TU WIEN Informatics

192.161 Management of Graph Data

(4.0 VU / 6.0 ECTS)

2025W

SPARQL

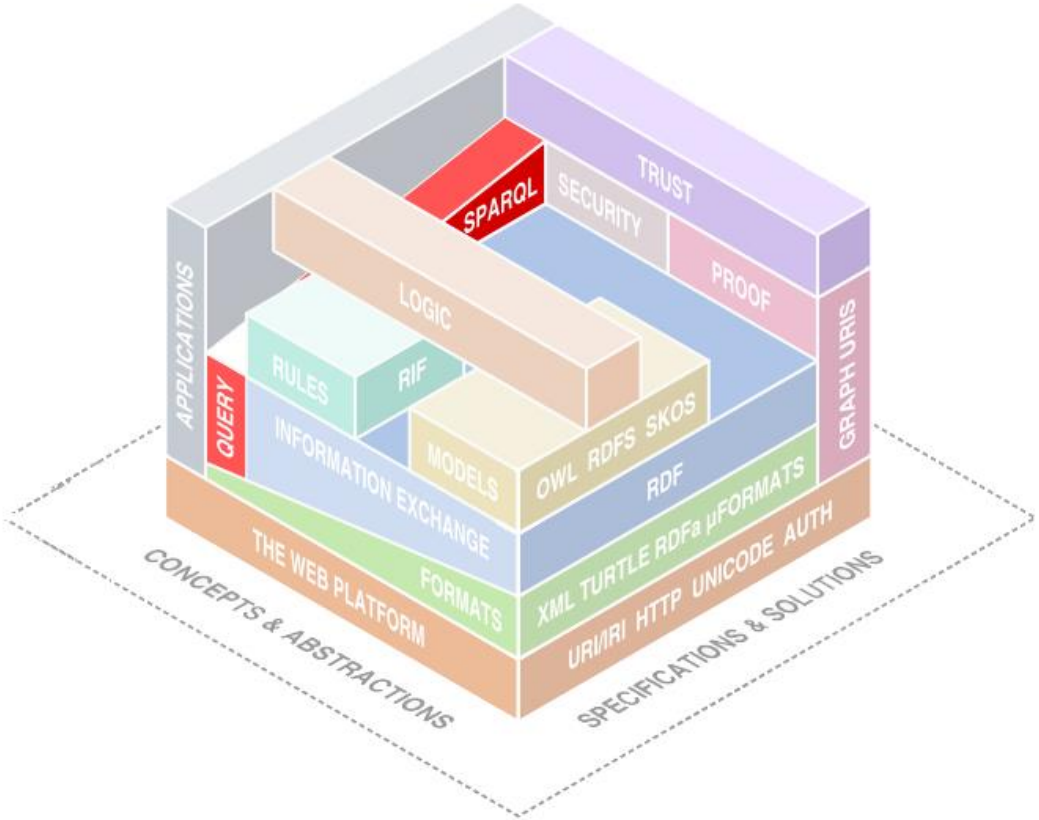
Katja Hose, Maxime Jakubowski
Milos Jovanovik

mogda@list.tuwien.ac.at

Learning Goals

- Querying Knowledge Graphs with SPARQL
- Updating Knowledge Graphs with SPARQL

Semantic Web Foundations



http://bnode.org/blog/2009/07/08/the-semantic-web-not-a-piece-of-cake

SPARQL

Agenda

- SPARQL basics
  - Introductory notions
  - The SELECT query
  - Other query types
- Advanced SPARQL
  - CONSTRUCT queries
  - Filter & Order
  - Aggregate & Group
  - Union
  - Subqueries
  - Federated queries
  - Optional
  - Entailment regimes
- Updates with SPARQL

Introduction

- SPARQL = **S**PARQL **P**rotocol and **R**DF **Q**uery **L**anguage
  - Semantic query language
  - Developed by W3C RDF Data Access Working group (DAWG)
  - January 2008: SPARQL 1.0
  - March 2013: SPARQL 1.1
    - https://www.w3.org/TR/sparql11-query/
  - November 2024: SPARQL 1.2
    - https://www.w3.org/TR/sparql12-query/

Introduction

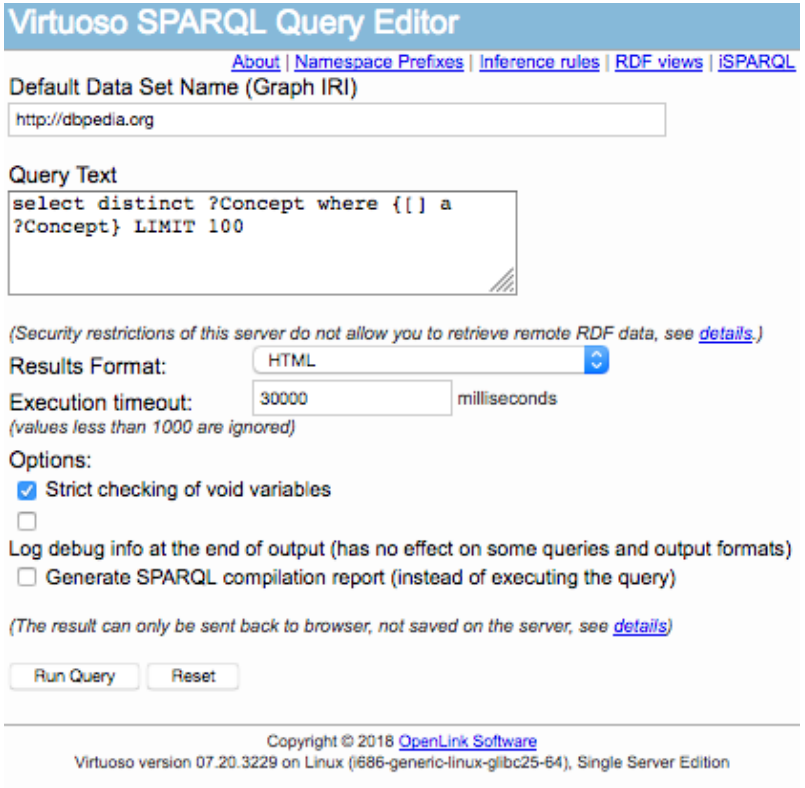
- SPARQL lets us
  - Retrieve and manipulate data stored in RDF
  - Explore data by querying unknown relationships
  - Perform complex joins of disparate databases in a single, simple query
  - Transform RDF data from one vocabulary to another

- Triple stores
  - Ontotext GraphDB (https://www.ontotext.com/products/graphdb/)
  - OpenLink Virtuoso (https://virtuoso.openlinksw.com)
  - Apache Jena ARQ (http://jena.apache.org)
  - Many more: https://www.w3.org/wiki/SparqlImplementations

SPARQL Fundamentals

- **RDF graph**: Set of RDF assertions, manipulated as a labeled directed graph.
- **RDF dataset**: set of RDF triples; comprised of
  - one default graph
  - zero or more named graphs
- **SPARQL protocol client**: HTTP client that sends requests for SPARQL Protocol operations (queries or updates)
- **SPARQL protocol service**: HTTP server that services requests for SPARQL Protocol operations
- **SPARQL endpoint**: The URI at which a SPARQL Protocol service listens for requests from SPARQL clients

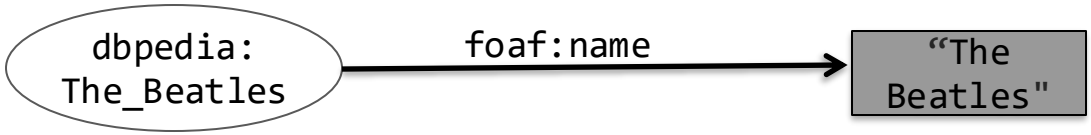
https://dbpedia.org/sparql



SPARQL Syntax Basics

RDF triple

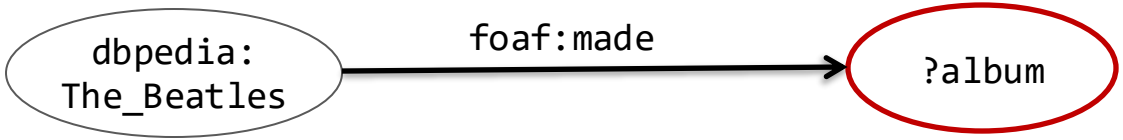
```
dbpedia:The_Beatles foaf:name "The Beatles" .
```



RDF triple pattern

```
dbpedia:The_Beatles foaf:made ?album .
```

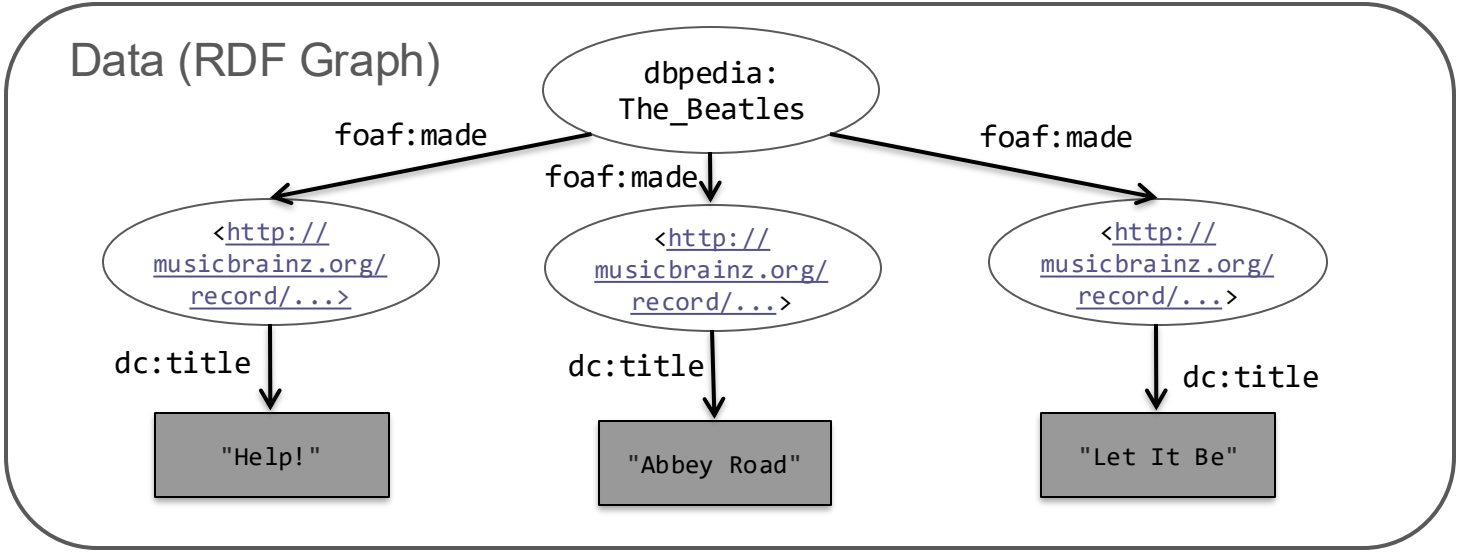
Just like triples, but any of their parts can be replaced by a variable.

