Classical DB Questions on New Kinds of Data

Marcelo Arenas[‡] & Pablo Barceló[*]

[‡,*]Center for Semantic Web Research
[‡]PUC Chile
[*]Universidad de Chile

Evolution of data models

80s $\rightarrow$ early 90s

Overcome expressiveness limitations of the relational model

Evolution of data models

80s $\rightarrow$ early 90s

Overcome expressiveness limitations of the relational model

- Logical data model
- OODB
- Active databases
- Disjunctive databases
- Temporal databases
- Constraint databases
- ...

Evolution of data models

The 90s on:

Flexibility, easy integration, topology (sometimes)

Evolution of data models

The 90s on:

Flexibility, easy integration, topology (sometimes)

- Graphs
- XML
- RDF
- JSON
- CSV
- ...

Classical DB questions on new kind of data

Classical DB questions on new kind of data

- Data design
- Languages (markup, querying)
- Optimization
- Updates

Classical questions that gain renewed interest

Classical questions that gain renewed interest

- Uncertainty
- Distribution
- Ranking
- Query workloads

New questions on new kind of data

New questions on new kind of data

- Access
- Schema extraction
- Trust
- Variety

In this talk

We talk about Graph DBs, RDF and tabular data (CSV)

In this talk

We talk about Graph DBs, RDF and tabular data (CSV)

Bottomline:

Despite similarities with old DB problems, new applications require:

- understanding the nature of their problems
- refining old/developing new techniques

Graph Databases

The need for a standard graph QL in practice

Recognized by major graph DB engines

The need for a standard graph QL in practice

Recognized by major graph DB engines

- Neo4J $\rightarrow$ Cypher

The need for a standard graph QL in practice

Recognized by major graph DB engines

- Neo4J $\rightarrow$ Cypher
- Sparksee $\rightarrow$ Graph algebra

The need for a standard graph QL in practice

Recognized by major graph DB engines

- Neo4J $\rightarrow$ Cypher
- Sparksee $\rightarrow$ Graph algebra
- Oracle $\rightarrow$ PGQL

The need for a standard graph QL in practice

Recognized by major graph DB engines

- Neo4J $\rightarrow$ Cypher
- Sparksee $\rightarrow$ Graph algebra
- Oracle $\rightarrow$ PGQL
- Apache Tinkerpop $\rightarrow$ Gremlin

The need for a standard graph QL in practice

Recognized by major graph DB engines

- Neo4J $\rightarrow$ Cypher
- Sparksee $\rightarrow$ Graph algebra
- Oracle $\rightarrow$ PGQL
- Apache Tinkerpop $\rightarrow$ Gremlin

Not all of them are declarative

LDBC GraphQL task force

Formed by researchers/practitioners
(Neo4J, Sparksee, IBM, Oracle, SAP, HP)

LDBC GraphQL task force

Formed by researchers/practitioners
(Neo4J, Sparksee, IBM, Oracle, SAP, HP)

Objectives:

LDBC GraphQL task force

Formed by researchers/practitioners
(Neo4J, Sparksee, IBM, Oracle, SAP, HP)

Objectives:

1. Define a graph data model: Property graphs (July 2015)

LDBC GraphQL task force

Formed by researchers/practitioners
(Neo4J, Sparksee, IBM, Oracle, SAP, HP)

Objectives:

1. Define a graph data model: Property graphs (July 2015)
2. Identify features/limitations of existing QLs (December 2015)

LDBC GraphQL task force

Formed by researchers/practitioners
(Neo4J, Sparksee, IBM, Oracle, SAP, HP)

Objectives:

1. Define a graph data model: Property graphs (July 2015)
2. Identify features/limitations of existing QLs (December 2015)
3. Identify features needed in a graph QL (ongoing)

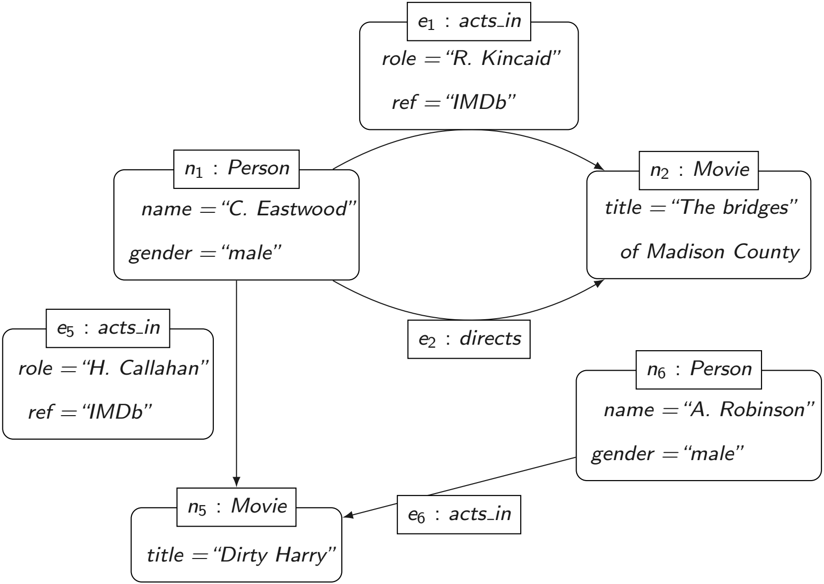
LDBC GraphQL task force

Formed by researchers/practitioners
(Neo4J, Sparksee, IBM, Oracle, SAP, HP)

Objectives:

1. Define a graph data model: Property graphs (July 2015)
2. Identify features/limitations of existing QLs (December 2015)
3. Identify features needed in a graph QL (ongoing)
4. Design a standard declarative graph QL (2017?)

