

LDOW, WWW'09

April 20, 2009,

Madrid, Spain

# MashQL

## A Data Mashup Language for the Data Web

Dr. Mustafa Jarrar

[mjarrar@cs.ucy.ac.cy](mailto:mjarrar@cs.ucy.ac.cy)

HPCLab, University of Cyprus

**Published As:**

Mustafa Jarrar and Marios D. Dikaiakos: A Data Mashup Language for the Data Web. Proceedings of LDOW, part of WWW'09. ACM. 2009.

University of Cyprus © 2009

## Motivation (Querying the Data Web)

Can we query and mash up the Data Web as simple as filtering and piping Web Feeds?

- We *fundamentally* investigate this problem from a **Query Formulation** viewpoint.
- A “data mashup” is a query.

# Outline

- Challenges
- Related Work
- MashQL (A Query Formulation Language)
- Evaluation
- Conclusions and Discussion

# Challenges

I don't know the schema!  
(Black-box query)



www.site1.com/rdf

```
:B1 :Title "Linked Data"
:B1 :Author "Lara T."
:B1 :PubYear 2008
:B1 :Publisher "Springer"
:B2 :Title "Data on the Web"
:B2 :Author "Abiteboul S."
```

## Problem Definition

We need a way to allow end-users formulate queries over structured data assuming that:

- *The user does not know the schema.*

(1)

# Challenges



www.site1.com/rdf

:B1 :Title "Linked Data"  
:B1 :Author "Lara T."  
:B1 :PubYear 2008  
:B1 :Publisher "Springer"  
:B2 :Title "Data on the Web"  
:B2 :Author "Abiteboul S."

Does not adhere to  
schema or ontology!

## Problem Definition

How to allow *end-users* formulate queries over structured data assuming that:

- *The user does not know the schema.* (1)
- *There is no offline or inline schema\ontology.* (2)

# Challenges

Allow me to easily  
Compose what I need !



www.site1.com/rdf

```
:B1 :Title "Linked Data"
:B1 :Author "Lara T."
:B1 :PubYear 2008
:B1 :Publisher "Springer"
:B2 :Title "Data on the Web"
:B2 :Author "Abiteboul S."
```

www.site2.com/rdf

```
:A1 rdf:type bibo:Article
:A1 :Title "Data Web"
:A1 :Author "Tom Lara"
:A1 :Author "Bob Hacker"
:A1 :Year 2007
:A2 rdf:type bibo:Article
:A2 :Title "Semantic Web"
:A2 :Author "Tom Lara"
:A2 :Year 2005
```

## Problem Definition

How to allow *end-users* formulate queries over structured data assuming that:

- *The user does not know the schema.* (1)
- *There is no offline or inline schema\ontology.* (2)
- *The query may involve multiple sources.* (3)

# Challenges

General Solution!



Problem Definition

www.site1.com/rdf

```
:B1 :Title "Linked Data"
:B1 :Author "Lara T."
:B1 :PubYear 2008
:B1 :Publisher "Springer"
:B2 :Title "Data on the Web"
:B2 :Author "Abiteboul S."
```

www.site2.com/rdf

```
:A1 rdf:type bibo:Article
:A1 :Title "Data Web"
:A1 :Author "Tom Lara"
:A1 :Author "Bob Hacker"
:A1 :Year 2007
:A2 rdf:type bibo:Article
:A2 :Title "Semantic Web"
:A2 :Author "Tom Lara"
:A2 :Year 2005
```

How to allow *end-users* formulate queries over structured data assuming that:

- *The user does not know the schema.* (1)
- *There is no offline or inline schema\ontology.* (2)
- *The query may involve multiple sources.* (3)
- *The query language is sufficiently expressive* (4)  
*(not a single-purpose interface)*

# Related Work

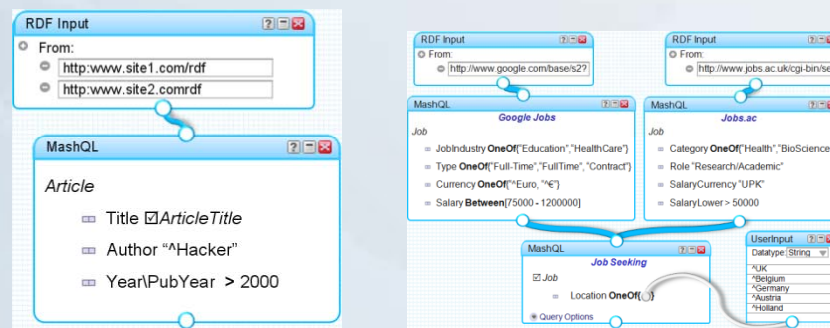
**Query formulation** is the art of accessing and consuming structured data *easily*. (interdisciplinary subject)

Assumption	Query by Form	Query by Example	Conceptual Queries (ConQuer)	NL Queries	Interactive Queries (Lorel)	Visual Scripting (DeriPipes)
<i>Don't know the schema</i>	✓	✗	✗	✓	✓	✗
<i>Schema-free data</i>	✗	✗	✗	✗	-	✓
<i>Multiple sources</i>	✗	✓	✗	✗	✗	✓
<i>Expressive</i>	✗	✓	✓	✓	-	✓
<i>Intuitive</i>	✓	✗	✗	✓	✓	✗



# MashQL

A graphical query formulation Language.



