
  • Filtering the results
  • Results can be graphs
Query Language: SPARQL

• Example: Who are the persons that art knows?

SELECT ?person
WHERE
<http://example.org/art> <http://xmlns.com/foaf/0.1/knows> ?person
Query Language: SPARQL

• Example: Who are the persons known by the persons that art knows?

SELECT ?person ?person1
WHERE
<http://example.org/art> <http://xmlns.com/foaf/0.1/knows> ?person
?person <http://xmlns.com/foaf/0.1/knows> ?person1

<table>
<thead>
<tr>
<th>?person</th>
<th>?person1</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://example.org/bob">http://example.org/bob</a></td>
<td><a href="http://example.org/cal">http://example.org/cal</a></td>
</tr>
<tr>
<td><a href="http://example.org/bob">http://example.org/bob</a></td>
<td><a href="http://example.org/cam">http://example.org/cam</a></td>
</tr>
<tr>
<td><a href="http://example.org/bea">http://example.org/bea</a></td>
<td><a href="http://example.org/coe">http://example.org/coe</a></td>
</tr>
<tr>
<td><a href="http://example.org/bea">http://example.org/bea</a></td>
<td><a href="http://example.org/cory">http://example.org/cory</a></td>
</tr>
</tbody>
</table>
Query Language: SPARQL

PREFIX ex: <http://example.org/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?person ?person1
WHERE
ex:art foaf:knows ?person
?person foaf:knows ?person1

Basic graph pattern match
Query Language: SPARQL

@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix : <http://example.org/book/> .
@prefix ns: <http://example.org/ns#> .

:book1 ns:price 42 .

PREFIX dc: <http://purl.org/dc/elements/1.1/> 
SELECT ?title 
WHERE  { ?x dc:title ?title 
  FILTER regex(?title, "^SPARQL") 
} 

<table>
<thead>
<tr>
<th>title</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;SPARQL Tutorial&quot;</td>
</tr>
</tbody>
</table>
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix :  <http://example.org/book/> .
@prefix ns: <http://example.org/ns#> .

:book1 ns:price 42 .

PREFIX dc: <http://purl.org/dc/elements/1.1/> 
PREFIX ns: <http://example.org/ns#> 
SELECT ?title ?price 
WHERE { ?x ns:price ?price .
FILTER (?price < 30.5)
?x dc:title ?title . }

<table>
<thead>
<tr>
<th>?title</th>
<th>?price</th>
</tr>
</thead>
<tbody>
<tr>
<td>“The Semantic Web”</td>
<td>23</td>
</tr>
</tbody>
</table>
Query Language: SPARQL

• Instead of SELECT, we can use CONSTRUCT
  • Returns a graph
• Queries can contain more than one graph pattern
• Eliminate duplicates, total number of results
Outline

• Two Popular Knowledge Graph Data Models
  • Resource Description Framework (RDF) (Query language: SPARQL)
    • Property Graphs (Query language: Cypher)
• Comparison of RDF and Property Graphs
• Comparison of Graph Models with Relational Model
• Limitations of Graph Data Models
• Summary
Property Graph Data Model

• Used by many graph databases
• General graph data
  • Do not require a predefined schema
• Optimize graph traversals
Property Graph Data Model

• Nodes, relationships and properties
• Each node and a relationship has a label and set of properties
• Properties are key value pairs
  • Keys are strings, values can be any data types
• Each relationship has a direction
Property Graph Data Model

- Nodes, relationships and properties
- Each node and a relationship has a label and set of properties
- Properties are key value pairs
  - Keys are strings, values can be any data types
- Each relationship has a direction
Property Graph Data Model

- Nodes, relationships and properties
- Each node and a relationship has a label and set of properties
- Properties are key value pairs
  - Keys are strings, values can be any data types
- Each relationship has a direction
Query Language: Cypher

• Query language for querying graph data
• Being considered for adoption as an ISO Standard
• Supports CRUD operations
  • Create, read, update, delete
Query Language: Cypher

• Which people does art know?

MATCH (p1:Person {name: art}) -[:knows]-> (p2: Person)
RETURN p2
Query Language: Cypher

• Which people does art know since 2010?

MATCH (p1:Person {name: art}) -[:knows {since: 2010}]-> (p2: Person)
RETURN p1, p2
Query Language: Cypher

• Which people does art know since 2010?

MATCH (p1:Person) -[:knows {since: Y}]-> (p2: Person)
WHERE Y <= 2010
RETURN p1, p2

• WHERE clause can be used to specify a variety of filtering constraints
Query Language: Cypher

• Constructs for
  • Counting
  • Grouping
  • Aggregating
  • Min/Max
Outline

• Two Popular Knowledge Graph Data Models
  • Resource Description Framework (RDF) (Query language: SPARQL)
  • Property Graphs (Query language: Cypher)

• Comparison of RDF and Property Graphs

• Comparison of Graph Models with Relational Model

• Limitations of Graph Data Models

• Summary
RDF and Property Graphs

• RDF supports several additional layers
  • RDF Schema, Web Ontology, etc.

