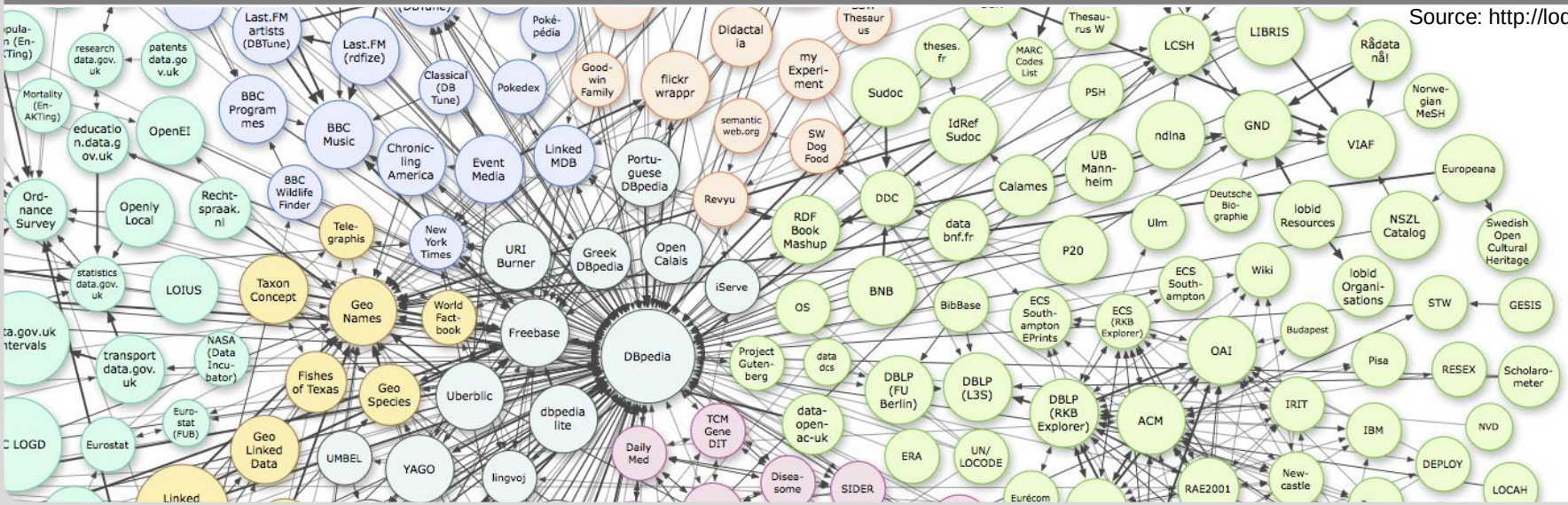
Distributed Knowledge Graphs: SPARQL

Dr. Tobias Käfer

AI4INDUSTRY SUMMER SCHOOL



CC - Creative Commons Licensing

- The slides were compiled by Tobias Käfer, prepared by Lars Heling based on Andreas Harth's input, with major modifications by Maribel Acosta

-

Agenda

1. **Introduction**
2. Structure of SPARQL Queries
3. Basic Graph Patterns
4. Querying Multiple (Named) RDF Graphs

Example Question

"What are the boroughs of Berlin?"

- How can we answer this question over RDF data?

Retrieving Data from a Dataset

- How to retrieve data from a dataset?
  - Queries are used in order to retrieve *relevant* data from a dataset

- Relational databases:
  - A set of tuples is stored in a table (Relation)
  - **S**tructured **Q**uery **L**anguage (SQL)

Relation: Cities

Name	Population	BoroughOf
Oststadt	21 091	Karlsruhe
Pankow	384 367	Berlin
…	…	…

```
SELECT Name
FROM Cities
WHERE BoroughOf = “Berlin” ;
```

- Graph databases:
  - What is a dataset in RDF?
  - How can we query data represented in RDF?