*all of these assumptions:*

- ✓ *The user does not know the schema.*
- ✓ *There is no offline or inline schema\ontology.*
- ✓ *The query may involve multiple sources.*
- ✓ *The query language is expressive.*  
(not a single-purpose interface)

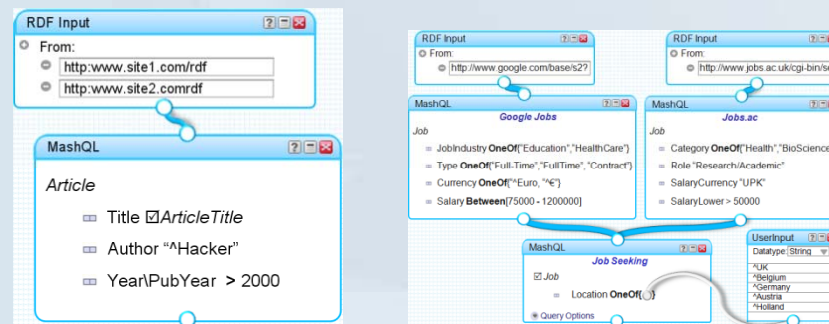
Challenging  
combination

# MashQL

A general structured-data retrieval solution.  
(not merely an interface)

*Without losing this generality, we*

- *Focus on RDF, as the most primitive query language, other data models can be mapped into it.*
- *Follow Yahoo Pipes' visualization, in order to illustrate that Data Web can be queried and mashed up as Web Feeds.*



# Example 1

“Lara’s articles after 2007?”

RDF Input

From:

- http://www.site1.com/rdf
- http://www.site2.com/rdf

Query

- Everything
  - Author ^Lara
  - Year\PubYear > 2007
  - Title ☒ ArticleTitle

www.site1.com/rdf

```
:B1 :Title "Linked Data"
:B1 :Author "Lara T."
:B1 :PubYear 2008
:B1 :Publisher "Springer"
:B2 :Title "Data on the Web"
:B2 :Author "Abiteboul S."
```


www.site2.com/rdf

```
:A1 rdf:type bibo:Article
:A1 :Title "Data Web"
:A1 :Author "Tom Lara"
:A1 :Author "Bob Hacker"
:A1 :Year 2007
:A2 rdf:type bibo:Article
:A2 :Title "Semantic Web"
:A2 :Author "Tom Lara"
:A2 :Year 2005
```

- Interactive Query Formulation.
- MashQL queries are translated into and executed as SPARQL.

# Example 1

“Lara’s articles after 2007?”



**RDF Input**

From:

- 
- 

**Query**

Everything

www.site1.com/rdf

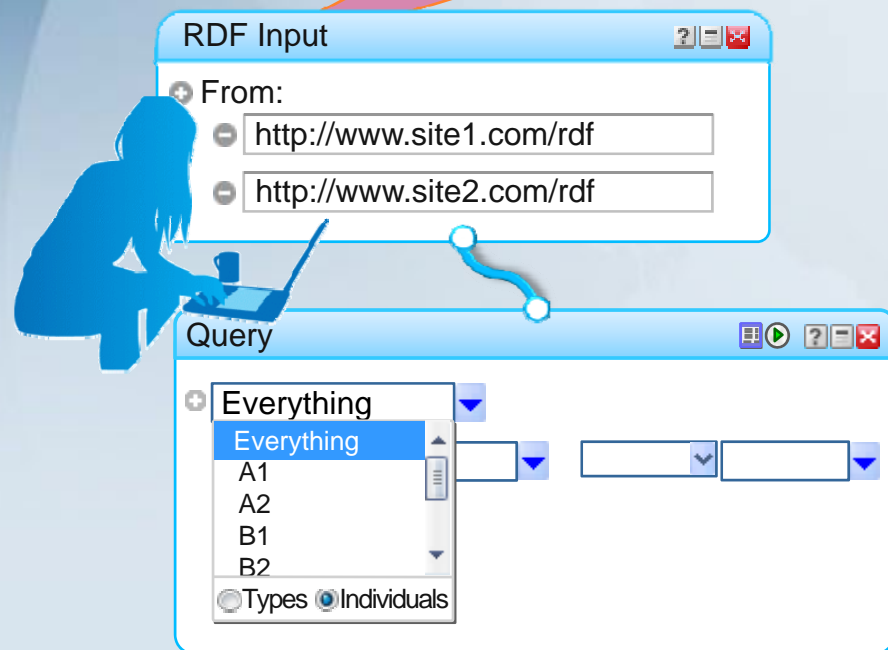
```
:B1 :Title "Linked Data"
:B1 :Author "Lara T."
:B1 :PubYear 2008
:B1 :Publisher "Springer"
:B2 :Title "Data on the Web"
:B2 :Author "Abiteboul S."
```

www.site2.com/rdf

```
:A1 rdf:type bibo:Article
:A1 :Title "Data Web"
:A1 :Author "Tom Lara"
:A1 :Author "Bob Hacker"
:A1 :Year 2007
:A2 rdf:type bibo:Article
:A2 :Title "Semantic Web"
:A2 :Author "Tom Lara"
:A2 :Year 2005
```

# Example 1

“Lara’s articles after 2007?”



www.site1.com/rdf

:B1 :Title "Linked Data"

:B1 :Author "Lara T."

:B1 :PubYear 2008

:B1 :Publisher "Springer"

:B2 :Title "Data on the Web"

:B2 :Author "Abiteboul S."

www.site2.com/rdf

:A1 ~~rdf:type~~ bibo:Article

:A1 :Title "Data Web"

:A1 :Author "Tom Lara"

:A1 :Author "Bob Hacker"

:A1 :Year 2007

:A2 ~~rdf:type~~ bibo:Article

:A2 :Title "Semantic Web"

:A2 :Author "Tom Lara"

:A2 :Year 2005


## Background queries

```
SELECT X WHERE { ?X ?P ?O }  
Union  
SELECT X WHERE { ?S ?P ?X }
```

```
SELECT X WHERE { ?S rdf:type ?X }
```

# Example 1

“Lara’s articles after 2007?”



**RDF Input**

From:

- http://www.site1.com/rdf
- http://www.site2.com/rdf

**Query**

Everything

Author

Author

Title

PubYear

Type

Year

Contains

Lara

Equals

Contains

OneOf

Not

Between

LessThan

MoreThan

www.site1.com/rdf

:B1 :Title "Linked Data"

:B1 :Author "Lara T."

