Stanford·CS520 | Knowledge Graphs (2021)

CS520(2021)· 课程资料包 @ShowMeAI

视频	课件	笔记	代码
中英双语字幕	一键打包下载	官方笔记翻译	作业项目解析

视频 ·B 站 [扫码或点击链接]

https://www.bilibili.com/video/BV1hb4y1r7fF

课件 & 代码 · 博客 [扫码或点击链接]

http://blog.showmeai.tech/cs520

斯坦福 实体关系 图谱应用 知识图谱 图谱构建 图谱 schema 实体 非结构化数据 知识推理

Awesome AI Courses Notes Cheatsheets 是 ShowMeAI 资料库的分支系列，覆盖最具知名度的 TOP20+ 门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

点击课程名称，跳转至课程**资料包**页面，**一键下载**课程全部资料！

机器学习	深度学习	自然语言处理	计算机视觉
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n
# Awesome AI Courses Notes Cheatsheets· 持续更新中			
知识图谱	图机器学习	深度强化学习	自动驾驶
Stanford · CS520	Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094

微信公众号

资料下载方式 2：扫码点击底部菜单栏

称为 **AI 内容创作者?** 回复 [添砖加瓦]

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

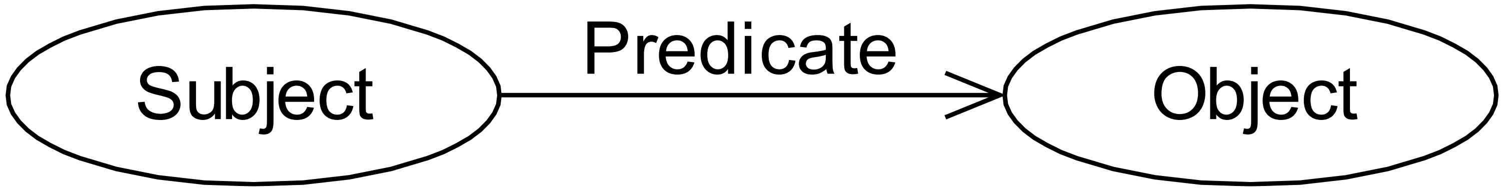
- Two Popular Knowledge Graph Data Models
  - Resource Description Framework (RDF) (Query language: SPARQL)
  - Property Graphs (Query language: Cypher)
- Comparison of RDF and Property Graphs
- Comparison of Graph Models with Relational Model
- Limitations of Graph Data Models
- Summary

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



art knows bob

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)

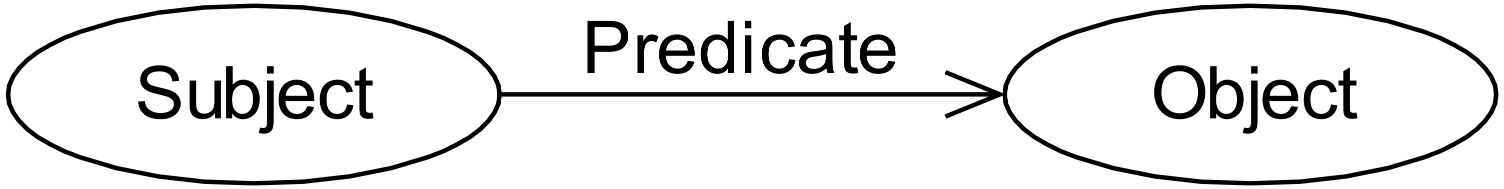
Examples:

URL: http://www.wikipedia.org

URI: www.wikipedia.org

IRI: https://hi.wikipedia.org/हिन्दी_विकिपीडिया

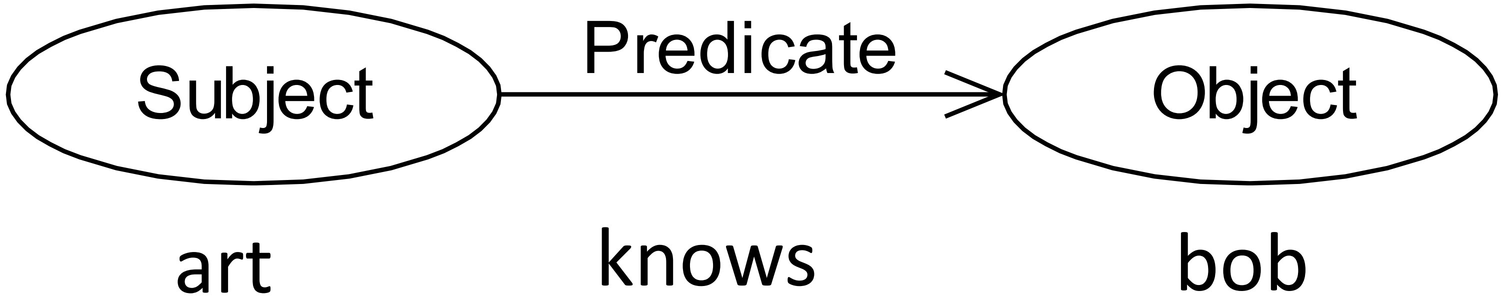
Internationalized Resource Identifiers



art knows bob

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

Internationalized Resource Identifiers



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

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   exterms:address   "1501 Grant Avenue, Bedford, Massachusetts 01730" .

exstaff:85740   exterms:address   _:art_address

_:art_address exterms:street “1501 Grant Avenue”

_:art_address exterms:city “Bedford”

_:art_address exterms:state “Massachusetts”

_:art_address exterms: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?

Graph Pattern

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

?person1
<http://example.org/bob>
<http://example.org/bea>

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
```

?person	?person1
<http://example.org/bob>	<http://example.org/cal>
<http://example.org/bob>	<http://example.org/cam>
<http://example.org/bea>	<http://example.org/coe>
<http://example.org/bea>	<http://example.org/cory>

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
```

?person	?person1
<http://example.org/bob>	<http://example.org/cal>
<http://example.org/bob>	<http://example.org/cam>
<http://example.org/bea>	<http://example.org/coe>
<http://example.org/bea>	<http://example.org/cory>

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  dc:title  "SPARQL Tutorial" .
:book1  ns:price  42 .
:book2  dc:title  "The Semantic Web" .
:book2  ns:price  23 .
```

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

title
“SPARQL Tutorial”

Query Language: SPARQL

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

:book1  dc:title  "SPARQL Tutorial" .
:book1  ns:price  42 .
:book2  dc:title  "The Semantic Web" .
:book2  ns:price  23 .
```

```
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 . }
```

?title	?price
“The Semantic Web”	23

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

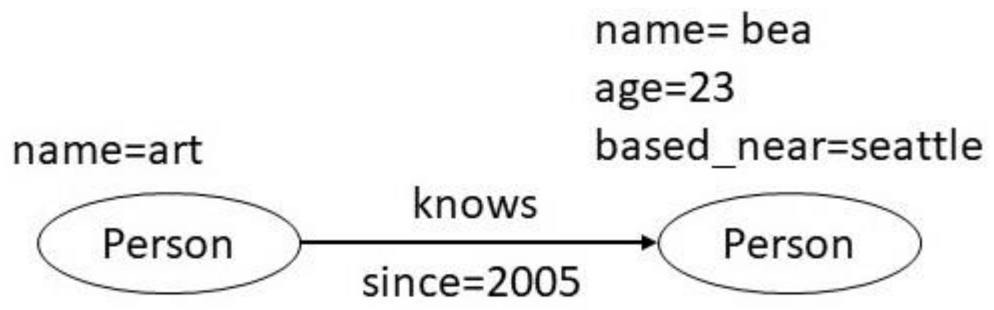
- Used by many graph databases
- General graph data
  - Do not require a predefined schema
- Optimize graph traversals