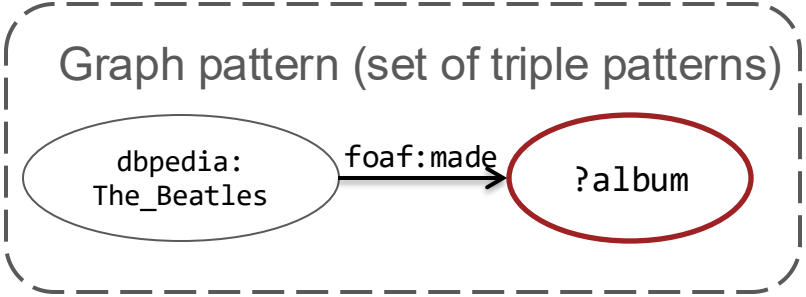


Other triple pattern examples:

```
?album mo:track ?track .
?s ?p ?o .
```

SPARQL Query Example



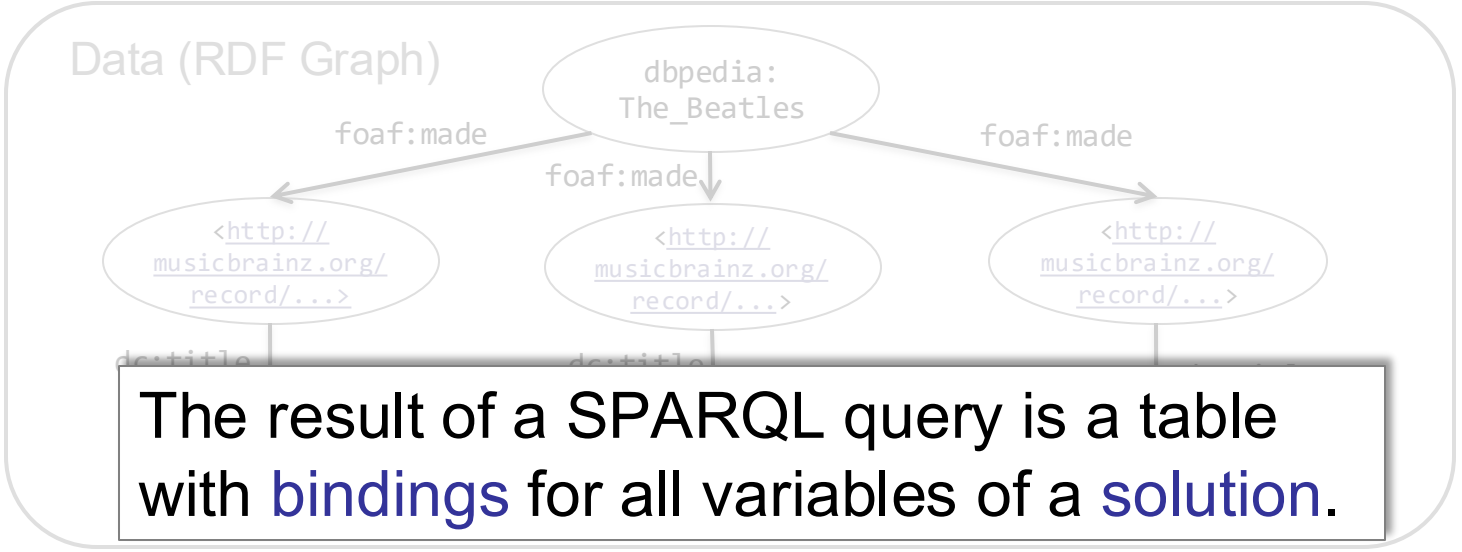


SPARQL

Results

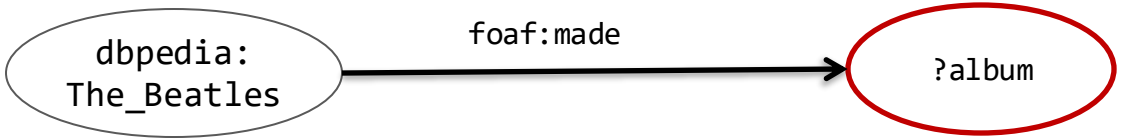
?album
<http://musicbrainz.org...>
<http://musicbrainz.org...>
<http://musicbrainz.org...>

SPARQL Query Example

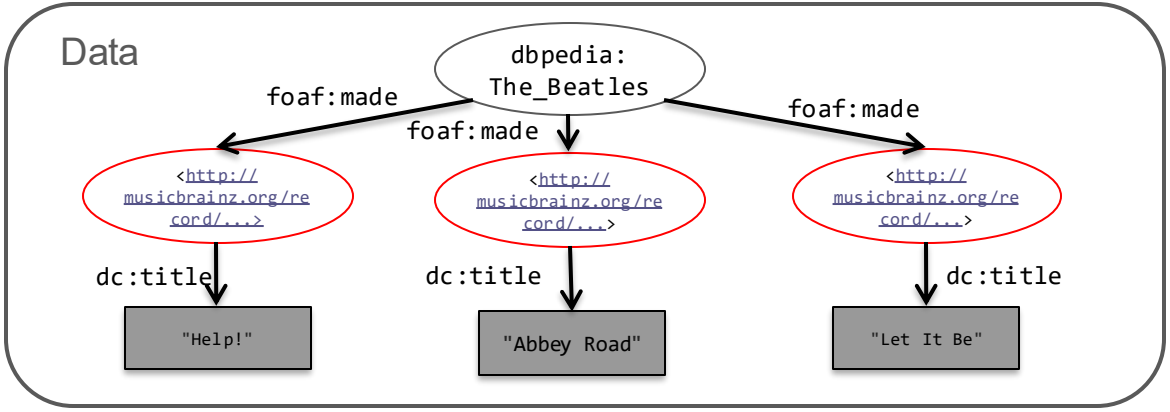


SPARQL Queries

- Key idea: Pattern matching
- **Queries** (graph patterns):
  - describe subgraphs of an RDF graph
  - Roughly correspond to RDF graphs specified in Turtle syntax but containing variables (prefixed by **?**)



- **Results**: Subgraphs matching the graph pattern
  → or rather a table with the variables of those subgraphs



Structure of a SPARQL query

```
# Prefix declarations, for abbreviating URIs
PREFIX dbpedia: http://dbpedia.org/resource/
...

# Result clause, identifying what information to return from the query
SELECT/ASK/CONSTRUCT ...

# Dataset definitions, stating what RDF graph(s) are being queried
FROM  <http://musicbrainz.org/20130302>

# Query pattern, specifying what to query for in the underlying dataset
WHERE {
    ...
}

# Query modifiers, slicing, ordering, rearranging query results
ORDER BY ...
```

Agenda

- SPARQL basics
  - Introductory notions
  - The SELECT query
  - Other query types
- Advanced SPARQL
  - CONSTRUCT queries
  - Filter & Order
  - Aggregate & Group
  - Union
  - Subqueries
  - Federated queries
  - Optional
  - Entailment regimes
- Updates with SPARQL

Anatomy of a SELECT query

```
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?album
FROM  <http://musicbrainz.org/20130302>
WHERE {
    dbpedia:The_Beatles foaf:made ?album .
}
```

Query form:

- SELECT retrieves variables and their bindings as a table
- Other query forms: ASK, DESCRIBE or CONSTRUCT

?album
<http://musicbrainz.org...>
<http://musicbrainz.org...>
<http://musicbrainz.org...>

Anatomy of a SELECT query

```
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?album
FROM  <http://musicbrainz.org/20130302>
WHERE {
    dbpedia:The_Beatles foaf:made ?album .
}
```

Dataset specification

- This clause is optional
- `FROM` or `FROM NAMED`
- Indicates the sources for the data against which to find matches

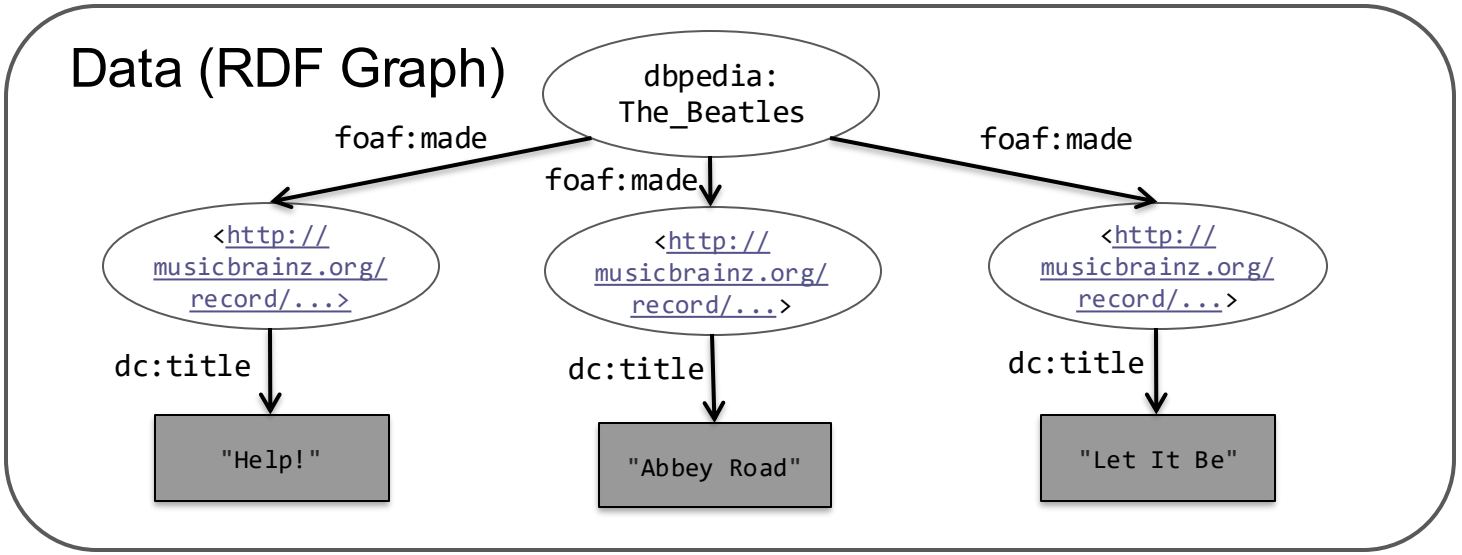
Anatomy of a SELECT query

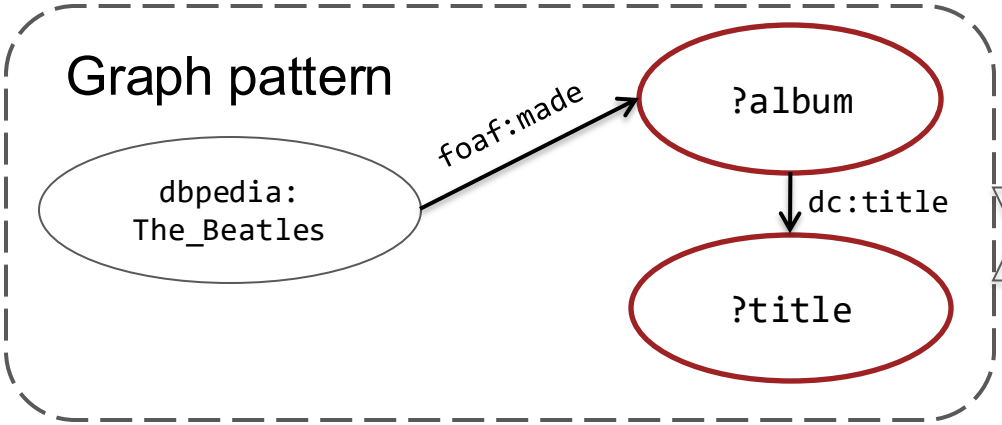
```
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?album
FROM  <http://musicbrainz.org/20130302>
WHERE {
    dbpedia:The_Beatles foaf:made ?album .
}
```