:B1 :PubYear 2008

:B1 :Publisher "Springer"

:B2 :Title "Data on the Web"

:B2 :Author "Abiteboul S."

www.site2.com/rdf

:A1 ~~rdf:type~~ bibo:Article

:A1 :Title "Data Web"

:A1 :Author "Tom Lara"

:A1 :Author "Bob Hacker"

:A1 :Year 2007

:A2 ~~rdf:type~~ bibo:Article

:A2 :Title "Semantic Web"

:A2 :Author "Tom Lara"

:A2 :Year 2005

Background query


```
SELECT P WHERE {?Everything ?P ?O}
```

University of Cyprus © 2009



# Example 1

“Lara’s articles after 2007?”



**RDF Input**

From:

- 
- 

**Query**

Everything

- Author
- Year\PubYear 
  - Author
  - Title
  - PubYear
  - Type
  - Year

MoreTr 

- Equals
- Contains
- OneOf
- Not
- Between
- LessThan
- MoreThan

www.site1.com/rdf

:B1 :Title "Linked Data"  
:B1 :Author "Lara T."  
:B1 :PubYear 2008  
:B1 :Publisher "Springer"  
:B2 :Title "Data on the Web"  
:B2 :Author "Abiteboul S."

www.site2.com/rdf

:A1 ~~rdf:type~~ bibo:Article  
:A1 :Title "Data Web"  
:A1 :Author "Tom Lara"  
:A1 :Author "Bob Hacker"  
:A1 :Year 2007  
:A2 ~~rdf:type~~ bibo:Article  
:A2 :Title "Semantic Web"  
:A2 :Author "Tom Lara"  
:A2 :Year 2005

Background query

```
SELECT P WHERE {?Everything ?P ?O}
```

University of Cyprus © 2009

# Example 1

“Lara’s articles after 2007?”

RDF Input

From:

- http://www.site1.com/rdf
- http://www.site2.com/rdf

Query

Everything

- Author ^Lara
- Year\PubYear > 2007
- Title ☒ ArticleTitle

Title

- Author
- Title
- PubYear
- Type
- Year

www.site1.com/rdf

```
:B1 :Title "Linked Data"
:B1 :Author "Lara T."
:B1 :PubYear 2008
:B1 :Publisher "Springer"
:B2 :Title "Data on the Web"
:B2 :Author "Abiteboul S."
```

www.site2.com/rdf

```
:A1 rdf:type bibo:Article
:A1 :Title "Data Web"
:A1 :Author "Tom Lara"
:A1 :Author "Bob Hacker"
:A1 :Year 2007
:A2 rdf:type bibo:Article
:A2 :Title "Semantic Web"
:A2 :Author "Tom Lara"
:A2 :Year 2005
```

Background query

```
SELECT P WHERE {?Everything ?P ?O}
```

University of Cyprus © 2009



# Example 1

“Lara’s articles after 2007?”

RDF Input

+ From:

-

-

Query

+ Everything

+ Author

+ Year\PubYear

+ Title ☒ ArticleTitle

www.site1.com/rdf

:B1 :Title "Linked Data"  
:B1 :Author "Lara T."  
:B1 :PubYear 2008  
:B1 :Publisher "Springer"  
:B2 :Title "Data on the Web"  
:B2 :Author "Abiteboul S."

www.site2.com/rdf

:A1 ~~rdf:type~~ bibo:Article  
:A1 :Title "Data Web"  
:A1 :Author "Tom Lara"  
:A1 :Author "Bob Hacker"  
:A1 :Year 2007  
:A2 ~~rdf:type~~ bibo:Article  
:A2 :Title "Semantic Web"  
:A2 :Author "Tom Lara"  
:A2 :Year 2005

```
PREFIX S1: <http://www.site1.com/rdf>
PREFIX S2: <http://site2.com/rdf>
SELECT ?ArticleTitle
FROM <http://site1.com/rdf>
FROM <http://site2.com/rdf>
WHERE {
  {?X S1:Author ?X1} UNION {?X S2:Author ?X1}
  {?X S1:PubYear ?X2} UNION {?X S2:Year ?X2}
  {{?X S1:Title ?ArticleTitle} UNION {?X S2:Title
?ArticleTitle}}
  FILTER regex(?X1, "^Hacker")
  FILTER (?X2 > 2000)}
```

## Example 2

“Recent articles from Cyprus?”

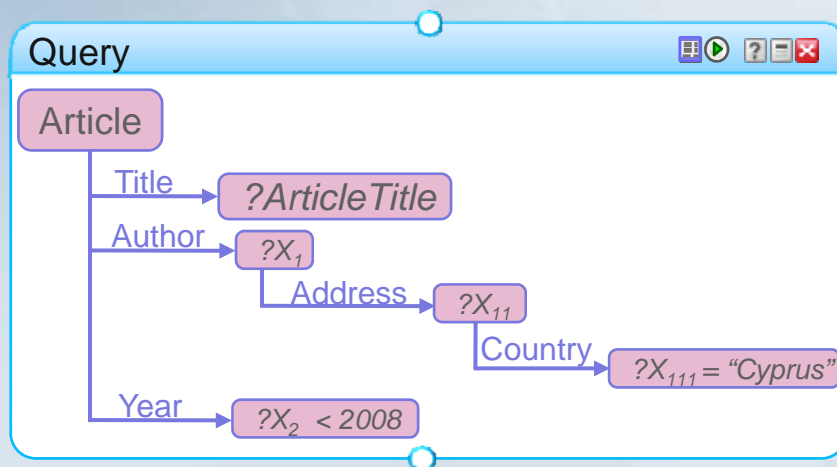


Query

- + Article
  - + Title ☒ *ArticleTitle*
  - + Author
    - + Address
      - + Country “Cyprus”
- + Year > 2008

→ Retrieve every Article that: has a title, written by author, who has address, this address has a country called Cyprus, and the article is published after 2008.