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

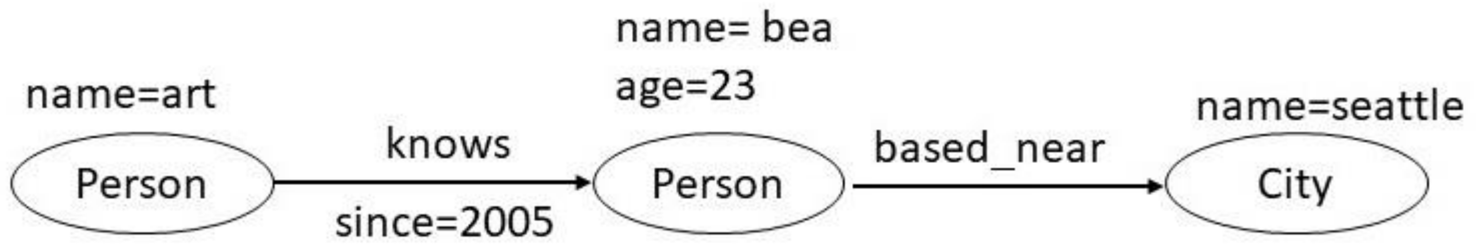
Property Graph Data Model

- 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:type rdf:Statement .
exproducts:triple12345 rdf:subject exproducts:item10245 .
exproducts:triple12345 rdf:predicate exterms:weight .
exproducts:triple12345 rdf:object "2.4"^^xsd:decimal .
exproducts:triple12345 dc:creator exstaff:85740 .
```

Translating Property Graphs into RDF

• Property Graph		• RDF
• Node properties	$\longrightarrow$	• Triples
• Edges	$\longrightarrow$	• Triples
• Edge properties	$\longrightarrow$	• Reified edges + Triples

Translating Property Graphs into RDF

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

- RDF
  - $\longleftarrow$ Triples

Translating Property Graphs into RDF

- Property Graph
  - Subject and object become nodes with predicates as the edges between those nodes ⟵
  - Create new nodes only for those RDF nodes that are IRIs or blank nodes ⟵
  - Literals become node properties

- RDF
  - Triples
  - Triples

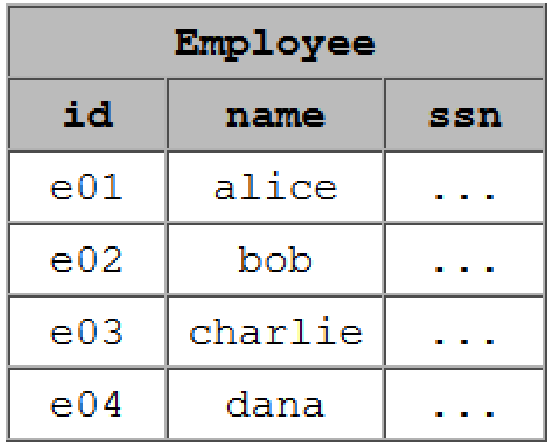
RDF and Property Graphs