A property graph by example



The property graph data model

Definition (Property graph)

A property graph G is a tuple $(V, E, \rho, \lambda, \sigma)$, where:

The property graph data model

Definition (Property graph)

A property graph G is a tuple $(V, E, \rho, \lambda, \sigma)$, where:

1. V is a finite set of *nodes*

The property graph data model

Definition (Property graph)

A property graph G is a tuple $(V, E, \rho, \lambda, \sigma)$, where:

1. V is a finite set of *nodes*
2. E is a finite set of *edges*

The property graph data model

Definition (Property graph)

A property graph G is a tuple $(V, E, \rho, \lambda, \sigma)$, where:

1. V is a finite set of *nodes*
2. E is a finite set of *edges*
3. $\rho : E \rightarrow (V \times V)$ is a total function
   - $\rho(e) = (v_1, v_2)$ means that e is of the form $v_1 \rightarrow v_2$

The property graph data model

Definition (Property graph)

A property graph G is a tuple $(V, E, \rho, \lambda, \sigma)$, where:

1. V is a finite set of *nodes*
2. E is a finite set of *edges*
3. $\rho : E \rightarrow (V \times V)$ is a total function
   - $\rho(e) = (v_1, v_2)$ means that e is of the form $v_1 \rightarrow v_2$
4. $\lambda : (V \cup E) \rightarrow Lab$ is a total function with Lab a set of labels
   - $\rho(v) = \ell$ means that ℓ is the label of node v in G

The property graph data model

Definition (Property graph)

A property graph G is a tuple $(V, E, \rho, \lambda, \sigma)$, where:

1. V is a finite set of *nodes*
2. E is a finite set of *edges*
3. $\rho : E \to (V \times V)$ is a total function
   - $\rho(e) = (v_1, v_2)$ means that e is of the form $v_1 \to v_2$
4. $\lambda : (V \cup E) \to Lab$ is a total function with Lab a set of labels
   - $\rho(v) = \ell$ means that ℓ is the label of node v in G
5. $\sigma : (V \cup E) \times Prop \to Val$ is a partial function with $Prop$ a finite set of properties and Val a set of values
   - $\sigma(v, p) = s$ means that s is the value of property p for v in G

In existing query languages

Basic unit for querying the structure:

- Graph patterns, a.k.a. conjunctive queries

In existing query languages

Basic unit for querying the structure:

- Graph patterns, a.k.a. conjunctive queries

Interpretation:

- "Match" a small graph pattern into a large property graph

In existing query languages

Basic unit for querying the structure:

- Graph patterns, a.k.a. conjunctive queries

Interpretation:

- “Match” a small graph pattern into a large property graph
- Semantics:
  homomorphism, isomorphism, edge-injective homomorphism

In existing query languages

Basic unit for querying the structure:

- Graph patterns, a.k.a. conjunctive queries

Interpretation:

- "Match" a small graph pattern into a large property graph
- Semantics:
  homomorphism, isomorphism, edge-injective homomorphism
- Property graph $\to$ Relational table

In existing query languages

Basic unit for querying the structure:

- Graph patterns, a.k.a. conjunctive queries

Interpretation:

- "Match" a small graph pattern into a large property graph
- Semantics:
  homomorphism, isomorphism, edge-injective homomorphism
- Property graph $\rightarrow$ Relational table

Example:

- $\exists y\,(x, \mathrm{directs}, y) \wedge (x, \mathrm{acts\_in}, y)$

In existing query languages

Basic unit for querying the structure:

- Graph patterns, a.k.a. conjunctive queries

Interpretation:

- "Match" a small graph pattern into a large property graph
- Semantics:
  homomorphism, isomorphism, edge-injective homomorphism
- Property graph $\to$ Relational table

Example:

- $\exists y\, (x, \mathrm{directs}, y) \wedge (x, \mathrm{acts\_in}, y)$
- MATCH (x) -[directs]-> (y) <-[acts_in]- (x)
  RETURN (x)

Dealing with paths is fundamental

Queries of the form $x \xrightarrow{a^*} y$ allowed in Cypher

- Retrieve pairs of nodes linked by a path of a's

Dealing with paths is fundamental

Queries of the form $x \xrightarrow{a^*} y$ allowed in Cypher

- Retrieve pairs of nodes linked by a path of a's
- Example: Friend-of-a-friend relationship

Dealing with paths is fundamental

Queries of the form $x \xrightarrow{a^*} y$ allowed in Cypher

- Retrieve pairs of nodes linked by a path of a's
- Example: Friend-of-a-friend relationship

Combined with patterns: patterns with reachability

Dealing with paths is fundamental

Queries of the form $x \xrightarrow{a^*} y$ allowed in Cypher

- Retrieve pairs of nodes linked by a path of a's
- Example: Friend-of-a-friend relationship

Combined with patterns: patterns with reachability

Reasonable data complexity so far: NLogspace

RPQs not implemented yet

... but considered to be desirable

RPQs not implemented yet

... but considered to be desirable

Regular Path Query (RPQ): $x \xrightarrow{L} y$, for L a regex

- Retrieve pairs of nodes linked by a path in L

RPQs not implemented yet

... but considered to be desirable

Regular Path Query (RPQ): $x \xrightarrow{L} y$, for L a regex

- Retrieve pairs of nodes linked by a path in L
- Example: $(\mathrm{acts\_in} \cdot \mathrm{acts\_in}^{-})^{+}$

RPQs not implemented yet

... but considered to be desirable

Regular Path Query (RPQ): $x \xrightarrow{L} y$, for L a regex

- Retrieve pairs of nodes linked by a path in L
- Example: $(\mathrm{acts\_in} \cdot \mathrm{acts\_in}^{-})^{+}$

Data complexity still reasonable: NLogspace

RPQs not implemented yet

... but considered to be desirable

Regular Path Query (RPQ): $x \xrightarrow{L} y$, for L a regex

- Retrieve pairs of nodes linked by a path in L
- Example: $(\mathrm{acts\_in} \cdot \mathrm{acts\_in}^{-})^{+}$

Data complexity still reasonable: NLogspace

- For a semantics based on arbitrary paths

Systems tend to favour a simple paths semantics

Systems tend to favour a simple paths semantics

Is there a simple path from x to y labeled in $(aa)^*$?

- NP-complete [Lapaugh & Papadimitriou, 1984]

Systems tend to favour a simple paths semantics

Is there a simple path from x to y labeled in $(aa)^*$?

- NP-complete [Lapaugh & Papadimitriou, 1984]

A simple path semantics already tried/discarded in SPARQL 1.1

- [Losseman & Martens, 2012; Arenas, Conca & Pérez, 2012]

They also return paths...