RDF Datasets

- A collection of graphs is called an RDF dataset.
- An RDF dataset has one default graph without a name,

and

- zero or more graphs with a name (a URI)

SPARQL

- Acronym:
  - **S**PARQL **P**rotocol **A**nd **R**DF **Q**uery **L**anguage
- Specified by W3C
  - Current version: SPARQL 1.1 [1] (March 2013)
- There are eleven SPARQL Recommendations, covering:
  - Syntax and semantics of queries over RDF
  - Protocol to pose queries against a SPARQL endpoint and to retrieve results
  - Various serialisations of query results
  - Entailment regimes
  - Update language
  - Federated query
  - …

1 http://www.w3.org/TR/sparql11-overview/

Agenda

1. Introduction
2. **Structure of SPARQL Queries**
3. Basic Graph Patterns
4. Querying Multiple (Named) RDF Graphs

Back to Our Question

"What are the boroughs of Berlin?"

```
PREFIX ex: <http://example.org/cities.ttl#>

SELECT ?borough
FROM <http://example.org/cities.ttl>
WHERE {
       (Some conditions)
}
```

Components of SPARQL Queries (1)

```
PREFIX ex: <http://example.org/cities.ttl#>

SELECT ?borough
FROM <http://example.org/cities.ttl>
WHERE {
       (Some conditions)
}
```

Prefix definitions:

- `PREFIX` keyword to introduce CURIEs
- Subtly different from Turtle syntax
  - The final period is not used
  - No “@” at the beginning

Components of SPARQL Queries (2)

```
PREFIX ex: <http://example.org/cities.ttl#>

SELECT ?borough
FROM <http://example.org/cities.ttl>
WHERE {
       (Some conditions)
}
```

Query form:

- ASK, SELECT, DESCRIBE, or CONSTRUCT
- Details in a bit…

Components of SPARQL Queries (3)

```
PREFIX ex: <http://example.org/cities.ttl#>

SELECT ?borough
FROM <http://example.org/cities.ttl>
WHERE {
       (Some conditions)
}
```

Variable projection:

- Variables are “placeholders” for RDF terms
- Variables are prefixed using “`?`” or “`$`”
- To select all variables contained in a query: “`SELECT * `“

Components of SPARQL Queries (4)

```
PREFIX ex: <http://example.org/cities.ttl#>

SELECT ?borough
FROM <http://example.org/cities.ttl>
WHERE {
       (Some conditions)
}
```

Dataset selection:

- `FROM` or `FROM NAMED` keyword to specify the RDF dataset
- Indicates the sources for the data against which to find matches

Components of SPARQL Queries (5)

```
PREFIX ex: <http://example.org/cities.ttl#>

SELECT ?borough
FROM <http://example.org/cities.ttl>
WHERE {
      (Some condition)
}
```

Query pattern:

- Specifies *what* we want to query
- Contains graph patterns that are matched against RDF data

Components of SPARQL Queries (6)

```
PREFIX ex: <http://example.org/cities.ttl#>

SELECT ?borough
FROM <http://example.org/cities.ttl>
WHERE {
       (Some condition)
} ORDER BY ?borough
```

Sequence modifiers:

- Modify the result set (query answers)
- `ORDER BY` changes the order of the result set
- `LIMIT, OFFSET` selects chunks of the result set
- `DISTINCT` (after `SELECT`), removes duplicate answers

Query Forms

- There are four different query forms that SPARQL supports:
  - `SELECT`

    Return all or a subset of the solution mappings

  - `CONSTRUCT`

    Return a set of triples/a graph, where the mappings are filled into a specific graph pattern template

  - `ASK`

    Return true or false, depending on whether there is a solution mapping or graph pattern

  - `DESCRIBE`

    Return a set of triples / a graph that describes a certain resource (URI)

Agenda

1. Introduction
2. Structure of SPARQL Queries
3. **Basic Graph Patterns**
4. Querying Multiple (Named) RDF Graphs

Triple Patterns

- Building block of SPARQL queries: **triple patterns.**
  - Similar to RDF triples but with variables (specified with `?` or `$`).