Query pattern

- Generalizes Turtle with variables and keywords
  - The final "." after the last triple pattern is optional
- One or multiple triple patterns between "{}"
  - together describing a graph pattern to be matched against RDF triples specified in the FROM clause

SPARQL Query Example

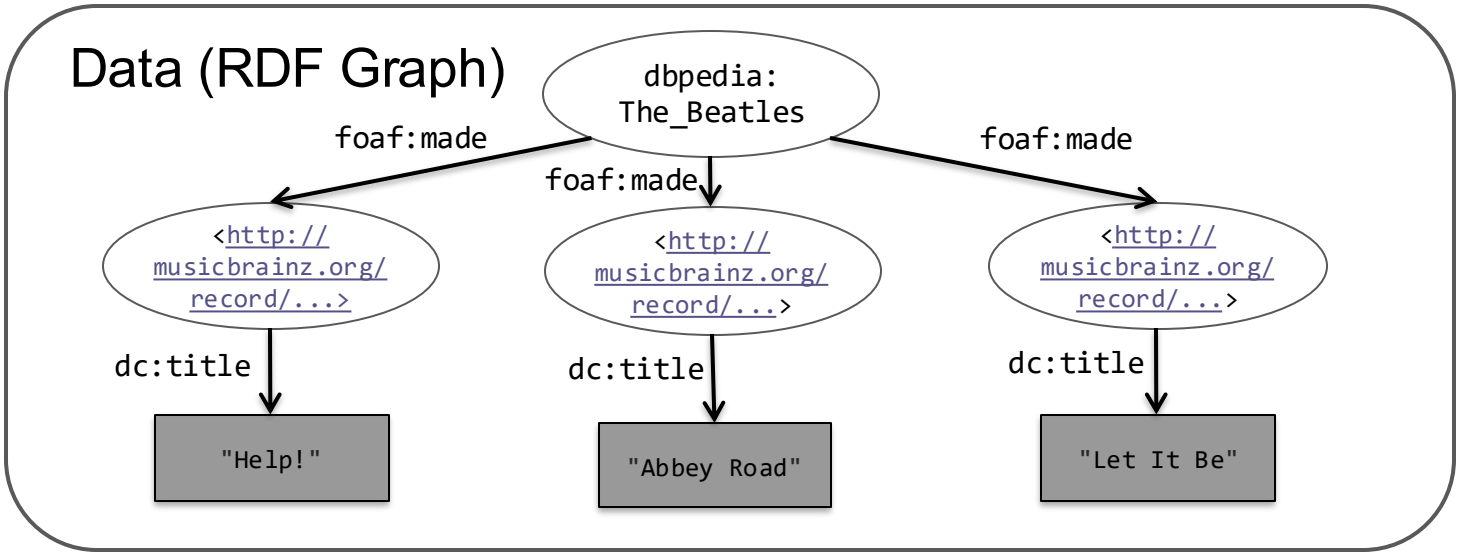


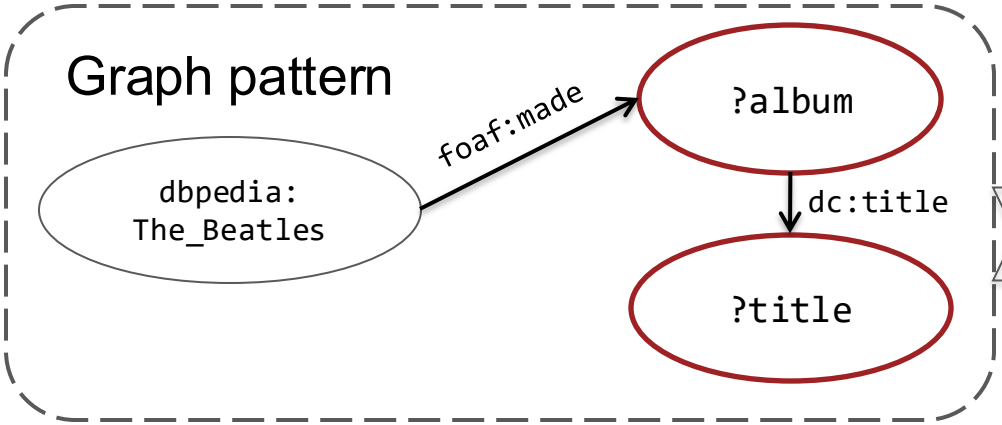


Results

What are the results?
→ Menti

SPARQL Query Example





Results

?album	?title
<http://...>	"Help!"
<http://...>	"Abbey Road"
<http://...>	"Let It Be"

Anatomy of a SELECT query

```
PREFIX dbpedia, foaf, dc, mo

SELECT *
FROM  <http://musicbrainz.org/20130302>
WHERE {
    dbpedia:The_Beatles foaf:made ?album .
    ?album dc:title ?title .
}
ORDER BY ?title
```

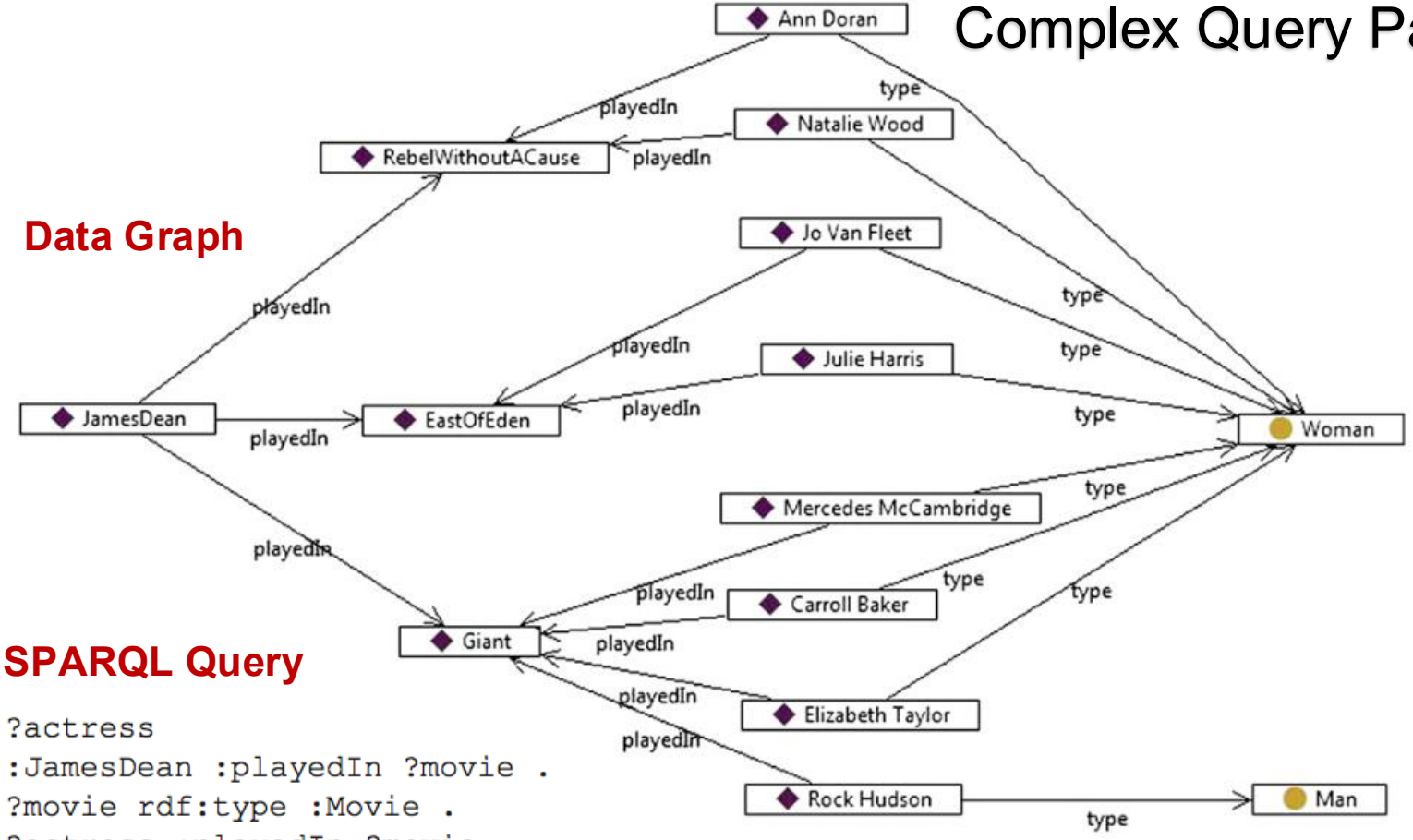
Solution modifiers:

- Modify the result set
- `ORDER BY`, `LIMIT` or `OFFSET`
- `GROUP BY`
- `SELECT *`

?album	?title
<http://...>	"Abbey Road"
<http://...>	"Help!"
<http://...>	"Let It Be"

Complex Query Patterns

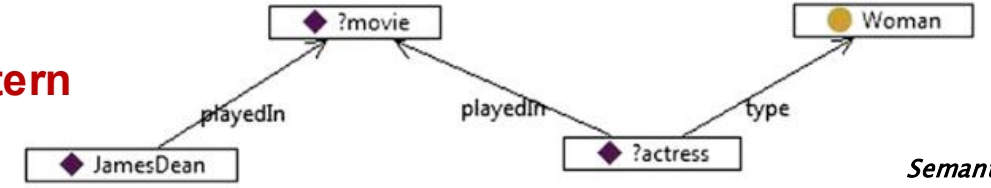
Data Graph



SPARQL Query

```
SELECT ?actress
WHERE {:JamesDean :playedIn ?movie .
       ?movie rdf:type :Movie .
       ?actress :playedIn ?movie .
       ?actress rdf:type :Woman }
```

Query Pattern



Semantic Web for the Working Ontologist, p71

Agenda

- SPARQL basics
  - Introductory notions
  - The SELECT query
  - Other query types
- Advanced SPARQL
  - CONSTRUCT queries
  - Filter & Order
  - Aggregate & Group
  - Union
  - Subqueries
  - Federated queries
  - Optional
  - Entailment regimes
- Updates with SPARQL

Query Types

SPARQL supports the following query types

- **SELECT**
  returns variables and their bindings directly
- **ASK**
  tests whether or not a query pattern has a solution
  Returns true or false
- **DESCRIBE**
  returns a single RDF graph containing RDF data about a resource
- **CONSTRUCT**
  returns a single RDF graph specified by a graph template

SPARQL ASK

ASK - tests whether or not a query pattern has a solution. Returns true/false

Query: *Is Paul McCartney a member of 'The Beatles'?*

```
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX mo: <http://purl.org/ontology/mo/>

ASK WHERE { dbpedia:The_Beatles mo:member dbpedia:Paul_McCartney. }
```

In your own words:
What does this query mean?

Results

true

SPARQL ASK

ASK - tests whether or not a query pattern has a solution. Returns true/false

Query: *Is Elvis Presley a member of 'The Beatles'?*

```
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX mo: <http://purl.org/ontology/mo/>

ASK WHERE { dbpedia:The_Beatles mo:member dbpedia:Elvis_Presley. }
```

In your own words:
What does this query mean?