• Basic differences
  • Property graph model supports edge properties
  • Property graph model does not require IRIs
  • Property graph model does not support blank nodes
Reification in RDF

• Suppose we wish to specify the provenance of a triple

exproducts:item10245 exterms:weight "2.4"^^xsd:decimal

• We wish to state who took the above measurement
  • In a property graph we would do it using an edge property
Reification in RDF

• Reification Vocabulary
  • rdf:type, rdf:Statement
  • rdf:subject
  • rdf:predicate
  • rdf:object
Reification in RDF

- Reification Vocabulary
  - rdf:type rdf:Statement
  - rdf:subject
  - rdf:predicate
  - rdf:object

```
exproducts:item10245 exterms:weight "2.4"^^xsd:decimal
exproducts:triple12345 rdf:subject exproducts:item10245 .
exproducts:triple12345 rdf:object "2.4"^^xsd:decimal .
```
Translating Property Graphs into RDF

- Property Graph
  - Node properties
  - Edges
  - Edge properties

- RDF
  - Triples
  - Triples
  - Reified edges + Triples
Translating Property Graphs into RDF

- Property Graph
  - Subject and object become nodes
  - with predicates as the edges between those nodes

- RDF
  - Triples
Translating Property Graphs into RDF

• Property Graph
  • Subject and object become nodes with predicates as the edges between those nodes

• RDF
  • Triples
  • Create new nodes only for those RDF nodes that are IRIs or blank nodes
  • Literals become node properties
RDF and Property Graphs

• RDF supports several additional layers
  • RDF Schema, Web Ontology, etc.

• Basic differences
  • Property graph model supports edge properties
  • Property graph model does not require IRIs
  • Property graph model does not support blank nodes

• Similarities
  • Data in one can be inter-converted into the other
Graph Model and Relational Model

• Graphs are easier to understand
  • Relational schemas can be visualized

• Graph queries are more compact and faster
  • Translator from graph queries to relational queries can be written
### Example

**Employee**

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>ssn</th>
</tr>
</thead>
<tbody>
<tr>
<td>e01</td>
<td>alice</td>
<td>...</td>
</tr>
<tr>
<td>e02</td>
<td>bob</td>
<td>...</td>
</tr>
<tr>
<td>e03</td>
<td>charlie</td>
<td>...</td>
</tr>
<tr>
<td>e04</td>
<td>dana</td>
<td>...</td>
</tr>
</tbody>
</table>

**Employee_Department**

<table>
<thead>
<tr>
<th>employee id</th>
<th>department id</th>
</tr>
</thead>
<tbody>
<tr>
<td>e01</td>
<td>d01</td>
</tr>
<tr>
<td>e01</td>
<td>d02</td>
</tr>
<tr>
<td>e02</td>
<td>d01</td>
</tr>
<tr>
<td>e03</td>
<td>d02</td>
</tr>
<tr>
<td>e04</td>
<td>d03</td>
</tr>
</tbody>
</table>

**Department**

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>d01</td>
<td>IT</td>
<td>...</td>
</tr>
<tr>
<td>d02</td>
<td>Finance</td>
<td>...</td>
</tr>
<tr>
<td>d03</td>
<td>HR</td>
<td>...</td>
</tr>
</tbody>
</table>
### Example

#### Employee

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>ssn</th>
</tr>
</thead>
<tbody>
<tr>
<td>e01</td>
<td>alice</td>
<td>...</td>
</tr>
<tr>
<td>e02</td>
<td>bob</td>
<td>...</td>
</tr>
<tr>
<td>e03</td>
<td>charlie</td>
<td>...</td>
</tr>
<tr>
<td>e04</td>
<td>dana</td>
<td>...</td>
</tr>
</tbody>
</table>

#### Employee_Department

<table>
<thead>
<tr>
<th>employee id</th>
<th>department id</th>
</tr>
</thead>
<tbody>
<tr>
<td>e01</td>
<td>d01</td>
</tr>
<tr>
<td>e01</td>
<td>d02</td>
</tr>
<tr>
<td>e02</td>
<td>d01</td>
</tr>
<tr>
<td>e03</td>
<td>d02</td>
</tr>
<tr>
<td>e04</td>
<td>d03</td>
</tr>
</tbody>
</table>

#### Department

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>d01</td>
<td>IT</td>
<td>...</td>
</tr>
<tr>
<td>d02</td>
<td>Finance</td>
<td>...</td>
</tr>
<tr>
<td>d03</td>
<td>HR</td>
<td>...</td>
</tr>
</tbody>
</table>
Example

List the employees in the IT Department
SELECT name FROM Employee
LEFT JOIN Employee_Department
  ON Employee.Id = Employee_Department.EmployeeId
LEFT JOIN Department
  ON Department.Id = Employee_Department.DepartmentId
WHERE Department.name = "IT"
**Example**

List the employees in the IT Department

```sql
SELECT name FROM Employee
LEFT JOIN Employee_Department
  ON Employee.Id = Employee_Department.EmployeeId
LEFT JOIN Department
  ON Department.Id = Employee_Department.DepartmentId
WHERE Department.name = "IT"
```

```cypher
MATCH (p:Employee) -[:works_in]-> (d:Department)
WHERE d.id = "IT"
RETURN p
```
Mapping Graph Model to Relational Model

• Provide two relational tables
  • A table that represents node properties and relationships as triples
  • A table that represents edge properties as four tuples
Mapping Graph Model to Relational Model

• Provide a translator from graph queries to relational queries
  • Incorporate optimizations in the translator
  • Can optimize queries across the graph data and legacy data in relational systems

Diagram:
- Graph Queries
  - Translator/Optimizer
  - Graph Data
  - Legacy Relational Data
Graph Model and Relational Model

• Graphs are easier to understand
  • Relational schemas can be visualized
• Graph queries are more compact and faster
  • Translator from graph queries to relational queries can be written
Limitations of the Graph Model

• Triples are not always sufficient
  • For example, the ternary relationships such as between

• Time series data is naturally modeled in relations
  • Evolving population of a country over a period of time
Summary

• RDF/SPARQL and Property Graph / Cypher are common graph data models in use today
• RDF addresses the need to model information on the web, while Property Graphs are used as a model in general graph databases
• Translations exist between RDF and property graph models
• Translations also exist from graph models to relations
• Unique features of graph models
  • More compact queries
  • Optimized for traversals
  • Graphical visualization
Distributed SPARQL Execution

Querying Property Graphs with [open]Cypher