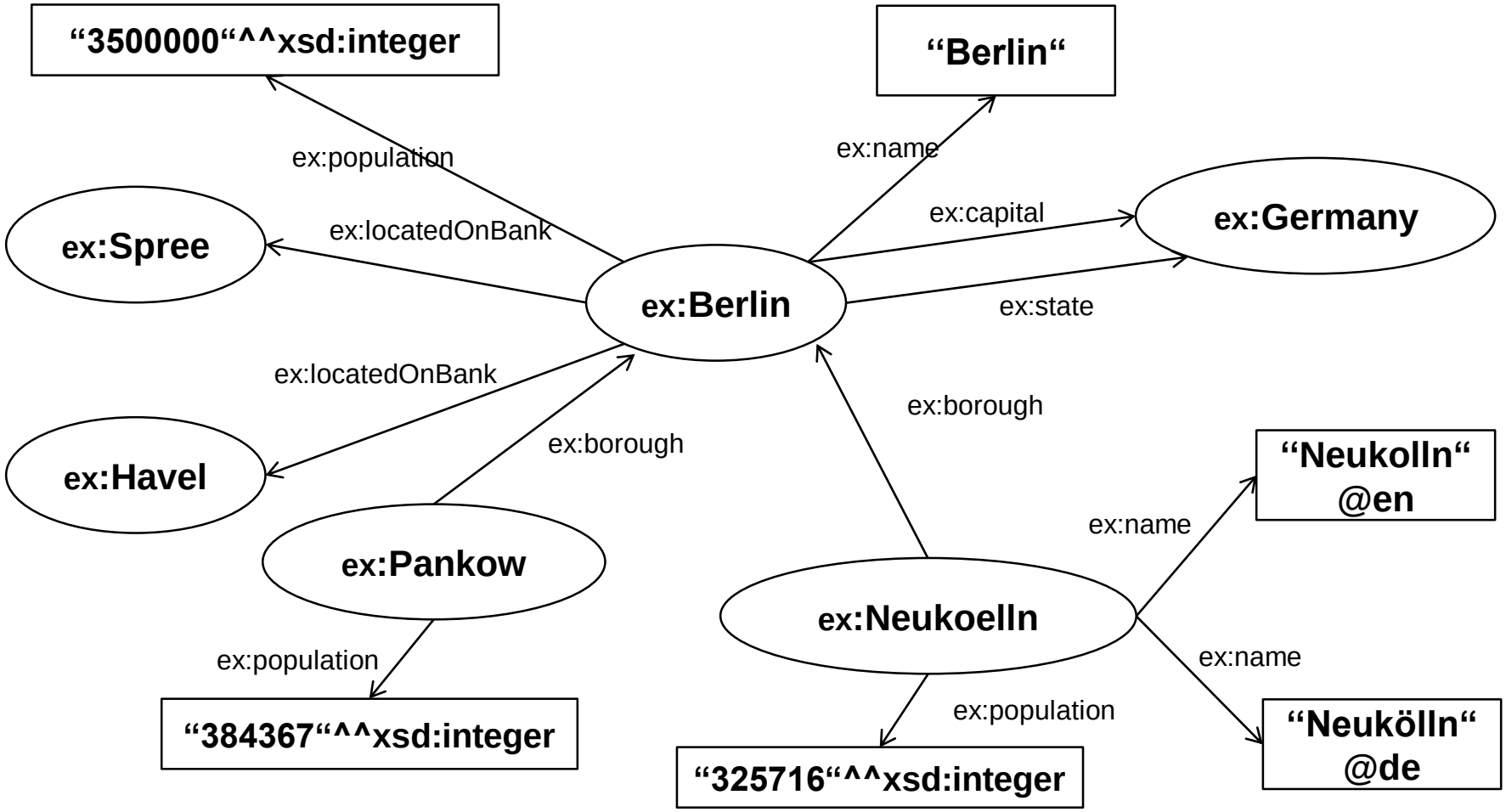
- **Example:** Berlin is the capital of _____.

Or:

```
ex:Berlin ex:capital ?x .
```

http://example.org/cities.ttl

"What are the boroughs of Berlin?"



http://example.org/cities.ttl

"What are the boroughs of Berlin?"

```
{
  ?berlin ex:name "Berlin" .
  ?borough ex:borough ?berlin .
}
```

Basic Graph Pattern (1)

- Basic Graph Pattern (BGP) contains several triple patterns.
- BGPs represent *conjunction* of triple patterns.

- **Example:** The following BGP obtains the boroughs of `ex:Berlin` **and** the population of the boroughs

```
{
        ?borough ex:borough    ex:Berlin .
        ?borough ex:population ?population .
}
```

A variable may be used on the subject, predicate or object position

Basic Graph Pattern (2)

- BGPs can be specified using Turtle syntax
  - Example:

```
{ ?borough ex:borough        ?berlin ;
           ex:population     ?population .
  ?berlin  ex:name           “Berlin” . }
```

- In BGPs blank nodes are treated similar to variables.
  - Example:

```
{  _:bn1   ex:name       ?name .
   _:bn1   ex:population ?population . }
```

  - But: blank nodes may only appear on subject and object position of a triple pattern.