Results

false

SPARQL ASK

ASK - tests whether or not a query pattern has a solution. Returns true/false

Query: *Is Elizabeth Taylor dead?*

```
ASK WHERE { :ElizabethTaylor :diedOn ?any . }
```

Results

true

(there is a triple matching this pattern)

Query: *Were any of the actors of "Giant" born after 1950?*

```
ASK WHERE { ?any :playedIn :Giant .
            ?any :bornOn ?birthday .
            FILTER (?birthday > "1950-01-01"^^xsd:date)}
```

Results

false

(everyone playing in Giant was born before 1950)

SPARQL DESCRIBE

Takes the resources within the solution and provides information about them as RDF statements. They can be identified by:

- Specifying **explicit IRIs**

```
PREFIX dbpedia: <http://dbpedia.org/resource/>
DESCRIBE dbpedia:Paul_McCartney
```

- **Bindings of variables** in the WHERE clause

```
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX mo: <http://purl.org/ontology/mo/>

DESCRIBE ?member
WHERE { dbpedia:The_Beatles mo:member ?member .}
```

Very useful for exploration when you are confronted with a new dataset that you don't know the structure of.

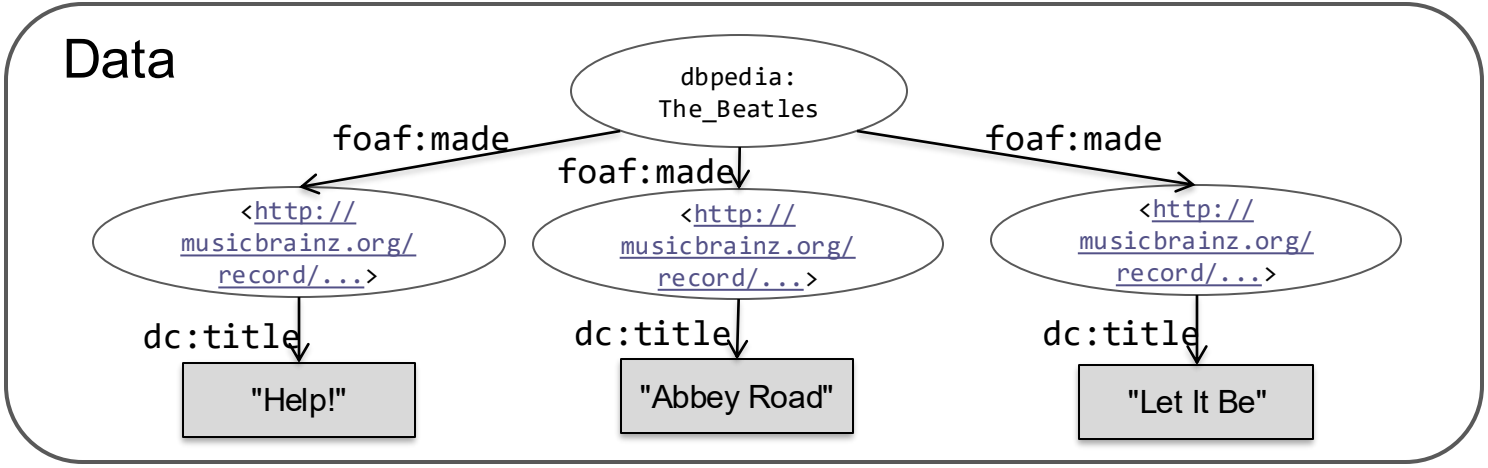
Advanced SPARQL

Agenda

- SPARQL basics
  - Introductory notions
  - The SELECT query
  - Other query types
- Advanced SPARQL
  - CONSTRUCT queries
  - Filter & Order
  - Aggregate & Group
  - Union
  - Subqueries
  - Federated queries
  - Optional
  - Entailment regimes
- Updates with SPARQL

SPARQL CONSTRUCT

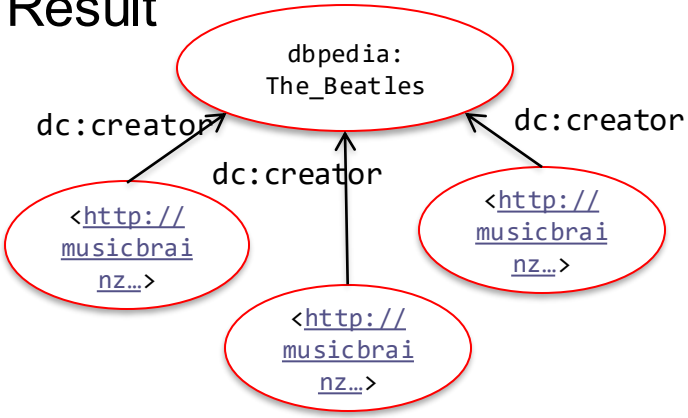
Returns RDF statements created from variable bindings



Query

```
CONSTRUCT {
  ?album dc:creator dbpedia:The_Beatles .}
WHERE {
  dbpedia:The_Beatles foaf:made ?album .}
```

Result



SPARQL CONSTRUCT

Query: *Create the `dc:creator` descriptions for albums and their tracks recorded by 'The Beatles'.*

```
PREFIX dbpedia, foaf, mo, dc ...
CONSTRUCT {
  ?album dc:creator dbpedia:The_Beatles .
  ?track dc:creator dbpedia:The_Beatles .}
WHERE {
  dbpedia:The_Beatles foaf:made ?album .
  ?album mo:track ?track .}
```

SPARQL CONSTRUCT

CONSTRUCT queries allow phrasing rules of the type
"IF this is found THEN generate that"

```
CONSTRUCT {?q1 :hasSibling ?q2} WHERE {?q1 :hasBrother ?q2}
CONSTRUCT {?q1 :hasSibling ?q2} WHERE {?q1 :hasSister ?q2}
CONSTRUCT {?q1 :hasParent ?q2} WHERE {?q1 :hasFather ?q2}
CONSTRUCT {?q1 :hasParent ?q2} WHERE {?q1 :hasMother ?q2}
```

```
CONSTRUCT {?q1 :hasUncle ?q2}
WHERE {?q2 :hasSibling ?parent .
       ?q2 a :Man .
       ?q1 :hasParent ?parent }
```

Semantic Web for the Working Ontologist, p90

Agenda

- SPARQL basics
  - Introductory notions
  - The SELECT query
  - Other query types
- Advanced SPARQL
  - CONSTRUCT queries
  - Filter & Order
  - Aggregate & Group
  - Union
  - Subqueries
  - Federated queries
  - Optional
  - Entailment regimes
- Updates with SPARQL

FILTER

- Different types of **filters and functions** may be used
- Filters use comparison and logical operators
- Positioned **within** the WHERE clause

Query: *Retrieve the albums and tracks recorded by 'The Beatles', where the duration of the song is more than 300 secs. and no longer than 400 secs.*

```
PREFIX dbpedia, foaf, dc, mo

SELECT ?album_name ?track_title ?duration
WHERE {
    dbpedia:The_Beatles foaf:made ?album .
    ?album dc:title ?album_name ;
           mo:track ?track .
    ?track dc:title ?track_title ;
          mo:duration ?duration;
    FILTER (?duration > 300000 && ?duration < 400000)
}
```

Filter expressions as regular expressions over strings

Query: *Create the `dc:creator` descriptions of the albums recorded by 'The Beatles' whose title contains the word "love".*

```
PREFIX dbpedia, foaf, dc

CONSTRUCT { ?album dc:creator dbpedia:The_Beatles . }

WHERE {
    dbpedia:The_Beatles foaf:made ?album .
    ?album dc:title ?album_name ;
    FILTER (REGEX(?album_name, ".*love.*", i))
}
```

Solution Modifier: DISTINCT

- Elimination of duplicates with DISTINCT or REDUCED
- REDUCED does not guarantee elimination of duplicates

Query: *Retrieve the name of the albums recorded by 'The Beatles' that have at least two different songs.*

```
PREFIX dbpedia, foaf, dc, mo

SELECT DISTINCT ?album_name

WHERE {
    dbpedia:The_Beatles foaf:made ?album .
    ?album dc:title ?album_name ;
         mo:track ?track1 .
         mo:track ?track2 .
     FILTER (?track1 != ?track2)
}
```

Results

DISTINCT

?album_name
"Revolver"
"Sessions"
"Abbey Road"

REDUCED

?album_name
"Revolver"
"Revolver"
"Revolver"
"Sessions"
"Abbey Road"
"Abbey Road"

Solution Modifier: ORDER

It is possible to combine the query with solution modifiers (`ORDER BY`, `LIMIT`, `OFFSET`) to get **subsets of results:**

- `ORDER BY (DESC/ASC)`: result set is shown in descending/ascending order
- `LIMIT`: limits the number of results returned by a query

Query *Create the `dc:creator` descriptions for the 10 most recent albums and their tracks recorded by 'The Beatles'.*

```
PREFIX dbpedia, foaf, mo, dc;

CONSTRUCT {
  ?album dc:creator dbpedia:The_Beatles .
  ?track dc:creator dbpedia:The_Beatles .}
WHERE {
  dbpedia:The_Beatles foaf:made ?album .
  ?album mo:track ?track ;
         dc:date ?date .
} ORDER BY DESC(?date)
  LIMIT 10
```

Agenda

- SPARQL basics
  - Introductory notions
  - The SELECT query
  - Other query types
- Advanced SPARQL
  - CONSTRUCT queries
  - Filter & Order
  - Aggregate & Group
  - Union
  - Subqueries
  - Federated queries
  - Optional
  - Entailment regimes
- Updates with SPARQL

Solution Modifier: GROUP BY

Aggregates

- Calculate aggregate values: `COUNT`, `SUM`, `MIN`, `MAX`, `AVG`, `GROUP_CONCAT` and `SAMPLE`
- In combination with the `GROUP BY` operator
- Prune at group level (cf. `FILTER`) using `HAVING`

Company	Amount	Year
ACME	$1250	2010
PRIME	$3000	2009
ABC	$2500	2009
ABC	$2800	2010
PRIME	$1950	2010
ACME	$2500	2009
ACME	$3100	2010
ABC	$1500	2009
ACME	$1250	2009
PRIME	$2350	2009
PRIME	$1850	2010

Q: How high are the total sales?

```
SELECT  (SUM(?val) AS ?total)
        WHERE {
            ?s a co:Sale .
            ?s co:amount ?val .
        }
```

```
:row1 a :Sale .
:row1 :company :ACME .
:row1 :amount 1250 .
:row1 :year 2010 .
```

?total
24050

Semantic Web for the Working Ontologist, p101-102

Solution Modifier: GROUP BY

DATA

Company	Amount	Year
ACME	$1250	2010
PRIME	$3000	2009
ABC	$2500	2009
ABC	$2800	2010
PRIME	$1950	2010
ACME	$2500	2009
ACME	$3100	2010
ABC	$1500	2009
ACME	$1250	2009
PRIME	$2350	2009
PRIME	$1850	2010

```
:row1 a :Sale .
:row1 :company :ACME .
:row1 :amount 1250 .
:row1 :year 2010 .
```

Q: How high are the sales per year?

```
SELECT ?year (SUM(?val) AS ?total)
        WHERE {
            ?s a co:Sale .
            ?s co:amount ?val .
            ?s co:year ?year .
        }
        GROUP BY ?year
```

?year	?total
2009	13100
2010	10950

Q: Which companies generated a yearly sales higher than 5K and in which year?

```
SELECT ?year ?company (SUM(?val) AS ?total)
          WHERE {
              ?s a co:Sale .
              ?s co:amount ?val .
              ?s co:year ?year .
              ?s co:company ?company .
          }
          GROUP BY ?year ?company
          HAVING (?total > 5000)
```

HAVING is similar to FILTER but applies to the variables of the groups and is **outside** the graph pattern