They also return paths...

Example

MATCH $p = (x)$ -[acts_in*0..5]-> (y)
RETURN p, length(p)

They also return paths...

Example

MATCH $p = (x)$ -[$\mathrm{acts\_in}$*0..5]-> (y)
RETURN p, length(p)

A single path, a simple path, all paths, all simple paths?

Just imagine this combined with RPQs...

...an exponential explosion in the size of the output

Just imagine this combined with RPQs...

...an exponential explosion in the size of the output

- What are the use cases for these functionalities?

Just imagine this combined with RPQs...

...an exponential explosion in the size of the output

- What are the use cases for these functionalities?
- Which paths users are looking for?

Just imagine this combined with RPQs...

...an exponential explosion in the size of the output

- What are the use cases for these functionalities?
- Which paths users are looking for?
- Can they be represented compactly?

Just imagine this combined with RPQs...

...an exponential explosion in the size of the output

- What are the use cases for these functionalities?
- Which paths users are looking for?
- Can they be represented compactly?
- Can they be enumerated with reasonable delay?

Just imagine this combined with RPQs...

...an exponential explosion in the size of the output

- What are the use cases for these functionalities?
- Which paths users are looking for?
- Can they be represented compactly?
- Can they be enumerated with reasonable delay?

From a theoretical point of view:

- Return a single shortest path (complexity in NLogspace)

Just imagine this combined with RPQs...

...an exponential explosion in the size of the output

- What are the use cases for these functionalities?
- Which paths users are looking for?
- Can they be represented compactly?
- Can they be enumerated with reasonable delay?

From a theoretical point of view:

- Return a single shortest path (complexity in NLogspace)
- Return others, one by one, if needed

A representation problem

How paths should be represented?

A representation problem

How paths should be represented?

- As a tuple in a table?

A representation problem

How paths should be represented?

- As a tuple in a table?
- As a graph?

A representation problem

How paths should be represented?

- As a tuple in a table?
- As a graph?
- As a new “path” data type?

To complicate things more

Existing QLs have an ungrouping operator:

- List set (bag) of nodes that belong to a path

To complicate things more

Existing QLs have an ungrouping operator:

- List set (bag) of nodes that belong to a path
- Path $\rightarrow$ Relational table

To complicate things more

Existing QLs have an ungrouping operator:

- List set (bag) of nodes that belong to a path
- Path $\rightarrow$ Relational table

The power of ungrouping:

1. Find a path that visits each node exactly once
2. Find a path that does not repeat values for a given property
3. Find paths that intersect in a node

To complicate things more

Existing QLs have an ungrouping operator:

- List set (bag) of nodes that belong to a path
- Path $\rightarrow$ Relational table

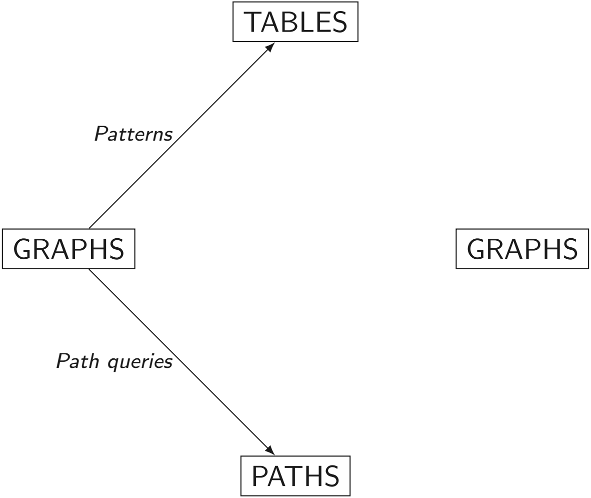
The power of ungrouping:

1. Find a path that visits each node exactly once
2. Find a path that does not repeat values for a given property
3. Find paths that intersect in a node

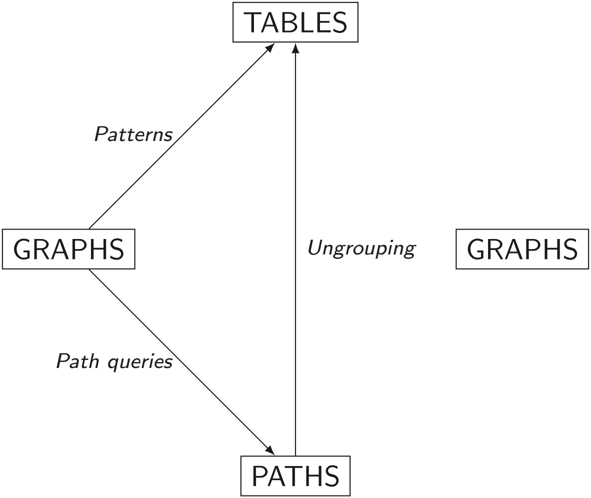
Theory: Graph QLs for data

- Complexity is astronomical [Barceló, Fontaine & Lin, 2013]
- Tractability conditions are delicate [Libkin & Vrgoč, 2012]

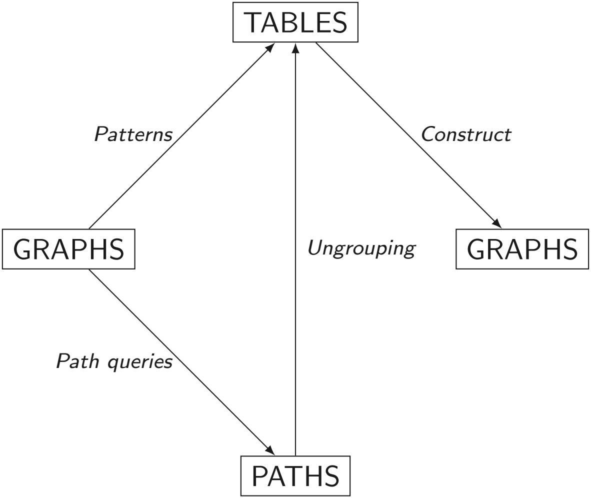
Type transformations



Type transformations



Type transformations



What have we learned?