- In contrast to variables, one may not specify blank nodes in the query form (e.g., `SELECT`)

Exercise

Write a SPARQL query into a file query.rq against the following RDF graph to *retrieve all systems* from your file production.ttl , which contains:

```
@prefix : <http://example.org/#> .
:myProductionSystem a :System ;
  :hasSubSystem :roboticArm1 , :conveyorBelt2 .
:roboticArm1 a :System , :RoboticArm ;
  :hasManufacturer :ABB .
:conveyorBelt2 a :System ;
  :hasSpeed "0.1" .
```

- Use roqet to evaluate your query:

```
roqet query.rq # if you use the FROM part
roqet -D production.ttl query.rq # if you don’t
```

- Step 2: update the query to also return their subsystems

Solution

```
PREFIX : <http://example.org/#>

SELECT ?thing
FROM <production.ttl>
WHERE {
  ?thing a :System .
}
```

Solution

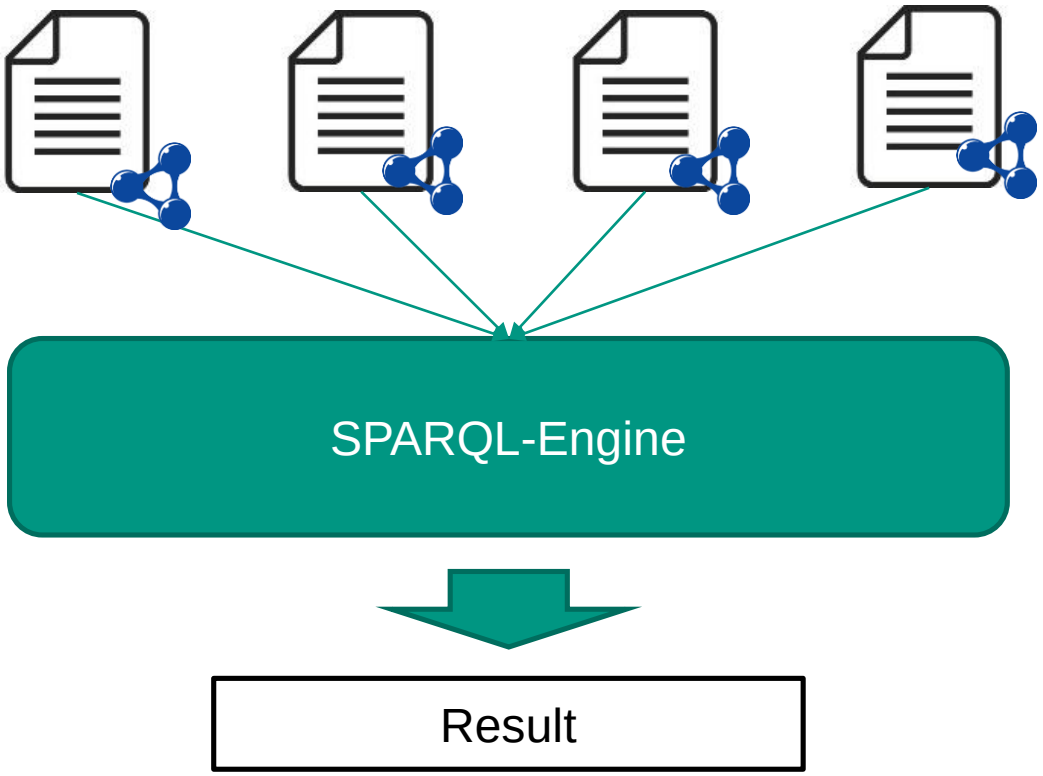
```
PREFIX : <http://example.org/#>

SELECT *
FROM <production.ttl>
WHERE {
  ?thing a :System ; :hasSubSystem ?sub .
}
```

Agenda

1. Introduction
2. Structure of SPARQL Queries
3. Basic Graph Patterns
4. **Querying Multiple (Named) RDF Graphs**

Multiple Graphs

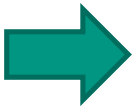
- Information may be spread over several documents
- Therefore, several documents should be addressable in a query