# The Intuition of MashQL



## A query is a tree

- The root is called the query *subject*.
- Each branch is a *restriction*.
- Branches can be expanded, (information path)
- Object value filters

Def. A **Query**  $Q$  with a subject  $S$ , denoted by  $Q(S)$ , is a set of restrictions on  $S$ .  $Q(S) = R_1 \text{ AND } \dots \text{ AND } R_n$ .

Dif. A **Subject**  $S \in (I \cup V)$ , where  $I$  is an identifier and  $V$  is a variable.

Dif. A **Restriction**  $R = \langle R_x, P, O_f \rangle$ , where  $R_x$  is an optional restriction prefix that can be (maybe | without),  $P$  is a predicate ( $P \in I \cup V$ ), and  $O_f$  is an object filter.

# The Intuition of MashQL

An Object filter is one of :

- Equals
- Contains
- MoreThan
- LessThan
- Between
- one of
- Not(f)
- Information Path (sub query)

Query

- + Article
  - + Title ☒ ArticleTitle
  - + Author
    - + Address
  - + Country "Cyprus"
- + Year > 2008

Def. An **object filter**  $O_f = \langle O, f \rangle$ , where  $O$  is an object and  $f$  is a filtering function one of :

$O_f = \langle O \rangle$ , where  $O$  is an object,  $O \in V \cup I$ .

$O_f = \langle O, \text{Equals}(X, T, L_t) \rangle$ , where  $X$  can be a variable or a constant,  $T$  is a datatype, and  $L_t$  is a language tag.

$O_f = \langle O, \text{Contains}(X, T, L_t) \rangle$ , where  $O$  is an object variable,  $X$  is a regex literal,  $T$  is a data type, and  $L_t$  is a language.

$O_f = \langle O, \text{MoreThan}(X, T) \rangle$ , where  $O$  is an object variable,  $X$  is a variable or a constant,  $T$  is a datatype.

$O_f = \langle O, \text{LessThan}(X, T) \rangle$ , where  $O$  is an object variable,  $X$  is a variable or a constant,  $T$  is a datatype identifier.

$O_f = \langle O, \text{Between}(X, Y, T) \rangle$ , where  $X$  and  $Y$  are variables or constants,  $T$  is a datatype identifier.

$O_f = \langle O, \text{OneOf}(V) \rangle$ , where  $O$  is an object variable, and  $V$  is a set of values  $\{v_1, \dots, v_n\}$ ,  $v_i$  is a variable or constant.

$O_f = \langle O, \text{Not}(f) \rangle$ , where  $f$  is one of the functions defined above.

$O_f = \langle O, Q_i(O) \rangle$ , where  $O$  is an object ( $O \in V \cup I$ ), and  $Q_i(O)$  is a sub-query with  $O$  being the query subject.

# More MashQL Constructs

## ➤ Resection Operators {Required, Maybe, or Without}

All restriction are required (i.e. AND), unless they are prefixed with “maybe” or “without”

The screenshot shows a window titled "Query" with a toolbar containing icons for list, play, help, and close. The query is defined as follows:

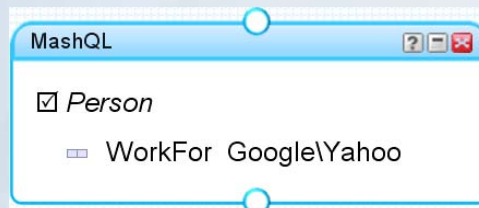
- + Song
  - + Title ☒ *SongTitle*
  - + Artist Shakeria
  - + **maybe** Album ☒ *AlbumTitle*
  - + **without** Copyright

```
SELECT ?SongTitle, AlbumTitle
WHERE {
  {?X :Title ?SongTitle.
   ?X :Artist :Shakeria
   Optional{?X :Album ?AlbumTitle}
   Optional{?X :Copyrihgt ?X2}
   Filter FILTER (!Bound(?X2)).
```

# More MashQL Constructs

- Union operator (denoted as “\”) between


## Objects



MashQL window showing object construction. The main area contains a tree structure: a checked box next to *Person*, and an unchecked box next to *WorkFor* with the value *Google\Yahoo* next to it.

```
SELECT ?Person
WHERE {
    ?Person :WorkFor :Google
    UNION
    ?Person WorkFor :Yahoo}
```

## Predicates



MashQL window showing predicate construction. The main area contains a tree structure: the text *Person* is shown, followed by an unchecked box next to *Surname\Firstname* and a checked box next to *FName*.

```
SELECT ?FName
WHERE {
    ?Person :Surname ?FName
    UNION
    ?Person :Firstname ?FName}
```

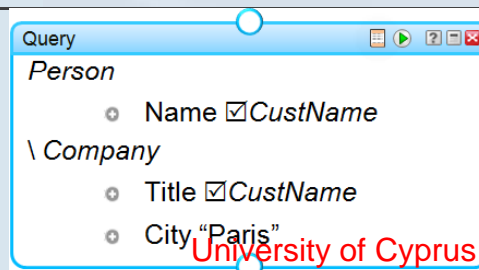
## Subjects



MashQL window showing subject construction. The main area contains a tree structure: the text *a Person\Company* is shown, followed by two unchecked boxes. The first box is next to *Name* with *AgentName* next to it. The second box is next to *Phone* with *AgentPone* next to it.

```
SELECT ?AgentName, ?AgentPhone
WHERE {
    {?Person rdf:type :Person.
    ?Person :Name ?AgentName.
    ?Person :Phone ?AgentPhone}
    UNION
    {?Company rdf:type :Company.
    ?Company :Name ?AgentName.
    ?Company :Phone ?AgentPhone}}
```