- Real need from graph DBs to have a standard declarative QL

What have we learned?

- Real need from graph DBs to have a standard declarative QL
- Design is non-trivial due to efficiency/expressiveness trade-off

What have we learned?

- Real need from graph DBs to have a standard declarative QL
- Design is non-trivial due to efficiency/expressiveness trade-off
- Theory is not so much far away from practice

What have we learned?

- Real need from graph DBs to have a standard declarative QL
- Design is non-trivial due to efficiency/expressiveness trade-off
- Theory is not so much far away from practice
- It requires a political effort to "close the gap"

Semantic Web

Semantic Web

"The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation."

[Tim Berners-Lee et al. 2001.]

Semantic Web

"The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation."

[Tim Berners-Lee et al. 2001.]

Specific goals:

Semantic Web

"The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation."

[Tim Berners-Lee et al. 2001.]

Specific goals:

- Build a description language with standard semantics

Semantic Web

"The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation."

[Tim Berners-Lee et al. 2001.]

Specific goals:

- Build a description language with standard semantics
  - Make semantics machine-processable and understandable

Semantic Web

"The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation."

[Tim Berners-Lee et al. 2001.]

Specific goals:

- Build a description language with standard semantics
  - Make semantics machine-processable and understandable
- Incorporate logical infrastructure to reason about resources

Semantic Web

"The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation."

[Tim Berners-Lee et al. 2001.]

Specific goals:

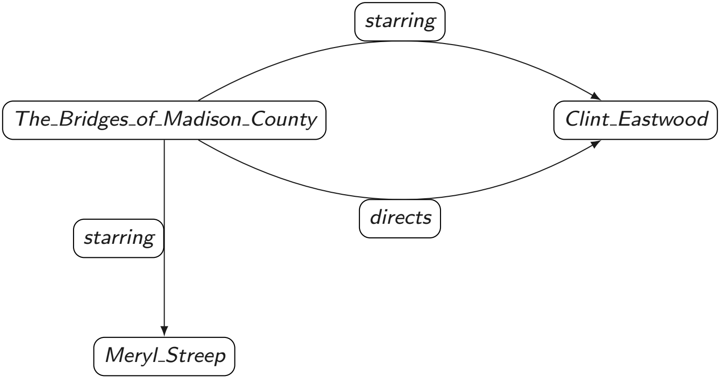
- Build a description language with standard semantics
  - Make semantics machine-processable and understandable
- Incorporate logical infrastructure to reason about resources

W3C proposals: Resource Description Framework (RDF) and SPARQL

RDF in a nutshell

- RDF is the W3C proposal framework for representing information in the Web
- Abstract syntax based on directed labeled graph
- Extensible URI-based vocabulary

An RDF graph



starring
The_Bridges_of_Madison_County
Clint_Eastwood
directs
starring
Meryl_Streep

An RDF graph in real life: DBpedia

```
http://dbpedia.org/resource/The_Bridges_of_Madison_County_(film)
http://dbpedia.org/property/director
http://dbpedia.org/resource/Clint_Eastwood .

http://dbpedia.org/resource/The_Bridges_of_Madison_County_(film)
http://dbpedia.org/property/starring
http://dbpedia.org/resource/Clint_Eastwood .

http://dbpedia.org/resource/The_Bridges_of_Madison_County_(film)
http://dbpedia.org/property/starring
http://dbpedia.org/resource/Meryl_Streep .
```

Prefixes simplify the notation

Prefixes can be defined in an RDF graph to simplify notation

- They are defined also using triples

Prefixes simplify the notation

Prefixes can be defined in an RDF graph to simplify notation

- They are defined also using triples

We can include in an RDF graph the following triples:

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
```

Prefixes simplify the notation

Prefixes can be defined in an RDF graph to simplify notation

- They are defined also using triples

We can include in an RDF graph the following triples:

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
```

Then `<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>` can be replaced by `rdf:type`

How are URIs created and assigned?

How are URIs created and assigned?

There is no centralized mechanism

How are URIs created and assigned?

There is no centralized mechanism

A key component to deal with this issue: `owl:sameAs`

```
http://dbpedia.org/resource/Meryl_Streep   owl:sameAs
http://cs.dbpedia.org/resource/Meryl_Streepová .

http://dbpedia.org/resource/Meryl_Streep   owl:sameAs
http://yago-knowledge.org/resource/Meryl_Streep .

http://dbpedia.org/resource/Meryl_Streep   owl:sameAs
http://data.nytimes.com/32250484050106278413 .
```

How are URIs created and assigned?

There is no centralized mechanism

A key component to deal with this issue: `owl:sameAs`

```
http://dbpedia.org/resource/Meryl_Streep   owl:sameAs
http://cs.dbpedia.org/resource/Meryl_Streepová .

http://dbpedia.org/resource/Meryl_Streep   owl:sameAs
http://yago-knowledge.org/resource/Meryl_Streep .

http://dbpedia.org/resource/Meryl_Streep   owl:sameAs
http://data.nytimes.com/32250484050106278413 .
```

Reasoning capabilities are needed to deal with `owl:sameAs`

Querying RDF: SPARQL

- SPARQL is the W3C recommendation query language for RDF (January 2008)
- Originally it was a graph-matching query language
- SPARQL 1.1 is the new version of this language, its was released in March 2013

An example of a SPARQL query

Retrieve all the movies in DBpedia

An example of a SPARQL query

Retrieve all the movies in DBpedia

```
?movie  rdf:type  <http://schema.org/Movie> .
```

An example of a SPARQL query

Retrieve all the movies in DBpedia

```
WHERE
{
   ?movie  rdf:type  <http://schema.org/Movie> .
}
```

An example of a SPARQL query

Retrieve all the movies in DBpedia

```
SELECT ?movie
WHERE
{
   ?movie  rdf:type  <http://schema.org/Movie> .
}
```

Important problems when querying RDF

Important problems when querying RDF

- Returning as much information as possible

Important problems when querying RDF

- Returning as much information as possible
- Reasoning with ontologies

Important problems when querying RDF

- Returning as much information as possible
- Reasoning with ontologies
- Dealing with incomplete information