Multiple Graphs

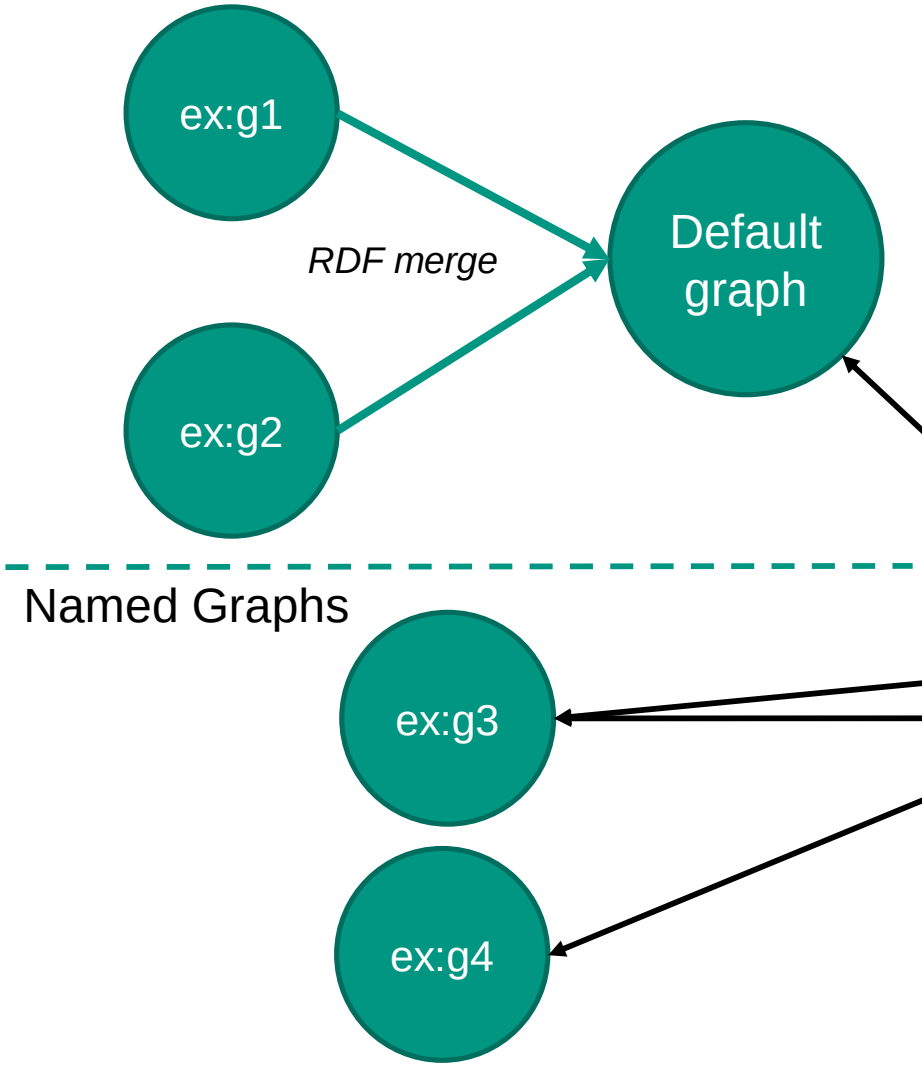
- SPARQL supports handling multiple graphs:
  - These graphs may be different data sources
  - Graphs can be added using the `FROM` keyword
  - All graphs specified in the `FROM` clause are combined to a default graph

- SPARQL supports handling of multiple **named** graphs:
  - Using the `FROM NAMED` keyword
  - These graphs can be accessed using the `GRAPH` keyword
  - Used to query data from specific graphs



To identify the triples belonging to a graph data we extend the triple model to quadruples, to be able to hold information on the context (name of the graph).

Multiple Graphs - Example




```
PREFIX ex : <...>
SELECT *
FROM ex:g1
FROM ex:g2
FROM NAMED ex:g3
FROM NAMED ex:g4
WHERE
{
…
?s      ?p      ?o.
GRAPH ex:g3 { ... }
GRAPH ?graph { ... }
}
```

SPARQL Query Processors vs. SPARQL Endpoints

Query Processor

- Acts as user agent
- Graphs are retrieved via HTTP during query processing
- Default graph is empty, so queries require FROM/FROM NAMED clauses

Endpoint

- Acts as server
- Graphs are indexed and stored on disk during installation (like a database)
- Default graph is configured, so no FROM/FROM NAMED clauses needed

Overview of Core SPARQL Features

- Basic concepts: Triple patterns
- SPARQL Query structure:
  - Prefix declarations: **`PREFIX`**
  - Query forms: `ASK`, **`SELECT`**, `DESCRIBE`, `CONSTRUCT`
  - Variable projection: Subset of variables that we want to return
  - Dataset selection: **`FROM`**, **`FROM NAMED`**
  - Query patterns
    - **Basic Graph Patterns (BGP)**
    - Graph Patterns (`UNION`, `OPTIONAL`, `GRAPH`)
    - Functions (`FILTER`, `BIND AS`)
  - Sequence modifiers: `ORDER BY`, `LIMIT`, `OFFSET`, `DISTINCT`