- 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

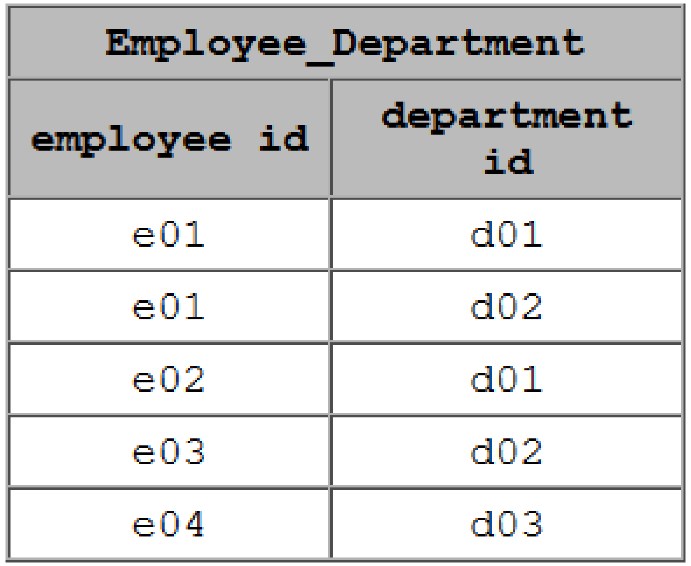
Graph Model and Relational Model

- 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

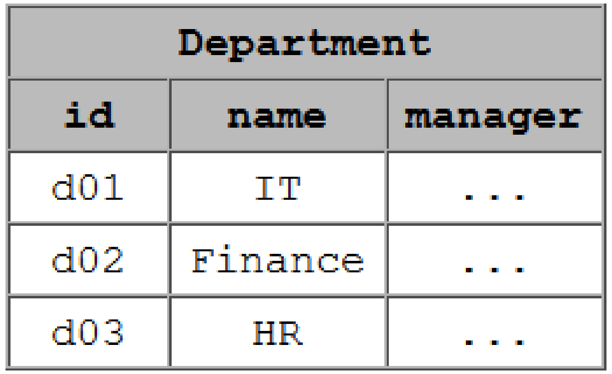
Example



Employee		
id	name	ssn
e01	alice	...
e02	bob	...
e03	charlie	...
e04	dana	...



Employee_Department	
employee id	department id
e01	d01
e01	d02
e02	d01
e03	d02
e04	d03



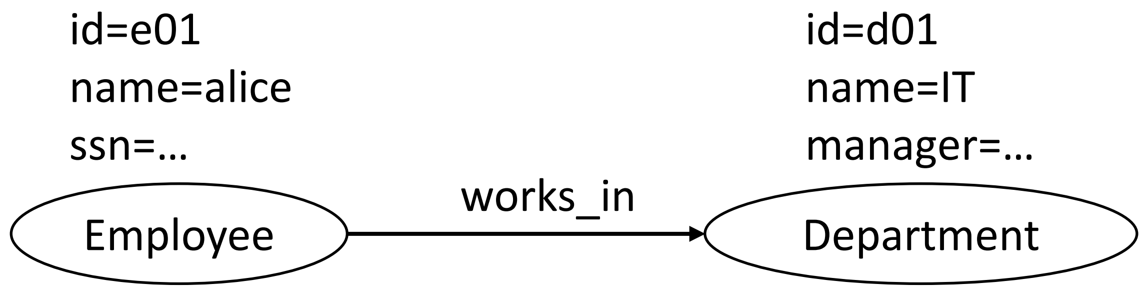
Department		
id	name	manager
d01	IT	...
d02	Finance	...
d03	HR	...

Example

Employee		
id	name	ssn
e01	alice	...
e02	bob	...
e03	charlie	...
e04	dana	...

Employee_Department	
employee id	department id
e01	d01
e01	d02
e02	d01
e03	d02
e04	d03

Department		
id	name	manager
d01	IT	...
d02	Finance	...
d03	HR	...



Example

Employee		
id	name	ssn
e01	alice	...
e02	bob	...
e03	charlie	...
e04	dana	...

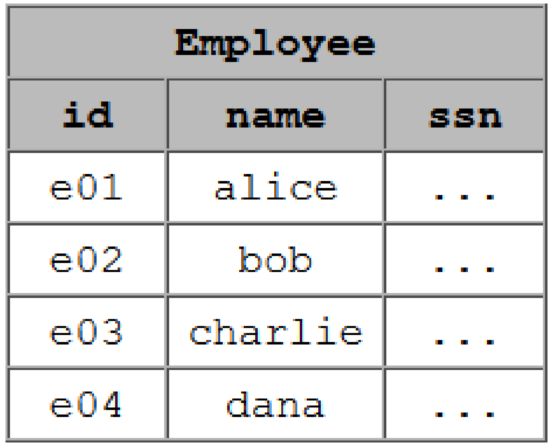
Employee_Department	
employee id	department id
e01	d01
e01	d02
e02	d01
e03	d02
e04	d03

Department		
id	name	manager
d01	IT	...
d02	Finance	...
d03	HR	...

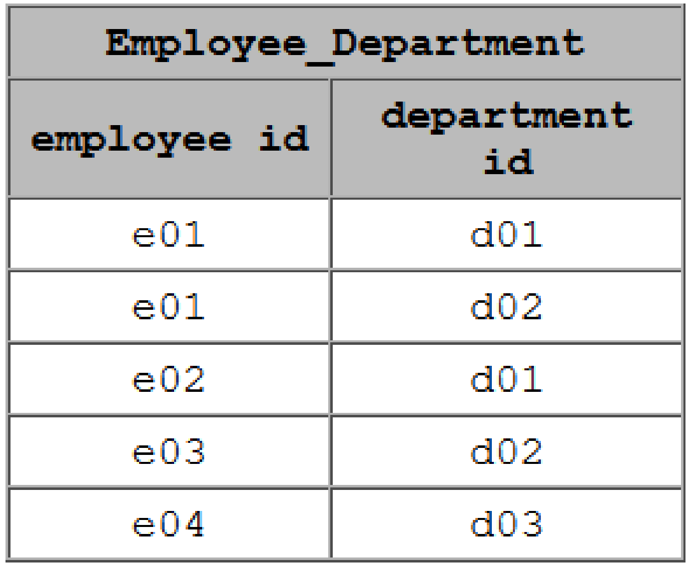
List the employees in the IT Department