Important problems when querying RDF

- Returning as much information as possible
- Reasoning with ontologies
- Dealing with incomplete information
- Exploiting the graph structure of RDF

Important problems when querying RDF

- Returning as much information as possible
- Reasoning with ontologies
- Dealing with incomplete information
- Exploiting the graph structure of RDF
- Working with highly distributed data

Returning as much information as possible

RDF follows an open world assumption

Users may be unaware of the structure of the data

Returning as much information as possible

RDF follows an open world assumption

Users may be unaware of the structure of the data

Thus, the possibility of obtaining additional information if possible is important in this scenario

- In fact, this feature was present from the very beginning in SPARQL

An optional operator

Retrieve each movie in DBpedia and its gross if this information is available

```
SELECT ?movie ?gross
WHERE
{
   ?movie  rdf:type  <http://schema.org/Movie> .
   OPTIONAL
   {
      ?movie  <http://dbpedia.org/property/gross>  ?gross .
   }
}
```

Part of the answer to the query

?movie	?gross
http://dbpedia.org/resource/Frozen_(2013_film)	"1.274E9"
http://dbpedia.org/resource/Amazon_Souls	

What is new?

The `OPTIONAL` operator essentially corresponds to a left-outer join in relational algebra

But ...

What is new?

The `OPTIONAL` operator essentially corresponds to a left-outer join in relational algebra

But ...

- The fragments of SPARQL that are natural to study are different than for the case of relational algebra
  - The complexity of evaluating these fragments was not known [Pérez, A. & Gutierrez 2009; Schmidt, Meier & Lausen 2010]
- New notions of safeness are needed to avoid a counterintuitive behavior [Pérez, A. & Gutierrez 2009]
- New optimization techniques are needed [Pérez, A. & Gutierrez 2009; Letelier, Pérez, Pichler & Skritek 2013; Pichler & Skritek 2014]

Openness influences other operators

```
SELECT ?voice_actor ?film_actor
WHERE
{
   {
      ?voice_actor  rdf:type
      <http://dbpedia.org/class/yago/AmericanVoiceActors> .
   }
   UNION
   {
      ?film_actor  rdf:type
      <http://dbpedia.org/class/yago/AmericanFilmActors> .
   }
}
```

The answer to the query

?voice_actor	?film_actor
http://dbpedia.org/resource/Alec_Baldwin	
	http://dbpedia.org/resource/Meryl_Streep

Reasoning with ontologies

Reasoning capabilities are needed

Reasoning with ontologies

Reasoning capabilities are needed

- We already mentioned owl:sameAs

Reasoning with ontologies

Reasoning capabilities are needed

- We already mentioned owl:sameAs
- An RDF graph can use RDF Schema (RDFS) to establish hierarchies of classes and properties

Reasoning with ontologies

Reasoning capabilities are needed

- We already mentioned owl:sameAs
- An RDF graph can use RDF Schema (RDFS) to establish hierarchies of classes and properties
- The Web Ontology Language (OWL) can be used to define more complex relations between classes and properties

The use of RDFS vocabulary

The following triples are included in DBpedia:

The use of RDFS vocabulary

The following triples are included in DBpedia:

```
http://dbpedia.org/class/yago/Professor110480730
rdfs:subClassOf
http://dbpedia.org/class/yago/Academician109759069 .

http://dbpedia.org/class/yago/Academician109759069
rdfs:subClassOf
http://dbpedia.org/class/yago/Educator110045713 .

http://dbpedia.org/ontology/championInDoubleFemale
rdfs:subPropertyOf
http://dbpedia.org/ontology/championInDouble .

http://dbpedia.org/ontology/championInDouble
rdfs:subPropertyOf
http://dbpedia.org/ontology/champion .
```

The use of RDFS vocabulary

Some numbers in DBpedia:

- triples with `rdfs:subClassOf` as predicate are at least 450K
- triples with `rdfs:subPropertyOf` as predicate are at least 1K

The use of RDFS vocabulary

Some numbers in DBpedia:

- triples with `rdfs:subClassOf` as predicate are at least 450K
- triples with `rdfs:subPropertyOf` as predicate are at least 1K

We need reasoning capabilities to deal with:

- `rdfs:subClassOf`, `rdfs:subPropertyOf`

The use of RDFS vocabulary

Some numbers in DBpedia:

- triples with `rdfs:subClassOf` as predicate are at least 450K
- triples with `rdfs:subPropertyOf` as predicate are at least 1K

We need reasoning capabilities to deal with:

- `rdfs:subClassOf`, `rdfs:subPropertyOf`
- and other elements of RDFS such as `rdfs:domain` and `rdfs:range`

Answering a query with RDFS vocabulary

Retrieve all the educators in DBpedia

```
SELECT ?educator
WHERE
{
   ?educator  rdf:type
   <http://dbpedia.org/class/yago/Educator110045713> .
}
```

Answering a query with RDFS vocabulary

The answer to the previous query should be the same as for the following query:

```
SELECT ?educator
WHERE
{
   {  ?educator  rdf:type
        . }
   UNION
   {  ?educator  rdf:type
      <http://dbpedia.org/class/yago/Educator110045713> . }
}
```

Open issues about reasoning with ontologies

Two important problems:

- Development of efficient query answering algorithms over large RDF graphs with RDFS vocabulary
- Identification of fragments of OWL that have good expressive power and can be efficiently evaluated

Exploiting the graph structure of RDF

The structure of an RDF graph stores information

It is important to have operators that can deal with this structure

- In particular, navigating an RDF graph is an important functionality

Exploiting the graph structure of RDF

The structure of an RDF graph stores information

It is important to have operators that can deal with this structure

- In particular, navigating an RDF graph is an important functionality

Properties paths in SPARQL allow to express reachability queries

Navigating RDF graphs

Get starring actors in the same movie:

```
SELECT ?actor1 ?actor2
WHERE
{
   ?movie  <http://dbpedia.org/property/starring>  ?actor1 .
   ?movie  <http://dbpedia.org/property/starring>  ?actor2 .
}
```