?year	?company	?total
2009	PRIME	5350

Semantic Web for the Working Ontologist, p102-103

Solution Modifier: GROUP BY

Query: *Retrieve albums recorded by 'The Beatles' that have a duration higher than a given value.*

```
PREFIX dbpedia, foaf, mo

SELECT ?album (SUM(?track_duration) AS ?album_duration)
        WHERE {
           dbpedia:The_Beatles foaf:made   ?album .
           ?album mo:track    ?track .
           ?track mo:duration ?track_duration .
        } GROUP BY ?album
           HAVING (?album_duration > 3600000)
```

Agenda

- SPARQL basics
  - Introductory notions
  - The SELECT query
  - Other query types
- Advanced SPARQL
  - CONSTRUCT queries
  - Filter & Order
  - Aggregate & Group
  - Union
  - Subqueries
  - Federated queries
  - Optional
  - Entailment regimes
- Updates with SPARQL

SPARQL UNION

- Allows the specification of alternatives (disjunctions)
- UNION = the set union of all solution bindings of the connected patterns

Query: *Which actors played in 'Giant' or 'Rebel without a cause' or both?*

```
PREFIX mo

SELECT ?actor
WHERE {
        {?actor mo:playedIn mo:Giant .}
        UNION
        {?actor mo:playedIn mo:RebelWithoutACause .}
       }
```

actor
Ann Doran
Carroll Baker
Elizabeth Taylor
James Dean
James Dean
Jim Backus
Mercedes McCambridge
Natalie Wood
Rock Hudson
Sal Mineo
Sal Mineo

Semantic Web for the Working Ontologist, p105-106

Agenda

- SPARQL basics
  - Introductory notions
  - The SELECT query
  - Other query types
- Advanced SPARQL
  - CONSTRUCT queries
  - Filter & Order
  - Aggregate & Group
  - Union
  - Subqueries
  - Federated queries
  - Optional
  - Entailment regimes
- Updates with SPARQL

Subqueries

```
PREFIX co

SELECT ?company
WHERE
{
{ SELECT ?company (SUM(?val) AS ?total09)
          WHERE {
             ?s a co:Sale .
             ?s co:amount ?val .
             ?s co:year 2009 .
             ?s co:company ?company .}
          GROUP BY ?company } .
{ SELECT ?company (SUM(?val) AS ?total10)
          WHERE {
             ?s a co:Sale .
             ?s co:amount ?val .
             ?s co:year 2010 .
             ?s co:company ?company .}
          GROUP BY ?company } .
FILTER (?total10 >?total09) .
}
```

Aggregate values can be ”named” using variables (keyword **AS**)

Query:

Select companies for which the total sales value in 2010 is larger than the total sales value in 2009.

?company
ACME

Subqueries

CONSTRUCT and GROUP BY can be combined

```
PREFIX co

CONSTRUCT {?company a co:PreferredCustomer .
           ?company co:totalSales ?total .}
WHERE
{ SELECT ?year ?company (SUM(?val) AS ?total)
           WHERE {
              ?s a co:Sale .
              ?s co:amount ?val .
              ?s co:year ?year .
              ?s co:company ?company .
           }
           GROUP BY ?year ?company
           HAVING (?total > 5000)
}
```

Query:

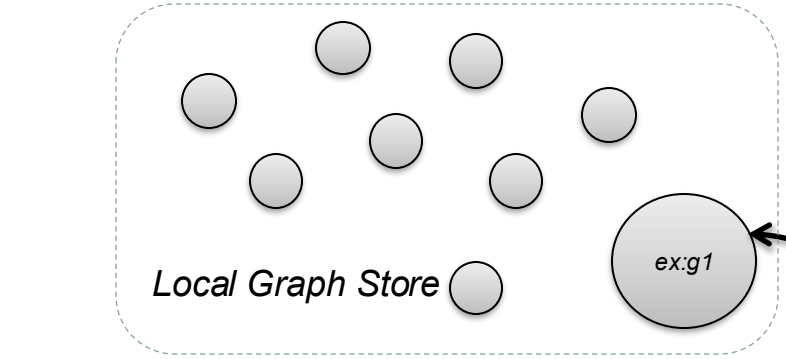
Companies for which the total sales value in any year is larger than 5000 are PreferredCustomers.

```
:PRIME a :PreferredCustomer .
:PRIME :totalSales 5350.00 .
```

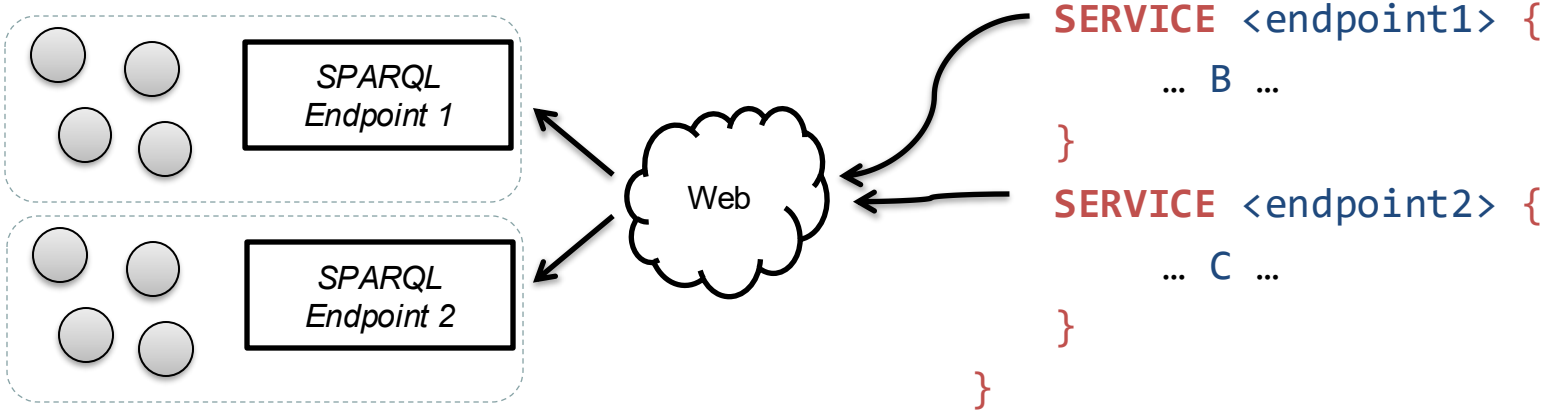
Agenda

- SPARQL basics
  - Introductory notions
  - The SELECT query
  - Other query types
- Advanced SPARQL
  - CONSTRUCT queries
  - Filter & Order
  - Aggregate & Group
  - Union
  - Subqueries
  - Federated queries
  - Optional
  - Entailment regimes
- Updates with SPARQL

Federated Query
(SPARQL 1.1)




```
PREFIX ex: <…>
SELECT …
FROM ex:g1
WHERE {
   … A …
   SERVICE <endpoint1> {
        … B …
   }
   SERVICE <endpoint2> {
        … C …
   }
}
```



https://www.w3.org/TR/sparql11-federated-query/
Slide adapted from: http://www.slideshare.net/LeeFeigenbaum/sparql-cheat-sheet

Federated Query Example

Find the birth dates of all of the actors in Star Trek: The Motion Picture

```
PREFIX movie: <http://data.linkedmdb.org/resource/movie/>
PREFIX dbpedia: <http://dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?actor_name ?birth_date
FROM <http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf> # placeholder graph
WHERE {
 {
  SERVICE <http://data.linkedmdb.org/sparql> {
     <http://data.linkedmdb.org/resource/film/675> movie:actor ?actor .
     ?actor movie:actor_name ?actor_name
   }
   BIND(STRLANG(?actor_name, "en") AS ?actor_name_en)
 }
  SERVICE <http://dbpedia.org/sparql> {
     ?actor2 a foaf:Person ;
             foaf:name ?actor_name_en ;
             dbpedia:birthDate ?birth_date
 }
}
```

?actor_name	?birth_date
"William Shatner"	"1931-03-22"^^<http://www.w3.org/2001/XMLSchema#date>
"Leonard Nimoy"	"1931-03-26"^^<http://www.w3.org/2001/XMLSchema#date>
"DeForest Kelley"	"1920-01-20"^^<http://www.w3.org/2001/XMLSchema#date>
"James Doohan"	"1920-03-03"^^<http://www.w3.org/2001/XMLSchema#date>
"Nichelle Nichols"	"1932-12-28"^^<http://www.w3.org/2001/XMLSchema#date>
"Stephen Collins"	"1947-10-01"^^<http://www.w3.org/2001/XMLSchema#date>
"George Takei"	"1937-04-20"^^<http://www.w3.org/2001/XMLSchema#date>

Example source: http://www.cambridgesemantics.com/semantic-university/sparql-by-example#%2851%29

More Advanced SPARQL

Agenda

- SPARQL basics
  - Introductory notions
  - The SELECT query
  - Other query types
- Advanced SPARQL
  - CONSTRUCT queries
  - Filter & Order
  - Aggregate & Group
  - Union
  - Subqueries
  - Federated queries
  - Optional
  - Entailment regimes
- Updates with SPARQL

OPTIONAL

```
PREFIX ex: <http://ex.com/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?s ?o ?c
WHERE {
  ?s foaf:name ?o.
  OPTIONAL { ?s ex:country ?c }.
}
```

```
ex:Anzhelika%20Sidorova  rdf:type foaf:Person.
ex:Sandi%20Morris        rdf:type foaf:Person.
ex:Katerina%20Stefanidi  rdf:type foaf:Person.
ex:Holly%20Bradshaw      rdf:type foaf:Person.
ex:Alysha%20Newman       rdf:type foaf:Person.
ex:Angelica%20Bengtsson  rdf:type foaf:Person.

ex:Anzhelika%20Sidorova  ex:score "4.95"^^xsd:decimal.
ex:Sandi%20Morris        ex:score "4.90"^^xsd:decimal.
ex:Katerina%20Stefanidi  ex:score "4.85"^^xsd:decimal.
ex:Holly%20Bradshaw      ex:score "4.80"^^xsd:decimal.
ex:Alysha%20Newman       ex:score "4.80"^^xsd:decimal.
ex:Angelica%20Bengtsson  ex:score "4.80"^^xsd:decimal.

ex:Anzhelika%20Sidorova  foaf:name "Anzhelika Sidorova"@en.
ex:Sandi%20Morris        foaf:name "Sandi Morris"@en.
ex:Katerina%20Stefanidi  foaf:name "Katerina Stefanidi"@en.
ex:Holly%20Bradshaw      foaf:name "Holly Bradshaw"@en.
ex:Alysha%20Newman       foaf:name "Alysha Newman"@en.
ex:Angelica%20Bengtsson  foaf:name "Angelica Bengtsson"@en.

ex:Anzhelika%20Sidorova  ex:country <http://ex.com/RU>.
ex:Sandi%20Morris        ex:country <http://ex.com/US>.
ex:Katerina%20Stefanidi  ex:country <http://ex.com/EL>.
ex:Holly%20Bradshaw      ex:country <http://ex.com/UK>.
ex:Alysha%20Newman       ex:country <http://ex.com/CA>.
ex:Angelica%20Bengtsson  ex:country <http://ex.com/SE>.
```

OPTIONAL

```
PREFIX ex: <http://ex.com/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?s ?o ?c
WHERE {
  ?s foaf:name ?o.
  OPTIONAL { ?s ex:country ?c }.
}
```