## Queries



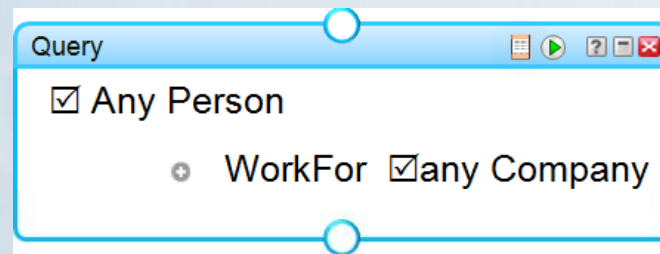
Query window showing query construction. The main area contains a tree structure: the text *Person* is shown, followed by a checked box next to *Name* with *CustName* next to it. Below this, the text *\ Company* is shown, followed by two checked boxes. The first box is next to *Title* with *CustName* next to it. The second box is next to *City* with the value *"Paris"* next to it.

```
SELECT ?CustName,
WHERE {
    ?Person :Name ?CustName.
    UNION
    {?Company :Title ?CustName.
    ?Company :City ?X1.
    FILTER regex(?X1, "Paris")}}
```

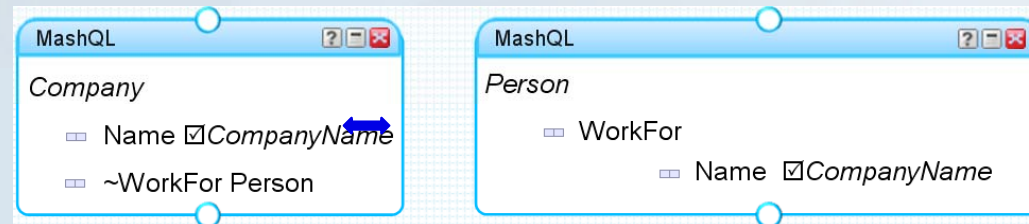
# More MashQL Constructs

And several other constructs, including:

- Types



- and Reverse Predicates



- Datatypes and Language Tags

- ....



# Formal Syntax and Semantics

**Def.1 (Dataset):** A dataset  $D$  is a set of triples, each triple  $t$  is formed as  $\langle S, P, O \rangle$ , where  $S \in I$ ,  $P \in I$ , and  $O \in I \cup L$ .

**Def.2 (Typed Literals):** Every object literal must have a datatype  $D$ : If  $O \in L$  then  $O \in D$ .

**Def.3 (Language Tags):** An object literal ( $O \in L$ ) may have a language tag  $L_t$ .

**Def. 4 (Query):** A Query  $Q$  with a subject  $S$ , denoted by  $Q(S)$ , is a set of restrictions on  $S$ .  $Q(S) = R_1 \text{ AND } \dots \text{ AND } R_n$ .

**Def. 5 (Subject):** A subject  $S \in (I \cup V)$ , where  $I$  is an identifier and  $V$  is a variable.

**Def. 6 (Restriction):** A restriction  $R = \langle R_x, P, O_f \rangle$ , where  $R_x$  is an optional restriction prefix that can be (maybe | without),  $P$  is a predicate ( $P \in I \cup V$ ), and  $O_f$  is an object filter.

**Def.7 (Object Filter):** An object filter  $O_f = \langle O, f \rangle$ , where  $O$  is an object and  $f$  is a filtering function. An object filter can have one of the following nine forms:

1.  $O_f = \langle O \rangle$ , where  $O$  is an object,  $O \in V \cup I$ . This is the simplest object filter, i.e., it does not add any restriction on the object value of the retrieved triples.
2.  $O_f = \langle O, \text{Equals}(X, T, L_t) \rangle$ , where  $X$  can be a variable or a constant,  $T$  is a datatype, and  $L_t$  is a language tag. This filter restricts the retrieved results, such that, the object value  $O$  should be equal to  $X$ , with datatype  $T$ , and with language  $L_t$ .
3.  $O_f = \langle O, \text{Contains}(X, T, L_t) \rangle$ , where  $O$  is an object variable,  $X$  is a regex literal,  $T$  is a data type, and  $L_t$  is a language. This filter restricts the retrieved results, such that, the object value  $O$  should be equal to  $\text{regex}(X)$ , with datatype  $T$ , and with language  $L_t$ . A regex literal is a literal that contains a regular expression matching pattern.
4.  $O_f = \langle O, \text{MoreThan}(X, T) \rangle$ , where  $O$  is an object variable,  $X$  is a variable or a constant,  $T$  is a datatype. This filter restricts the retrieved results, such that, the object value  $O$  should be more than  $X$  and with datatype  $T$ .
5.  $O_f = \langle O, \text{LessThan}(X, T) \rangle$ , where  $O$  is an object variable,  $X$  is a variable or a constant,  $T$  is a datatype identifier. This filter restricts the retrieved results, such that, the object value  $O$  should be less than  $X$  and with datatype  $T$  (see rule-9).
6.  $O_f = \langle O, \text{Between}(X, Y, T) \rangle$ , where  $X$  and  $Y$  are variables or constants,  $T$  is a datatype identifier. This filter restricts the retrieved results, such that, the object value  $O$  should be more than or equals  $X$ , less than or equals  $Y$ , and with datatype  $T$ .
7.  $O_f = \langle O, \text{OneOf}(V) \rangle$ , where  $O$  is an object variable, and  $V$  is a set of values  $\{v_1, \dots, v_n\}$ ,  $v_i$  is a variable or constant. This filter restricts the retrieved results, such that, the object value  $O$  should be equal to one of the values in  $V$ .
8.  $O_f = \langle O, \text{Not}(f) \rangle$ , where  $f$  is one of the functions defined above. This filter extends all of the above functions with simple negation. The filter is same as the *Equals* filter but with negation, i.e., *Not Equal*.
9.  $O_f = \langle O, Q_i(O) \rangle$ , where  $O$  is an object ( $O \in V \cup I$ ), and  $Q_i(O)$  is a sub-query with  $O$  being the query subject. The restrictions defined in the sub-query  $Q_i(O)$  should be satisfied as well. Notice that this definition is recursive; however, this does not mean the query itself is recursive.