Navigating RDF graphs

Previous query can be rewritten by using navigation patterns:

Navigating RDF graphs

Previous query can be rewritten by using navigation patterns:

```
SELECT ?actor1 ?actor2
WHERE
{
   ?actor1 ^<http://dbpedia.org/property/starring> ?movie .
   ?movie <http://dbpedia.org/property/starring> ?actor2 .
}
```

Navigating RDF graphs

Previous query can be rewritten by using navigation patterns:

```
SELECT ?actor1 ?actor2
WHERE
{
   ?actor1
   ^<http://dbpedia.org/property/starring>/

   ?actor2 .
}
```

Navigating RDF graphs

Previous query can be rewritten by using navigation patterns:

```
SELECT ?actor1 ?actor2
WHERE
{
   ?actor1
   ^<http://dbpedia.org/property/starring>/

   ?actor2 .
}
```

The expression in red is called a property path

Navigating RDF graphs

Get starring actors that are connected:

```
SELECT ?actor1 ?actor2
WHERE
{
   ?actor1
   (^<http://dbpedia.org/property/starring>/
     <http://dbpedia.org/property/starring>)+
   ?actor2 .
}
```

Navigating RDF graphs

Get starring actors that are connected:

```
SELECT ?actor1 ?actor2
WHERE
{
   ?actor1
   (^<http://dbpedia.org/property/starring>/
     <http://dbpedia.org/property/starring>)+
   ?actor2 .
}
```

Can this query be answered?

- Can it be answered starting from a specific node?

Open issues in exploiting the graph structure of RDF

Some important problems:

- Development of efficient evaluation algorithms for reachability queries over large RDF graphs
- Standardization of a query language where paths are first-class citizens

Web data is highly distributed

Data can be stored in different repositories

- Different pieces of data have to be collected to answer a query

Web data is highly distributed

Data can be stored in different repositories

- Different pieces of data have to be collected to answer a query

An important notion to deal with this issue: SPARQL endpoint

- A Web service that accepts a SPARQL query as input, and returns (part of) the result to the query

Web data is highly distributed

Data can be stored in different repositories

- Different pieces of data have to be collected to answer a query

An important notion to deal with this issue: SPARQL endpoint

- A Web service that accepts a SPARQL query as input, and returns (part of) the result to the query

SPARQL has an operator `SERVICE` to query an endpoint

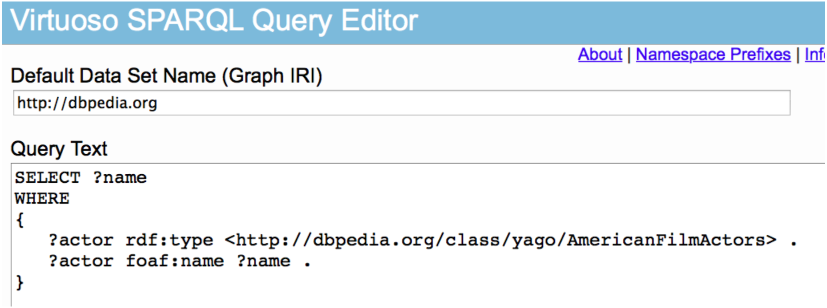
The SPARQL endpoint of DBpedia

Querying DBpedia

We want to retrieve the list of American actors in DBpedia

Querying DBpedia

We want to retrieve the list of American actors in DBpedia



The answer to the query

name
"Courtenay Taylor"@en
"Taylor, Courtenay"@en
"Nakia Burrise"@en
"Burrise, Nakia"@en
"Alan Hale, Sr."@en
"Hale, Alan, Sr."@en
"Alec Baldwin"@en
"Baldwin, Alec"@en

...

The SPARQL endpoint of DBLP

We want to retrieve the list of authors in DBLP

The SPARQL endpoint of DBLP

We want to retrieve the list of authors in DBLP

SPARQL:

```
PREFIX d2r: <http://sites.wiwiss.fu-berlin.de/suhl/bizer/d2r-server/config.rdf#>
PREFIX swrc: <http://swrc.ontoware.org/ontology#>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX map: <file:///home/diederich/d2r-server-0.3.2/dblp-mapping.n3#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
```

```
SELECT ?name
WHERE
{
  ?paper  dc:creator  ?author .
  ?author  foaf:name  ?name .
}
```

Results: Browse | Go! | Reset

The answer to the query

SPARQL results:

name
"Sanjeev Saxena"
"Hans-Ulrich Simon"
"Nathan Goodman"
"Oded Shmueli"
"Norbert Blum"
"Arnold Schönhage"
"Juha Honkala"
"Chua-Huang Huang"
"Christian Lengauer"

...

We would like to combine the previous results . . .

```
SELECT ?name
WHERE
{
  ?actor rdf:type <http://dbpedia.org/class/yago/AmericanActors> .
  ?actor foaf:name ?name .
  SERVICE <http://dblp.l3s.de/d2r/sparql>
  {
     SELECT ?name
     WHERE
     {
        ?paper dc:creator ?author .
        ?author foaf:name ?name .
     }
  }
}
```

Open issues when dealing with distribution

Some important problems:

- The notion of SPARQL endpoint needs to be formalized
  - What queries are accepted?
  - How is the time distributed between them?
  - Should a pricing model be used?
  - What is the protocol to return the answer to a query?

- A more general notion of endpoint should be formalized and studied

Open issues when dealing with distribution (cont'd)

- Usability needs to be hugely improved
  - schema/structure extraction and visualization play a fundamental role here
- Approaches for discovering relevant data should be studied
- Operators to distribute the execution of queries should be studied in more depth

A first issue about the SERVICE operator

SERVICE operators can be nested

```
SELECT ...
WHERE
{
   ...
   SERVICE <uri1>
   {
      ...
      SERVICE <uri2>
   }
}
```

A second issue about the SERVICE operator

SERVICE can be used not only with a URI but also with a variable

A second issue about the SERVICE operator

SERVICE can be used not only with a URI but also with a variable

```
SELECT ...
WHERE
{
   ...
   ?address rdf:type <http://example.org/SparqlEndpoint> .
   SERVICE ?address
   {
      ...
   }
}
```

A second issue about the SERVICE operator

SERVICE can be used not only with a URI but also with a variable

```
SELECT ...
WHERE
{
   ...
   SERVICE ?address
   {
      ...
      ?uri rdf:type <http://local_example.org/SparqlEndpoint> .
   }
   SERVICE ?uri
   {
      ...
   }
}
```

A second issue about the SERVICE operator

SERVICE can be used not only with a URI but also with a variable

Several specific issues have to be addressed [Buil-Aranda, A. & Corcho 2011]:

- If ?var does not have a value, how SERVICE ?var should be evaluated?
  - A notion of safeness is needed
- The situation gets more involved if we also have nested SERVICE operators
- The syntax of SPARQL 1.1 allows SERVICE ?var, but its semantics is not defined

What have we learned?