?s	?o	?c
ex:Alysha%20Newman	"Alysha Newman"@en	ex:CA
ex:Angelica%20Bengtsson	"Angelica Bengtsson"@en	ex:SE
ex:Anzhelika%20Sidorova	"Anzhelika Sidorova"@en	ex:RU
ex:Holly%20Bradshaw	"Holly Bradshaw"@en	ex:UK
ex:Katerina%20Stefanidi	"Katerina Stefanidi"@en	ex:EL
ex:Sandi%20Morris	"Sandi Morris"@en	

```
ex:Anzhelika%20Sidorova rdf:type foaf:Person.
ex:Sandi%20Morris rdf:type foaf:Person.
ex:Katerina%20Stefanidi rdf:type foaf:Person.
ex:Holly%20Bradshaw rdf:type foaf:Person.
ex:Alysha%20Newman rdf:type foaf:Person.
ex:Angelica%20Bengtsson rdf:type foaf:Person.

ex:Anzhelika%20Sidorova ex:score "4.95"^^xsd:decimal.
ex:Sandi%20Morris ex:score "4.90"^^xsd:decimal.
ex:Katerina%20Stefanidi ex:score "4.85"^^xsd:decimal.
ex:Holly%20Bradshaw ex:score "4.80"^^xsd:decimal.
ex:Alysha%20Newman ex:score "4.80"^^xsd:decimal.
ex:Angelica%20Bengtsson ex:score "4.80"^^xsd:decimal.

ex:Anzhelika%20Sidorova foaf:name "Anzhelika Sidorova"@en.
ex:Sandi%20Morris foaf:name "Sandi Morris"@en.
ex:Katerina%20Stefanidi foaf:name "Katerina Stefanidi"@en.
ex:Holly%20Bradshaw foaf:name "Holly Bradshaw"@en.
ex:Alysha%20Newman foaf:name "Alysha Newman"@en.
ex:Angelica%20Bengtsson foaf:name "Angelica Bengtsson"@en.

ex:Anzhelika%20Sidorova ex:country <http://ex.com/RU>.
ex:Sandi%20Morris ex:country <http://ex.com/US>.
ex:Katerina%20Stefanidi ex:country <http://ex.com/EL>.
ex:Holly%20Bradshaw ex:country <http://ex.com/UK>.
ex:Alysha%20Newman ex:country <http://ex.com/CA>.
ex:Angelica%20Bengtsson ex:country <http://ex.com/SE>.
```

Adapted from Anastasia Dimou (course on “Data & Knowledge on the Web“)

OPTIONAL

```
PREFIX ex: <http://ex.com/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?s ?o ?c
WHERE {
  ?s foaf:name ?o.
        ?s ex:country ?c  .
}
```

?s	?o	?c
ex:Alysha%20Newman	"Alysha Newman"@en	ex:CA
ex:Angelica%20Bengtsson	"Angelica Bengtsson"@en	ex:SE
ex:Anzhelika%20Sidorova	"Anzhelika Sidorova"@en	ex:RU
ex:Holly%20Bradshaw	"Holly Bradshaw"@en	ex:UK
ex:Katerina%20Stefanidi	"Katerina Stefanidi"@en	ex:EL
ex:Sandi%20Morris	"Sandi Morris"@en	

```
ex:Anzhelika%20Sidorova rdf:type foaf:Person.
ex:Sandi%20Morris rdf:type foaf:Person.
ex:Katerina%20Stefanidi rdf:type foaf:Person.
ex:Holly%20Bradshaw rdf:type foaf:Person.
ex:Alysha%20Newman rdf:type foaf:Person.
ex:Angelica%20Bengtsson rdf:type foaf:Person.

ex:Anzhelika%20Sidorova ex:score "4.95"^^xsd:decimal.
ex:Sandi%20Morris ex:score "4.90"^^xsd:decimal.
ex:Katerina%20Stefanidi ex:score "4.85"^^xsd:decimal.
ex:Holly%20Bradshaw ex:score "4.80"^^xsd:decimal.
ex:Alysha%20Newman ex:score "4.80"^^xsd:decimal.
ex:Angelica%20Bengtsson ex:score "4.80"^^xsd:decimal.

ex:Anzhelika%20Sidorova foaf:name "Anzhelika Sidorova"@en.
ex:Sandi%20Morris foaf:name "Sandi Morris"@en.
ex:Katerina%20Stefanidi foaf:name "Katerina Stefanidi"@en.
ex:Holly%20Bradshaw foaf:name "Holly Bradshaw"@en.
ex:Alysha%20Newman foaf:name "Alysha Newman"@en.
ex:Angelica%20Bengtsson foaf:name "Angelica Bengtsson"@en.

ex:Anzhelika%20Sidorova ex:country <http://ex.com/RU>.
ex:Sandi%20Morris ex:country <http://ex.com/US>.
ex:Katerina%20Stefanidi ex:country <http://ex.com/EL>.
ex:Holly%20Bradshaw ex:country <http://ex.com/UK>.
ex:Alysha%20Newman ex:country <http://ex.com/CA>.
ex:Angelica%20Bengtsson ex:country <http://ex.com/SE>.
```

OPTIONAL

```
PREFIX ex: <http://ex.com/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?s ?o ?c
WHERE {
  ?s foaf:name ?o.
        ?s ex:country ?c  .
}
```

?s	?o	?c
ex:Alysha%20Newman	"Alysha Newman"@en	ex:CA
ex:Angelica%20Bengtsson	"Angelica Bengtsson"@en	ex:SE
ex:Anzhelika%20Sidorova	"Anzhelika Sidorova"@en	ex:RU
ex:Holly%20Bradshaw	"Holly Bradshaw"@en	ex:UK
ex:Katerina%20Stefanidi	"Katerina Stefanidi"@en	ex:EL

```
ex:Anzhelika%20Sidorova  rdf:type foaf:Person.
ex:Sandi%20Morris        rdf:type foaf:Person.
ex:Katerina%20Stefanidi  rdf:type foaf:Person.
ex:Holly%20Bradshaw      rdf:type foaf:Person.
ex:Alysha%20Newman       rdf:type foaf:Person.
ex:Angelica%20Bengtsson rdf:type foaf:Person.

ex:Anzhelika%20Sidorova  ex:score "4.95"^^xsd:decimal.
ex:Sandi%20Morris        ex:score "4.90"^^xsd:decimal.
ex:Katerina%20Stefanidi  ex:score "4.85"^^xsd:decimal.
ex:Holly%20Bradshaw      ex:score "4.80"^^xsd:decimal.
ex:Alysha%20Newman       ex:score "4.80"^^xsd:decimal.
ex:Angelica%20Bengtsson ex:score "4.80"^^xsd:decimal.

ex:Anzhelika%20Sidorova  foaf:name "Anzhelika Sidorova"@en.
ex:Sandi%20Morris        foaf:name "Sandi Morris"@en.
ex:Katerina%20Stefanidi  foaf:name  "Katerina Stefanidi"@en.
ex:Holly%20Bradshaw      foaf:name "Holly Bradshaw"@en.
ex:Alysha%20Newman       foaf:name "Alysha Newman"@en.
ex:Angelica%20Bengtsson foaf:name "Angelica Bengtsson"@en.

ex:Anzhelika%20Sidorova  ex:country <http://ex.com/RU>.
ex:Sandi%20Morris        ex:country <http://ex.com/US>.
ex:Katerina%20Stefanidi  ex:country <http://ex.com/EL>.
ex:Holly%20Bradshaw      ex:country <http://ex.com/UK>.
ex:Alysha%20Newman       ex:country <http://ex.com/CA>.
ex:Angelica%20Bengtsson ex:country <http://ex.com/SE>.
```

OPTIONAL

```
PREFIX ex: <http://ex.com/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?s ?o ?c ?sc
WHERE { ?s foaf:name ?o.
        OPTIONAL { ?s ex:country ?c }.
        OPTIONAL { ?s ex:score ?sc }.
}
```

```
ex:Anzhelika%20Sidorova  rdf:type foaf:Person.
ex:Sandi%20Morris        rdf:type foaf:Person.
ex:Katerina%20Stefanidi  rdf:type foaf:Person.
ex:Holly%20Bradshaw      rdf:type foaf:Person.
ex:Alysha%20Newman       rdf:type foaf:Person.
ex:Angelica%20Bengtsson  rdf:type foaf:Person.

ex:Anzhelika%20Sidorova  ex:score "4.95"^^xsd:decimal.
ex:Sandi%20Morris        ex:score "4.90"^^xsd:decimal.
ex:Katerina%20Stefanidi  ex:score "4.85"^^xsd:decimal.
ex:Holly%20Bradshaw      ex:score "4.80"^^xsd:decimal.
ex:Alysha%20Newman       ex:score "4.80"^^xsd:decimal.
ex:Angelica%20Bengtsson  ex:score "4.80"^^xsd:decimal.

ex:Anzhelika%20Sidorova  foaf:name "Anzhelika Sidorova"@en.
ex:Sandi%20Morris        foaf:name "Sandi Morris"@en.
ex:Katerina%20Stefanidi  foaf:name "Katerina Stefanidi"@en.
ex:Holly%20Bradshaw      foaf:name "Holly Bradshaw"@en.
ex:Alysha%20Newman       foaf:name "Alysha Newman"@en.
ex:Angelica%20Bengtsson  foaf:name "Angelica Bengtsson"@en.

ex:Anzhelika%20Sidorova  ex:country <http://ex.com/RU>.
ex:Sandi%20Morris        ex:country <http://ex.com/US>.
ex:Katerina%20Stefanidi  ex:country <http://ex.com/EL>.
ex:Holly%20Bradshaw      ex:country <http://ex.com/UK>.
ex:Alysha%20Newman       ex:country <http://ex.com/CA>.
ex:Angelica%20Bengtsson  ex:country <http://ex.com/SE>.
```

TU WIEN Informatics

OPTIONAL

```
PREFIX ex: <http://ex.com/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?s ?o ?c ?sc
WHERE { ?s foaf:name ?o.
        OPTIONAL { ?s ex:country ?c }.
        OPTIONAL { ?s ex:score ?sc }.
}
```

?s	?o	?c	?sc
ex:Anzhelika%20Sidorova	"Anzhelika Sidorova"@en	ex:RU	"4.95"^^xsd:decimal
ex:Sandi%20Morris	"Sandi Morris"@en		"4.90"^^xsd:decimal
ex:Katerina%20Stefanidi	"Katerina Stefanidi"@en	ex:EL	"4.85"^^xsd:decimal
ex:Holly%20Bradshaw	"Holly Bradshaw"@en	ex:UK	"4.80"^^xsd:decimal
ex:Alysha%20Newman	"Alysha Newman"@en	ex:CA	"4.80"^^xsd:decimal
ex:Angelica%20Bengtsson	"Angelica Bengtsson"@en	ex:SE	"4.80"^^xsd:decimal

```
ex:Anzhelika%20Sidorova rdf:type foaf:Person.
ex:Sandi%20Morris rdf:type foaf:Person.
ex:Katerina%20Stefanidi rdf:type foaf:Person.
ex:Holly%20Bradshaw rdf:type foaf:Person.
ex:Alysha%20Newman rdf:type foaf:Person.
ex:Angelica%20Bengtsson rdf:type foaf:Person.

ex:Anzhelika%20Sidorova ex:score "4.95"^^xsd:decimal.
ex:Sandi%20Morris ex:score "4.90"^^xsd:decimal.
ex:Katerina%20Stefanidi ex:score "4.85"^^xsd:decimal.
ex:Holly%20Bradshaw ex:score "4.80"^^xsd:decimal.
ex:Alysha%20Newman ex:score "4.80"^^xsd:decimal.
ex:Angelica%20Bengtsson ex:score "4.80"^^xsd:decimal.

ex:Anzhelika%20Sidorova foaf:name "Anzhelika Sidorova
ex:Sandi%20Morris foaf:name "Sandi Morris"@en.
ex:Katerina%20Stefanidi foaf:name "Katerina Stefanidi"
ex:Holly%20Bradshaw foaf:name "Holly Bradshaw"@
ex:Alysha%20Newman foaf:name "Alysha Newman"@
ex:Angelica%20Bengtsson foaf:name "Angelica Bengtsson

ex:Anzhelika%20Sidorova ex:country <http://ex.com/RU>
ex:Sandi%20Morris ex:country <http://ex.com/US>
ex:Katerina%20Stefanidi ex:country <http://ex.com/EL>
ex:Holly%20Bradshaw ex:country <http://ex.com/UK>
ex:Alysha%20Newman ex:country <http://ex.com/CA>
ex:Angelica%20Bengtsson ex:country <http://ex.com/SE>
```