```
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"
```

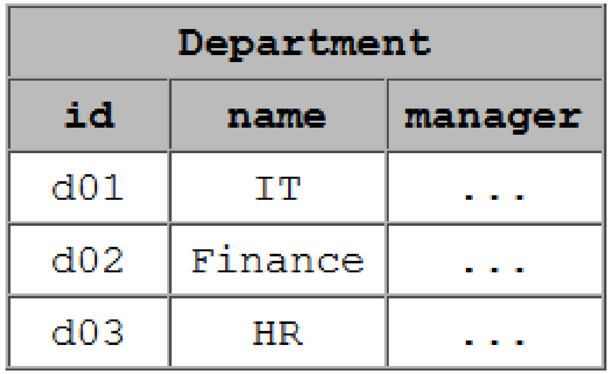
Example



Employee		
id	**name**	**ssn**
e01	alice	...
e02	bob	...
e03	charlie	...
e04	dana	...



Employee_Department	
employee id	**department id**
e01	d01
e01	d02
e02	d01
e03	d02
e04	d03



Department		
id	**name**	**manager**
d01	IT	...
d02	Finance	...
d03	HR	...

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"
```

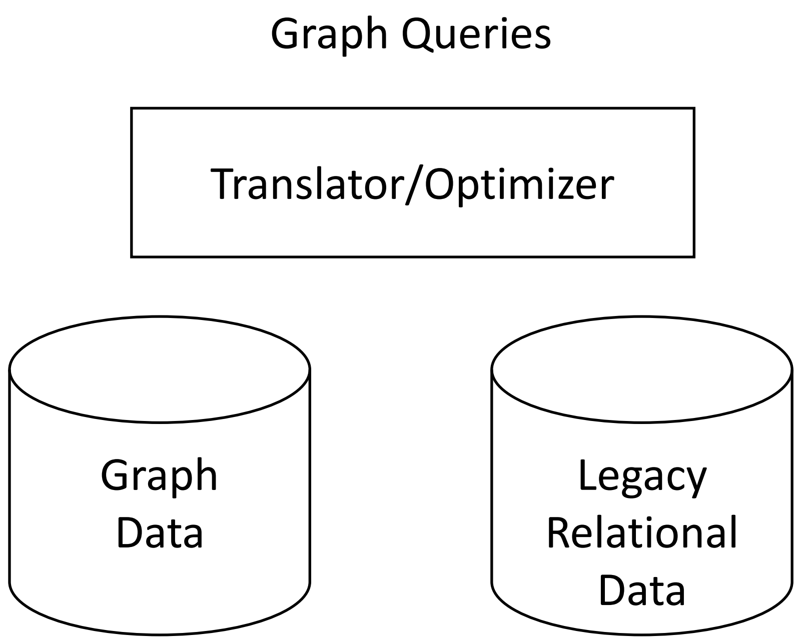
```
MATCH (p:Employee) -[:works_in]-> (d:Department)
WHERE d.id = "IT"
RETURN p
```

Mapping Graph Model to Relational Model

- Provide two relational tables
  - A table that represents node properties and relationships as riples
  - 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



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

Prof. Tamer Özsu

Distributed SPARQL Execution

Dr. Petra Selmer

Querying Property Graphs with [open]Cypher

Stanford·CS520 | Knowledge Graphs (2021)

CS520(2021)· 课程资料包 @ShowMeAI

视频	课件	笔记	代码
中英双语字幕	一键打包下载	官方笔记翻译	作业项目解析

视频 ·B 站 [扫码或点击链接]

https://www.bilibili.com/video/BV1hb4y1r7fF

课件 & 代码 · 博客 [扫码或点击链接]

http://blog.showmeai.tech/cs520

斯坦福 实体关系 图谱应用 图谱构建 知识图谱 图谱 schema 实体 非结构化数据 知识推理

Awesome AI Courses Notes Cheatsheets 是 ShowMeAI 资料库的分支系列，覆盖最具知名度的 **TOP20+** 门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

点击课程名称，跳转至课程**资料包**页面，**一键下载**课程全部资料！

机器学习	深度学习	自然语言处理	计算机视觉
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n
# Awesome AI Courses Notes Cheatsheets· 持续更新中			
知识图谱	图机器学习	深度强化学习	自动驾驶
Stanford · CS520	Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094

微信公众号

资料下载方式 2：扫码点击底部菜单栏

称为 **AI 内容创作者?** 回复 [添砖加瓦]