- There are many interesting and challenging questions to be answered

What have we learned?

- There are many interesting and challenging questions to be answered
- Most of these questions can be considered as classical DB questions

What have we learned?

- There are many interesting and challenging questions to be answered
- Most of these questions can be considered as classical DB questions
- But answering them require of a combination of classical and new techniques

What have we learned?

- There are many interesting and challenging questions to be answered
- Most of these questions can be considered as classical DB questions
- But answering them require of a combination of classical and new techniques
- The Semantic Web community has been receptive to our ideas

Tabular Data (CSV)

Comma-separated values (CSV) documents

Comma-separated values (CSV) documents

The IBM Fortran compiler supported CSV documents in 1972

Comma-separated values (CSV) documents

The IBM Fortran compiler supported CSV documents in 1972

The W3C recommendation for CSV documents (tabular data) was released on December 2015

Comma-separated values (CSV) documents

The IBM Fortran compiler supported CSV documents in 1972

The W3C recommendation for CSV documents (tabular data) was released on December 2015

Why such a recommendation is needed?

- This is just tabular data, what is new?

The main goals of the W3C recommendation

- Define a schema language for CSV
  - For example, it can be used to specify the name and the type of the elements of each column in a CSV document
- Develop a metadata vocabulary for CSV
  - It describes how the data should be interpreted
- Define mechanisms for transforming CSV into RDF, JSON and XML

CSV documents can be messy

A use case from `http://www.w3.org/TR/csvw-ucr`:

> The US National Institute of Standards and Technology has run various conferences on extracting information from text. The extracted information is submitted in a tab-separated format.

CSV documents can be messy

A use case from `http://www.w3.org/TR/csvw-ucr`:

> The US National Institute of Standards and Technology has run various conferences on extracting information from text. The extracted information is submitted in a tab-separated format.

An example document:

```
    ...
:e4 type         PER
:e4 mention      "Bart"  D00124 283-286
:e4 mention      "JoJo"  D00124 145-149 0.9
:e4 per:siblings :e7     D00124 283-286 173-179 274-281
:e4 per:age      "10"    D00124 180-181 173-179 182-191 0.9
    ...
```

A schema language for CSV

We describe an approach proposed by [Martens, Neven & Vansummeren 2015]

A schema language for CSV

We describe an approach proposed by [Martens, Neven & Vansummeren 2015]

The model for a CSV document:

	1	2	3	4	5	6
1	:e4	type	PER			
2	:e4	mention	"Bart"	D00124	283-286	
3	:e4	mention	"JoJo"	D00124	145-149	0.9
4	:e4	per:siblings	:e7	D00124	283-286	173-179

A schema language for CSV

We describe an approach proposed by [Martens, Neven & Vansummeren 2015]

The model for a CSV document:

	1	2	3	4	5	6
1	:e4	type	PER			
2	:e4	mention	"Bart"	D00124	283-286	
3	:e4	mention	"JoJo"	D00124	145-149	0.9
4	:e4	per:siblings	:e7	D00124	283-286	173-179

Every cell has a coordinate (i, j)

- The cell with content `"Bart"` has coordinate $(2, 3)$

Checking conditions on regions

	1	2	3	4	5	6
1	:e4	type	PER			
2	:e4	mention	"Bart"	D00124	283-286	
3	:e4	mention	"JoJo"	D00124	145-149	0.9
4	:e4	per:siblings	:e7	D00124	283-286	173-179

Checking conditions on regions

	1	2	3	4	5	6
1	:e4	type	PER			
2	:e4	mention	"Bart"	D00124	283-286	
3	:e4	mention	"JoJo"	D00124	145-149	0.9
4	:e4	per:siblings	:e7	D00124	283-286	173-179

[root]

Checking conditions on regions

	1	2	3	4	5	6
1	:e4	type	PER			
2	:e4	mention	"Bart"	D00124	283-286	
3	:e4	mention	"JoJo"	D00124	145-149	0.9
4	:e4	per:siblings	:e7	D00124	283-286	173-179

[root] · right

Checking conditions on regions

	1	2	3	4	5	6
1	:e4	type	PER			
2	:e4	mention	"Bart"	D00124	283-286	
3	:e4	mention	"JoJo"	D00124	145-149	0.9
4	:e4	per:siblings	:e7	D00124	283-286	173-179

[root] · right · right

Checking conditions on regions

	1	2	3	4	5	6
1	:e4	type	PER			
2	:e4	mention	"Bart"	D00124	283-286	
3	:e4	mention	"JoJo"	D00124	145-149	0.9
4	:e4	per:siblings	:e7	D00124	283-286	173-179

[root] · right · right · right

Checking conditions on regions

	1	2	3	4	5	6
1	:e4	type	PER			
2	:e4	mention	"Bart"	D00124	283-286	
3	:e4	mention	"JoJo"	D00124	145-149	0.9
4	:e4	per:siblings	:e7	D00124	283-286	173-179

$[\mathtt{root}] \cdot \mathtt{right} \cdot \mathtt{right} \cdot \mathtt{right} \cdot \mathtt{right}^*$

Checking conditions on regions

	1	2	3	4	5	6
1	:e4	type	PER			
2	:e4	mention	"Bart"	D00124	283-286	
3	:e4	mention	"JoJo"	D00124	145-149	0.9
4	:e4	per:siblings	:e7	D00124	283-286	173-179