TU WIEN Informatics

OPTIONAL

```
PREFIX ex: <http://ex.com/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?s ?o ?c ?sc
WHERE { ?s foaf:name ?o.
        OPTIONAL { ?s ex:country ?c .
                   ?s ex:score ?sc }.
}
```

?s	?o	?c	?sc
ex:Anzhelika%20Sidorova	"Anzhelika Sidorova"@en	ex:RU	"4.95"^^xsd:decimal
ex:Sandi%20Morris	"Sandi Morris"@en		"4.90"^^xsd:decimal
ex:Katerina%20Stefanidi	"Katerina Stefanidi"@en	ex:EL	"4.85"^^xsd:decimal
ex:Holly%20Bradshaw	"Holly Bradshaw"@en	ex:UK	"4.80"^^xsd:decimal
ex:Alysha%20Newman	"Alysha Newman"@en	ex:CA	"4.80"^^xsd:decimal
ex:Angelica%20Bengtsson	"Angelica Bengtsson"@en	ex:SE	"4.80"^^xsd:decimal

```
ex:Anzhelika%20Sidorova  rdf:type foaf:Person.
ex:Sandi%20Morris        rdf:type foaf:Person.
ex:Katerina%20Stefanidi  rdf:type foaf:Person.
ex:Holly%20Bradshaw      rdf:type foaf:Person.
ex:Alysha%20Newman       rdf:type foaf:Person.
ex:Angelica%20Bengtsson rdf:type foaf:Person.

ex:Anzhelika%20Sidorova  ex:score "4.95"^^xsd:decimal.
ex:Sandi%20Morris        ex:score "4.90"^^xsd:decimal.
ex:Katerina%20Stefanidi  ex:score "4.85"^^xsd:decimal.
ex:Holly%20Bradshaw      ex:score "4.80"^^xsd:decimal.
ex:Alysha%20Newman       ex:score "4.80"^^xsd:decimal.
ex:Angelica%20Bengtsson ex:score "4.80"^^xsd:decimal.

ex:Anzhelika%20Sidorova  foaf:name "Anzhelika Sidorova
ex:Sandi%20Morris        foaf:name "Sandi Morris"@en.
ex:Katerina%20Stefanidi  foaf:name  "Katerina Stefanidi"
ex:Holly%20Bradshaw      foaf:name "Holly Bradshaw"@
ex:Alysha%20Newman       foaf:name "Alysha Newman"@
ex:Angelica%20Bengtsson foaf:name "Angelica Bengtsson

ex:Anzhelika%20Sidorova  ex:country <http://ex.com/RU>
ex:Sandi%20Morris        ex:country <http://ex.com/US>
ex:Katerina%20Stefanidi  ex:country <http://ex.com/EL>
ex:Holly%20Bradshaw      ex:country <http://ex.com/UK>
ex:Alysha%20Newman       ex:country <http://ex.com/CA>
ex:Angelica%20Bengtsson ex:country <http://ex.com/SE>
```

TU WIEN Informatics

OPTIONAL

```
PREFIX ex: <http://ex.com/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?s ?o ?c ?sc
WHERE { ?s foaf:name ?o.
        OPTIONAL { ?s ex:country ?c .
                   ?s ex:score ?sc }.
}
```

?s	?o	?c	?sc
ex:Anzhelika%20Sidorova	"Anzhelika Sidorova"@en	ex:RU	"4.95"^^xsd:decimal
ex:Sandi%20Morris	"Sandi Morris"@en		
ex:Katerina%20Stefanidi	"Katerina Stefanidi"@en	ex:EL	"4.85"^^xsd:decimal
ex:Holly%20Bradshaw	"Holly Bradshaw"@en	ex:UK	"4.80"^^xsd:decimal
ex:Alysha%20Newman	"Alysha Newman"@en	ex:CA	"4.80"^^xsd:decimal
ex:Angelica%20Bengtsson	"Angelica Bengtsson"@en	ex:SE	"4.80"^^xsd:decimal

```
ex:Anzhelika%20Sidorova  rdf:type foaf:Person.
ex:Sandi%20Morris        rdf:type foaf:Person.
ex:Katerina%20Stefanidi  rdf:type foaf:Person.
ex:Holly%20Bradshaw      rdf:type foaf:Person.
ex:Alysha%20Newman       rdf:type foaf:Person.
ex:Angelica%20Bengtsson rdf:type foaf:Person.

ex:Anzhelika%20Sidorova  ex:score "4.95"^^xsd:decimal.
ex:Sandi%20Morris        ex:score "4.90"^^xsd:decimal.
ex:Katerina%20Stefanidi  ex:score "4.85"^^xsd:decimal.
ex:Holly%20Bradshaw      ex:score "4.80"^^xsd:decimal.
ex:Alysha%20Newman       ex:score "4.80"^^xsd:decimal.
ex:Angelica%20Bengtsson ex:score "4.80"^^xsd:decimal.

ex:Anzhelika%20Sidorova  foaf:name "Anzhelika Sidorova
ex:Sandi%20Morris        foaf:name "Sandi Morris"@en.
ex:Katerina%20Stefanidi  foaf:name  "Katerina Stefanidi"
ex:Holly%20Bradshaw      foaf:name "Holly Bradshaw"@
ex:Alysha%20Newman       foaf:name "Alysha Newman"@
ex:Angelica%20Bengtsson foaf:name "Angelica Bengtsson

ex:Anzhelika%20Sidorova  ex:country <http://ex.com/RU>
ex:Sandi%20Morris        ex:country <http://ex.com/US>
ex:Katerina%20Stefanidi  ex:country <http://ex.com/EL>
ex:Holly%20Bradshaw      ex:country <http://ex.com/UK>
ex:Alysha%20Newman       ex:country <http://ex.com/CA>
ex:Angelica%20Bengtsson ex:country <http://ex.com/SE>
```

Agenda

- SPARQL basics
  - Introductory notions
  - The SELECT query
  - Other query types
- Advanced SPARQL
  - CONSTRUCT queries
  - Filter & Order
  - Aggregate & Group
  - Union
  - Subqueries
  - Federated queries
  - Optional
  - Entailment regimes
- Updates with SPARQL

Inference from Schema during Querying

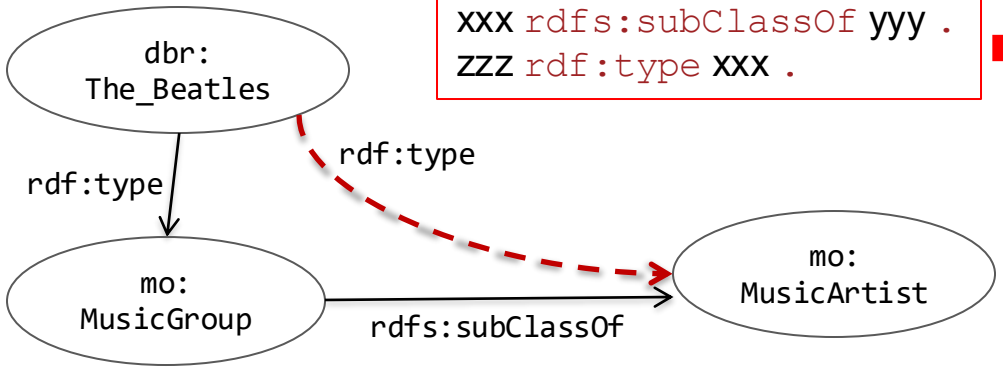
- A triple store can use the knowledge encoded in the schema to infer new facts

Schema:
```
mo:MusicGroup rdfs:subClassOf mo:MusicArtist .
```

Inferred facts:
```
mo:MusicGroup  a rdfs:Class .
mo:MusicArtist a rdfs:Class .
```

- Whether a triple store supports reasoning, influences the outcome of querying

Reasoning with – rdfs:subClassOf




```
xxx rdfs:subClassOf yyy .
zzz rdf:type xxx .
```

→

```
zzz rdf:type yyy .
```

Schema:

```
mo:MusicGroup rdfs:subClassOf mo:MusicArtist .
```

Query:

```
SELECT ?x
WHERE {?x a mo:MusicArtist.}
```

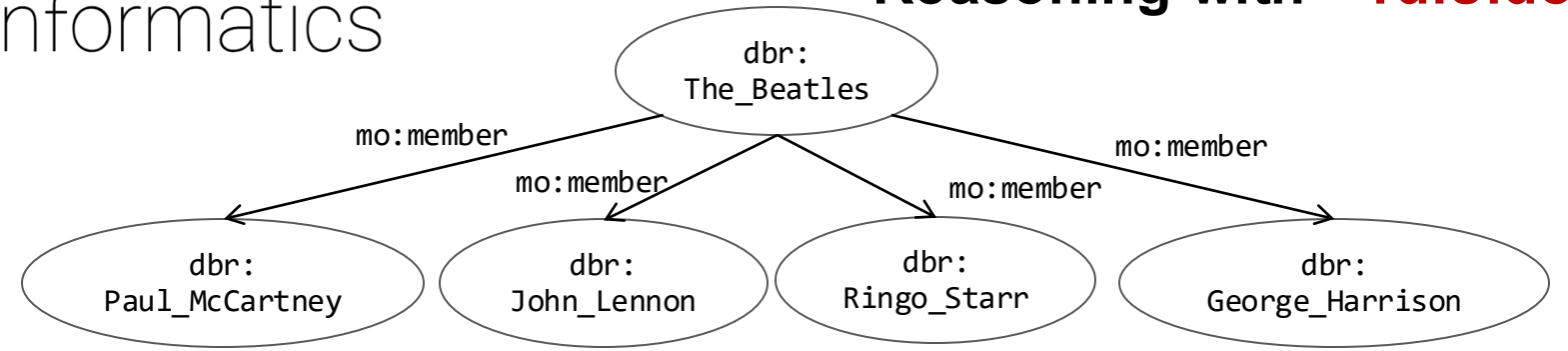
Result set with inference:

?x
dbr:The_Beatles …

Result set without inference:

?x

Reasoning with – rdfs:domain



Schema:

```
mo:member rdfs:domain mo:MusicGroup .
```

Query:

```
SELECT ?x
WHERE {?x a mo:MusicGroup.}
```

Result set with inference:

?x
dbr:The_Beatles …

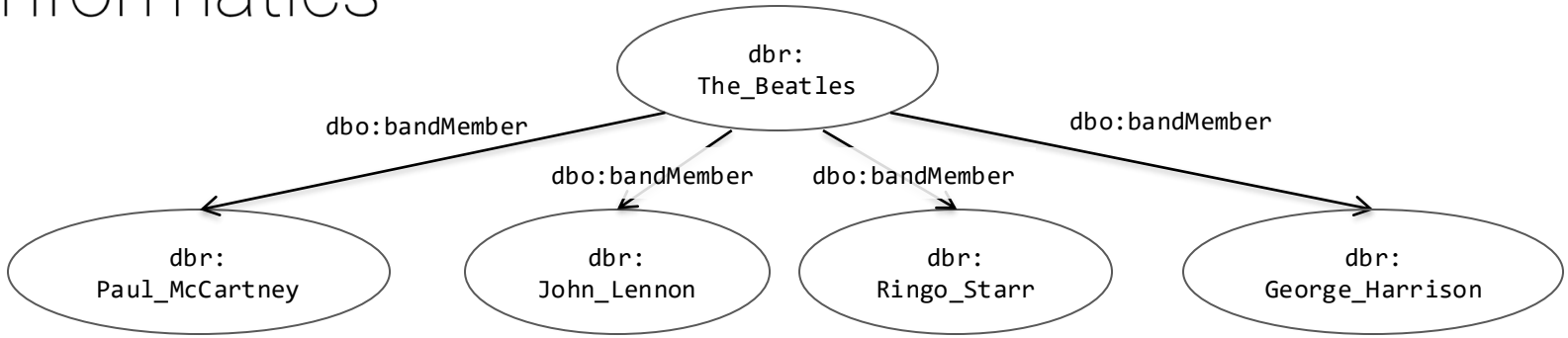
Result set without interference:

?x

aaa `rdfs:domain` xxx .
yyy aaa zzz .

→ yyy `rdf:type` xxx .

Reasoning with – rdfs:range



Schema:

```
dbo:bandMember rdfs:range foaf:Musician .
```

Query:

```
SELECT ?x
WHERE {?x a foaf:Musician .}
```

Result set with inference:

?x
dbr:Paul_McCartney
dbr:John_Lennon
dbr:Ringo_Starr
dbr:George_Harrison …

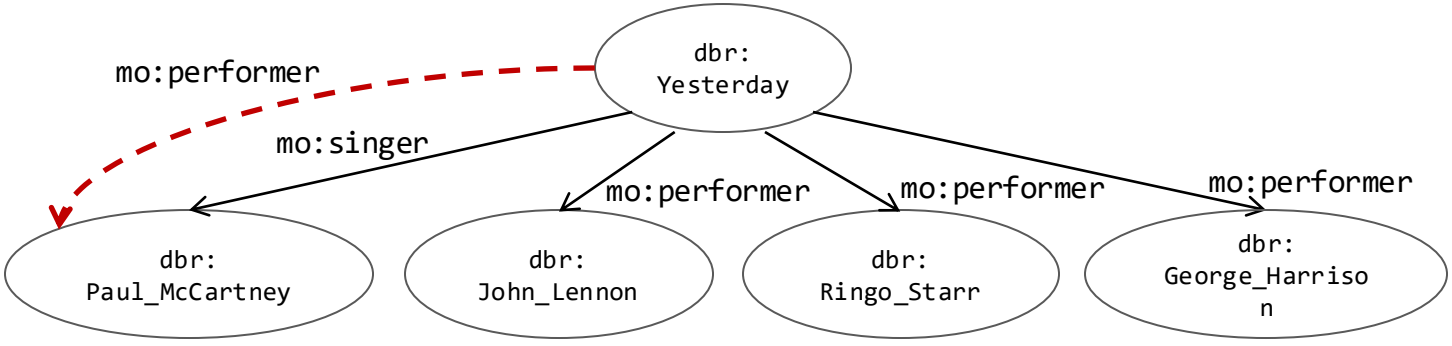
Result set without inference:

?x

aaa `rdfs:range` xxx .
yyy aaa zzz .

→ zzz `rdf:type` xxx .

Reasoning with – rdfs:subPropertyOf



Schema:

```
mo:singer rdfs:subPropertyOf mo:performer .
```

Query:

```
SELECT ?x
WHERE {dbr:Yesterday mo:performer ?x .}
```

Result set with inference:

?x
dbr:John_Lennon
dbr:Ringo_Starr
dbr:George_Harrison
dbr:Paul_McCartney

Result set without inference:

?x
dbr:John_Lennon
dbr:Ringo_Starr
dbr:George_Harrison

aaa `rdfs:subPropertyOf` bbb .
yyy aaa zzz .

→ yyy bbb zzz .

RDFS Entailment Patterns

	If S contains:	then S RDFS entails recognizing D:
rdfs1	any IRI aaa in D	aaa `rdf:type rdfs:Datatype .`
rdfs2	aaa `rdfs:domain` xxx . yyy aaa zzz .	yyy `rdf:type` xxx .
rdfs3	aaa `rdfs:range` xxx . yyy aaa zzz .	zzz `rdf:type` xxx .
rdfs4a	xxx aaa yyy .	xxx `rdf:type rdfs:Resource .`
rdfs4b	xxx aaa yyy.	yyy `rdf:type rdfs:Resource .`
rdfs5	xxx `rdfs:subPropertyOf` yyy . yyy `rdfs:subPropertyOf` zzz .	xxx `rdfs:subPropertyOf` zzz .
rdfs6	xxx `rdf:type rdf:Property .`	xxx `rdfs:subPropertyOf` xxx .
rdfs7	aaa `rdfs:subPropertyOf` bbb . xxx aaa yyy .	xxx bbb yyy .
rdfs8	xxx `rdf:type rdfs:Class .`	xxx `rdfs:subClassOf rdfs:Resource .`
rdfs9	xxx `rdfs:subClassOf` yyy . zzz `rdf:type` xxx .	zzz `rdf:type` yyy .
rdfs10	xxx `rdf:type rdfs:Class .`	xxx `rdfs:subClassOf` xxx .
rdfs11	xxx `rdfs:subClassOf` yyy . yyy `rdfs:subClassOf` zzz .	xxx `rdfs:subClassOf` zzz .
rdfs12	xxx `rdf:type rdfs:ContainerMembershipProperty .`	xxx `rdfs:subPropertyOf rdfs:member .`
rdfs13	xxx `rdf:type rdfs:Datatype .`	xxx `rdfs:subClassOf rdfs:Literal .`

S.. subgraph D.. Set of IRIs identifying datatypes

Source: https://www.w3.org/TR/2014/REC-rdf11-mt-20140225/#rdfs-interpretations

Updating RDF with SPARQL 1.1

SPARQL Query 1.1

- SPARQL 1.0 only allows data access (querying)
- SPARQL 1.1 introduces:

Query extensions

Aggregates, subqueries, negation, expressions in the `SELECT` clause, property paths, assignment, expanded set of functions and operators

Updates

`Insert, Delete, Delete/Insert`

`Create, Load, Clear, Drop`
`Copy, Move, Add`

Federation extension

Service, values, service variables (informative)

SPARQL Updates

SPARQL 1.1 provides data update operations

- `INSERT (DATA)`: adds triples, given inline in the request, into a graph
- `DELETE (DATA)`: removes triples, given inline in the request, if the respective graphs contains those
- `DELETE/INSERT (DATA)`: uses in parallel `INSERT` and `DELETE`

SPARQL INSERT

INSERT + GP that specifies a set of RDF triples, given bindings for any variables that are included in the pattern

Insert "Peter Best" as a `formerMember` of `dbr:The_Beatles`

```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

INSERT { dbr:The_Beatles dbr:formerMember ?x }
WHERE  { dbr:The_Beatles dbr:currentMember ?x .
         ?x foaf:name "Peter Best"
       }
```

SPARQL INSERT

Insert the following albums recorded by The Beatles into the graph <http://musicbrainz.org/20130209-004702>

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

INSERT DATA {
        GRAPH { <http://musicbrainz.org/20130309-004702>
<http://musicbrainz.org/artist/b10bbbfc-cf9e-42e0-be17-e2c3e1d2600d>
    foaf:made <http://musicbrainz.org/release/3a685770-7326-34fc-9f18-e5f5626f3dc5>
,
         <http://musicbrainz.org/release/cb6f8798-d51e-4fa5-a4d1-2c0602bfe1b6> .

<http://musicbrainz.org/release/3a685770-7326-34fc-9f18-e5f5626f3dc5>
        dc:title "Please Please Me".

<http://musicbrainz.org/release/cb6f8798-d51e-4fa5-a4d1-2c0602bfe1b6>
        dc:title "Something New". } }
```

SPARQL DELETE

__Delete__ all the information about the album "Casualities" of The Beatles.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

DELETE { ?album ?predicate ?object . }
WHERE {
        <http://musicbrainz.org/artist/b10bbbfc-cf9e-42e0-be17-e2c3e1d2600d>
                foaf:made ?album .
        ?album dc:title "Casualities".
        ?album ?predicate ?object .
}
```

SPARQL DELETE/INSERT

Delete the status of 'Peter Best' as current member of "The Beatles" and **insert** his status as former member of the band.

```
PREFIX dbr, dbo, foaf …

DELETE { dbr:The_Beatles dbo:currentMember ?x . }
INSERT { GRAPH <http://musicbrainz.org/20130209-004702> {
          dbr:The_Beatles dbo:formerBandMember ?x .}
       }
WHERE {
    dbr:The_Beatles dbo:currentMember ?x .
    ?x foaf:name "Peter Best" .}
```

SPARQL 1.1 Graph Update Operations

- SPARQL 1.1 provides graph update operations:
  - CREATE creates an empty graph in the graph store
  - LOAD reads the content of a document into a graph in the graph store
  - CLEAR removes all triples in one or more graphs
  - DROP removes the graph from the graph store
  - Other operations: COPY, MOVE, ADD

SPARQL CREATE

- Creates a new named graph
- Can be used with the **DEFAULT** and **ALL** keywords

```
CREATE GRAPH <http://musicbrainz.org/20130302>
```

- **LOAD**: An RDF graph can be loaded from a URL

```
LOAD <http://xmlns.com/foaf/spec/20100809.rdf>
```

```
LOAD <http://xmlns.com/foaf/spec/20100809.rdf>
        INTO <http://xmlns.com/foaf/0.1/>
```

Named graph

- **LOAD** can be used with the **SILENT** keyword (suppresses errors)

SPARQL CLEAR and DROP

```
CLEAR GRAPH <http://musicbrainz.org/20130302>
```

- **CLEAR** removes all triples in the graph (emptied, but not deleted!)
- The graph(s) can be specified with the following keywords: `DEFAULT`, `NAMED`, `ALL`, `GRAPH`
- Can be used with the `SILENT` keyword

```
DROP GRAPH <http://musicbrainz.org/20130302>
```

- **DROP** removes the given graph from the graph store, including its content
- Can be used with the `DEFAULT` and `ALL` keywords

SPARQL COPY, MOVE, ADD

- COPY … TO …

```
COPY GRAPH <http://musicbrainz.org/20130302>
TO GRAPH <http://musicbrainz.org/20130303>
```

- MOVE … TO …

```
MOVE GRAPH <http://musicbrainz.org/temp>
TO GRAPH <http://musicbrainz.org/20130303>
```

- ADD … TO …

```
ADD GRAPH <http://musicbrainz.org/20130302>
TO GRAPH <http://musicbrainz.org/20130303>
```