**Def.8 (Types):** A subject ( $S \in I$ ) or an object ( $O \in I$ ) can be prefixed with “a” or “an” to mean the instances of this subject/object type, instead of the subject/object itself.

**Def.9 (Union):** A union can be declared between objects, predicates, subjects and/or queries, in the following forms:

1.  $O_n = \langle O_1 \setminus O_2 \setminus \dots \setminus O_n \rangle$ , to indicate unions between objects, where  $O_i \in I$ .
2.  $P_n = \langle P_1 \setminus P_2 \setminus \dots \setminus P_n \rangle$ , to indicate unions between predicates, where  $P_i \in I$ .
3.  $S_n = \langle S_1 \setminus S_2 \setminus \dots \setminus S_n \rangle$ , to indicate unions between subjects, where  $S_i \in I$ .
4.  $Q_n = \langle Q_1 \setminus Q_2 \setminus \dots \setminus Q_n \rangle$ , to indicate unions between queries.

**Def.10 (Reverse):**  $\langle \sim P \rangle$  indicates the reverse of the predicate  $P$ . Let  $R_1$  be a restriction on  $S$  such that  $\langle S P O \rangle$ , and  $R_2$  be  $\langle O \sim P S \rangle$ ,  $R_1$  and  $R_2$  have the same meaning.



# Query Formulation Algorithm

## Formalization of the background queries

### Select Subject S

$$(1) S \in S_T : \pi_O(\sigma_{P=:Type'}(D))$$

$$(2) S \in S_I : \pi_S(D) \cup \pi_O(\sigma_{O \in}(D))$$

$$(3) S \in V$$

$$(1') O1 : \{ (?S1 <:Type> ?O1) \}$$

$$(2') S1 : \{ (?S1 ?P1 ?O1) \} \text{ UNION } O1 : \{ (?S1 ?P1 ?O1) \}. \text{ Filter isURI(?O1) \}$$

### Select a property P

$$(4) (S \in S_T) \rightarrow P \in \pi_{P2}(\sigma_{P1=:Type' \wedge O1=Subject}(D) \bowtie_{S1=S2} \sigma(D))$$

$$(5) (S \in S_I) \rightarrow P \in \pi_P(\sigma_{S=Subject}(D))$$

$$(6) (S \in V) \rightarrow P \in \pi_P(\sigma(D))$$

$$(7) P \in V$$

$$(4') P2 : \{ (?S1 <:Type> <S>)(?S2 ?P2 ?O2) \}$$

$$(5') P1 : \{ (<S> ?P1 ?O1) \}$$

$$(6') P1 : \{ (?S1 ?P1 ?O1) \}$$

### Add a filter on P

$$(8) (S \in S_I) \wedge (P \in V) \rightarrow O \in \pi_{O1}(\sigma_{S1=S \wedge O1 \in}(D))$$

$$(9) (S \in S_I) \wedge (P \notin V) \rightarrow O \in \pi_{O1}(\sigma_{S1=S \wedge P1=P \wedge O1 \in}(D))$$

$$(10) (S \in S_T) \wedge (P \in V) \rightarrow O \in \pi_{O2}(\sigma_{P1=:Type' \wedge O1=S}(D) \bowtie_{S1=S2} \sigma(D))$$

$$(11) (S \in S_T) \wedge (P \notin V) \rightarrow O \in \pi_{O2}(\sigma_{P1=:rdf:Type' \wedge O1=S}(D) \bowtie_{S1=S2} \sigma_{P2=P}(D))$$