$[\mathtt{root}] \cdot \mathtt{right} \cdot \mathtt{right} \cdot \mathtt{right} \cdot \mathtt{right}^* \cdot \mathtt{down}^*$

Checking conditions on regions

	1	2	3	4	5	6
1	:e4	type	PER			
2	:e4	mention	"Bart"	D00124	283-286	
3	:e4	mention	"JoJo"	D00124	145-149	0.9
4	:e4	per:siblings	:e7	D00124	283-286	173-179

$[\mathtt{root}] \cdot \mathtt{right} \cdot \mathtt{right} \cdot \mathtt{right} \cdot \mathtt{right}^* \cdot \mathtt{down}^* \rightarrow R$

An annotation language

We describe our proposal in [A., Maturana, Riveros & Vrgoč 2016]

An annotation language

We describe our proposal in [A., Maturana, Riveros & Vrgoč 2016]

A CSV document is represented as a string

An annotation language

We describe our proposal in [A., Maturana, Riveros & Vrgoč 2016]

A CSV document is represented as a string

```
:e4 type         PER
:e4 mention      "Bart"  D00124 283-286
:e4 mention      "JoJo"  D00124 145-149 0.9
:e4 per:siblings :e7     D00124 283-286 173-179 274-281
:e4 per:age      "10"    D00124 180-181 173-179 182-191 0.9
```

is represented as:

```
: e 4 \t t y p e \t P E R \n : e 4 \t m e n t i o n \t ...
```

Annotation of regions

Let w be the string representing a CSV document

The main ingredients of our approach:

Annotation of regions

Let w be the string representing a CSV document

The main ingredients of our approach:

- A region of w is a span (i, j) [Fagin, Kimelfeld, Reiss & Vansummeren 2013]

Annotation of regions

Let w be the string representing a CSV document

The main ingredients of our approach:

- A region of w is a span (i, j) [Fagin, Kimelfeld, Reiss & Vansummeren 2013]
- Regular expression with variables are used to extract spans from w
  - Fragments that can be evaluated very efficiently have been identified

Annotation of regions

Let w be the string representing a CSV document

The main ingredients of our approach:

- A region of w is a span (i, j) [Fagin, Kimelfeld, Reiss & Vansummeren 2013]
- Regular expression with variables are used to extract spans from w
  - Fragments that can be evaluated very efficiently have been identified
- Datalog programs are used to combine the results of the extraction process and annotate spans [Shen, Doan, Naughton & Ramakrishnan 2007]

Annotation of regions

Recall that w is the string representation of:

```
:e4 type          PER
:e4 mention       "Bart"  D00124 283-286
:e4 mention       "JoJo"  D00124 145-149 0.9
:e4 per:siblings :e7      D00124 283-286 173-179 274-281
:e4 per:age       "10"    D00124 180-181 173-179 182-191 0.9
```

Assume that the alphabet of w is Σ and $\Delta = (\Sigma - \{\backslash\mathtt{t}, \backslash\mathtt{n}\})$

Annotation of regions

Recall that w is the string representation of:

```
:e4 type          PER
:e4 mention       "Bart"  D00124 283-286
:e4 mention       "JoJo"  D00124 145-149 0.9
:e4 per:siblings  :e7     D00124 283-286 173-179 274-281
:e4 per:age       "10"    D00124 180-181 173-179 182-191 0.9
```

Assume that the alphabet of w is Σ and $\Delta = (\Sigma - \{\backslash\mathtt{t}, \backslash\mathtt{n}\})$

The following Datalog program extracts and annotates all spans in the first column of w:

Annotation of regions

Recall that w is the string representation of:

```
:e4 type          PER
:e4 mention       "Bart"  D00124 283-286
:e4 mention       "JoJo"  D00124 145-149 0.9
:e4 per:siblings :e7      D00124 283-286 173-179 274-281
:e4 per:age       "10"    D00124 180-181 173-179 182-191 0.9
```

Assume that the alphabet of w is Σ and $\Delta = (\Sigma - \{\backslash\mathtt{t}, \backslash\mathtt{n}\})$

The following Datalog program extracts and annotates all spans in the first column of w:

$$w.x\backslash\mathtt{t}\Sigma^* \;\wedge\; x.\Delta^* \quad\rightarrow\quad \mathrm{FC}(x)$$

Annotation of regions

Recall that w is the string representation of:

```
:e4 type         PER
:e4 mention      "Bart"  D00124 283-286
:e4 mention      "JoJo"  D00124 145-149 0.9
:e4 per:siblings :e7     D00124 283-286 173-179 274-281
:e4 per:age      "10"    D00124 180-181 173-179 182-191 0.9
```

Assume that the alphabet of w is Σ and $\Delta = (\Sigma - \{\backslash\mathtt{t}, \backslash\mathtt{n}\})$

The following Datalog program extracts and annotates all spans in the first column of w:

$$
\begin{array}{rcl}
w.x\backslash\mathtt{t}\Sigma^* \;\wedge\; x.\Delta^* & \rightarrow & \mathrm{FC}(x) \\
w.\Sigma^*\backslash\mathtt{n}x\backslash\mathtt{t}\Sigma^* \;\wedge\; x.\Delta^* & \rightarrow & \mathrm{FC}(x)
\end{array}
$$

Open issues in the area

Many questions need to be answered:

- What is a good schema language for CSV?
- What is a good annotation language for CSV?
- How metadata should be specified for a CSV document?
- What is a good mapping language to specify the transformation of a CSV document to an RDF graph?
  - The same question needs to be answered for JSON and XML

Final remarks

Do not underestimate the theoretical interest behind new apps:

- Classical DB questions gain new flavors over new data models
- It might involve developing techniques of independent interest
- Some areas are in need of theoretical help
- It requires an effort to understand problems and transfer tools

Final remarks

Do not underestimate the theoretical interest behind new apps:

- Classical DB questions gain new flavors over new data models
- It might involve developing techniques of independent interest
- Some areas are in need of theoretical help
- It requires an effort to understand problems and transfer tools

Many problems are still in need of formalization

- structure extraction, access, trust, ...

Thank you!