$$(12) (S \in V) \wedge (P \in V) \rightarrow O \in \pi_O(\sigma(D))$$

$$(13) (S \in V) \wedge (P \notin V) \rightarrow O \in \pi_O(\sigma_{P=P}(D))$$

$$(8') O1 : \{ (<S> ?P1 ?O1) \text{ Filter isURI(?O1)} \}$$

$$(9') O1 : \{ (<S> <P> ?O1) \text{ Filter isURI(?O1)} \}$$

$$(10') O1 : \{ (?S1 <:Type> <S>)(?S1 ?P2 ?O2) \}$$

$$(11') O : \{ (?S <rdf:Type> <S>)(?S <P> ?O) \}$$

$$(12') O1 : \{ (?S1 ?P1 ?O1) \}$$

$$(13') O1 : \{ (?S1 <P> ?O1) \}$$

# MashQL-SPARQL Mapping Rules

- Rule-1: The symbol  $\square$  before a variable means that it will be returned in the results; i.e., included in the SELECT part of in SPARQL. If the output of the query is input to another, use "CONSTRUCT \*".
- Rule-2: In any of the following rules, if a subject, predicate, or object is italicized: it is seen as a SPARQL variable, i.e. prefixed with "?".
- Rule-3: If S is a subject and  $R = <, P, Of>$ , the mapping is:  $\{S P O\}$ .
- Rule-4: If S is a subject and  $R = <\text{maybe}, P, Of>$ , the mapping is:  $\{OPTIONAL\{S P O\}\}$ .
- Rule-5: If S is a subject and  $R = <\text{without}, P, Of>$ , the mapping is:  $\{S P O. FILTER (!bound(?O))\}$ .
- Rule 6. If  $Of = <O, Equals(X, T, Lt)>$ :  
Append the mapping with:  $FILTER(?O = X)$   
If  $T \neq \text{Null}$ : Append the mapping with:  $FILTER(datatype(?O)=T)$   
If  $Lt \neq \text{Null}$ : Append the mapping with:  $FILTER(land(?O) = Lt)$
- Rule 7. If  $Of = <O, Correlation(X, Y, T)>$ :  
Append the mapping with:  $FILTER(?O = X)$   
If  $T \neq \text{Null}$ : Append the mapping with:  $FILTER(datatype(?O)=T)$   
If  $Lt \neq \text{Null}$ : Append the mapping with:  $FILTER(land(?O) = Lt)$
- Rule 8. If  $Of = <O, Between(X, Y, T)>$ :  
Append the mapping with:  $FILTER(?O >= X \&\& FILTER(?O <= Y)$   
If  $T \neq \text{Null}$ : Append the mapping with:  $FILTER(datatype(?O)=T)$
- Rule 9. If  $Of = <O, LessThan(X, Y, T)>$ :  
Append the mapping with:  $FILTER(?O < X)$   
If  $T \neq \text{Null}$ : Append the mapping with:  $FILTER(datatype(?O)=T)$
- Rule 10. If  $Of = <O, Between(X, Y, T)>$ :  
Append the mapping with:  $FILTER(?O >= X) \&\& FILTER(?O <= Y)$   
If  $T \neq \text{Null}$ : Append the mapping with:  $FILTER(datatype(?O)=T)$
- Rule 11. If  $Of = <O, OneOf(V)>$ :  
Append the mapping with:  $\{FILTER(?O = V1) || \dots || FILTER(?O = Vn)\}$   
If  $V_i$  is a regex-ed literal, the  $i$ th filter above should be replaced with:  $FILTER Regex(?O, V_i)$
- Rule 12. If  $Of = <O, Not(f)>$ : The  $f$  filter will be generated as above, but with a negation.
- Rule 13. If  $Of = <O, Qi(O)>$ : Repeat all mapping rules to generate  $Qi(O)$ .
- Rule 14. If a subject S is prefixed with "a" or "an": Append the mapping with:  $\{?S rdf:type :S\}$
- Rule 15. If an object O is prefixed with "a" or "an": Append the mapping with:  $\{?O rdf:type :O\}$
- Rule 16. Given  $On$ , If  $n > 1$  and  $O_i \in I$ : The mapping in rules 3-4 will be:  $\{S P :O_1\} \cup \dots \cup \{S P :O_n\}$
- Rule 17. Given  $Pn$ , If  $n > 1$  and  $P_i \in I$ : The mapping in rules 3-4 will be:  $\{S :P_1 O\} \cup \dots \cup \{S :P_n O\}$
- Rule 18. Given  $Sn$ , If  $n > 1$  and  $S_i \in I$ : Regenerate the query  $n$  times, each time with  $S_i$  as a root, and with a UNION between the queries.
- Rule 19. Given  $Qn$ , If  $n > 1$ : Add UNION between the  $n$  queries.
- Rule 20. If S is a subject and  $R = <\sim P, O>$ , the mapping is:  $\{O P S\}$ .

➔ Also mapped into  
Oracle's SPARQL

# MashQL Editor

MashQL : editing pipe - Mozilla firefox

http://192.168.1.2:8990/MashQL-MashQL-context-root/test/editor1.jsp

MashQL Pipes

MashQL JobSeeking

Welcome Demo

Publish New Save Properties LOGOUT

RDF Input

From

http://RDFDistiller.ucy.ac.cy/R

RDF Input

From

http://RDFDistiller.ucy.ac.cy/

Output

- Alpha version (will be public soon)
- Web Ajax-based.
- Open sources Java Script libraries (from Yahoo)
- Oracle 11g as RDF store.
- **Graphs Summaries** for fast user-interaction.
- URI Normalization based on heuristics.  
(but some URIs are too cryptic)

# MashQL Firefox Add-On (Light-mashups @ your browser)

Google - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://scholar.google.com/scholar?q=mustafa+jarrar&hl=en&lr=&btnG=Search

**MashQL** x

Sources:

- http://scholar.google.com/schol
- C:\MashQL\MyMashups\MyA

Filter:

— Filter: —

— Stored Mashups —

**Google Scholar** x

Web Images Video News Maps more »

mustafa jarrar Search

Advanced Scholar Search  
Scholar Preferences  
Scholar Help

**Scholar** All articles - [Recent articles](#) Results 1 - 100 of about 307 for mustafa jarrar. (0.44 sec)

[Data modelling versus ontology engineering](#) - [uni-konstanz.de](#) [PDF]  
P Spyns, R Meersman, M Jarrar - ACM SIGMOD Record, 2002 - portal.acm.org  
ABSTRACT Ontologies in current computer science parlance are computer based resources that represent agreed domain semantics. Unlike data models, the fundamental asset of ontologies is their relative independence of ...  
[Cited by 165](#) - [Related articles](#) - [Web Search](#) - [Import into BibTeX](#) - [BL Direct](#) - [All 14 versions](#)

[On Using Conceptual Data Modeling for Ontology Engineering](#) - [jarrar.info](#) [PDF]  
M JARRAR, JAN DEMEY, R MEERSMAN - papers.ssm.com  
Abstract. This paper tackles two main disparities between conceptual data schemes and ontologies, which should be taken into account when (re)using conceptual data modeling techniques for building ontologies. Firstly, ...  
[Cited by 73](#) - [Related articles](#) - [Web Search](#) - [Import into BibTeX](#) - [BL Direct](#) - [All 7 versions](#)

[Formal Ontology Engineering in the DOGMA Approach](#) - [epfl.ch](#) [PDF]  
M Jarrar, R Meersman - LECTURE NOTES IN COMPUTER SCIENCE, 2002 - Springer  
Abstract. This paper presents a specifically database-inspired approach (called DOGMA) for engineering formal ontologies, implemented as shared resources used to express agreed formal semantics for a real world domain. We address ...  
[Cited by 66](#) - [Related articles](#) - [Web Search](#) - [Import into BibTeX](#) - [BL Direct](#) - [All 9 versions](#)

[OntoWeb-A Semantic Web Community Portal](#) - [vub.ac.be](#) [PDF]  
... , D Oberle, R Volz, J Zheng, M Jarrar, Y Sure, R ... - LECTURE NOTES IN COMPUTER SCIENCE, 2002  
http://scholar.google.com/scholar?q=mustafa+jarrar&hl=e

# Evaluation (DBLP, Experiment 1)

## (User Interaction Response Time)

How long it takes to generate the next list?

Query

Any  <sup>1</sup>

<sup>2</sup>

<sup>2</sup>

<sup>3</sup>  <sup>4</sup>

<sup>3</sup>

<sup>2</sup>

❶  $O: (?S \text{ Type } ?O)$

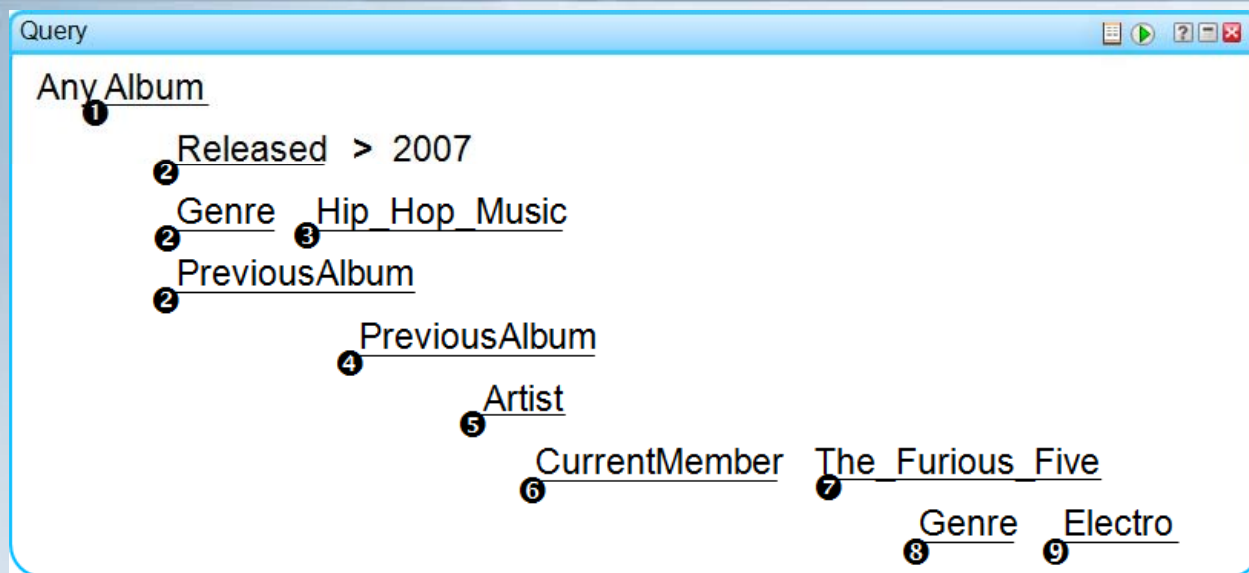
❷  $P: (?S \text{ Type Article}) (?S ?P ?O1)$

❸  $P: (?S \text{ Type Article})$   
 $(?S \text{ Creator } ?O1) (?O1 ?P ?O2)$

❹  $O: (?S \text{ Type Article})$   
 $(?S \text{ Creator } ?O1) (?O1 \text{ Type } ?O)$

Query	DBLP (9M triples)		DBLP (4M triples)		DBLP (2M triples)	
	GS	Oracle	GS	Oracle	GS	Oracle
Q <sub>1</sub>	0.003	0.005	0.003	0.004	0.003	0.003
Q <sub>2</sub>	0.001	0.136	0.001	0.148	0.001	0.108
Q <sub>3</sub>	0.001	0.187	0.001	0.846	0.001	0.671
Q <sub>4</sub>	0.001	1.208	0.001	0.835	0.001	0.650

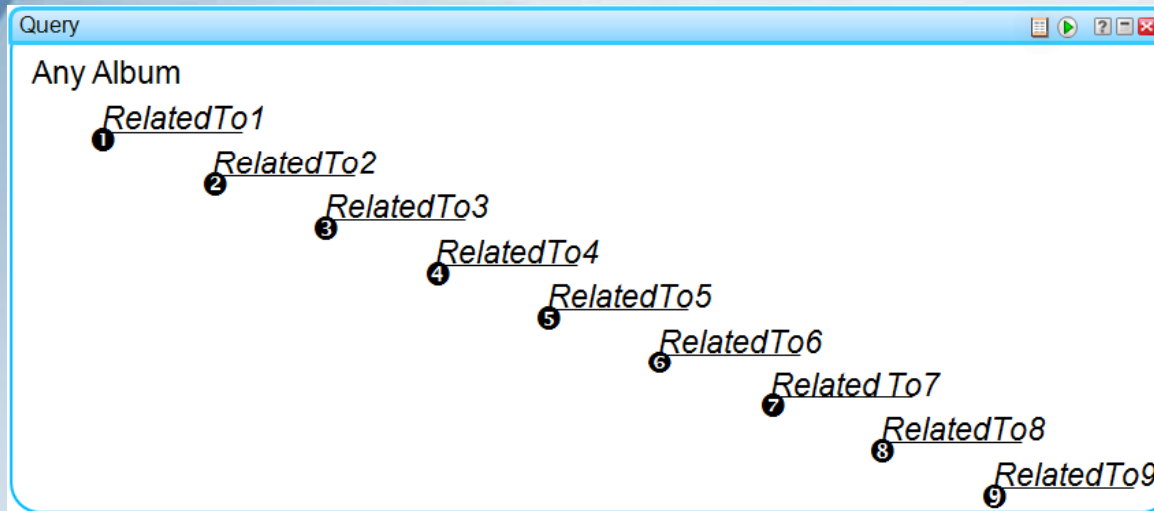
# Evaluation (DBPedia, Experiment 2)



4 P:(?S :Type :Album)  
 (?S :PreviousAlbum ?O1)  
 (?O1 ?P ? O2)

Query	DBPedia (32 M)	DBPedia (16 M)	DBPedia (8 M)	DBPedia (4 M)
Q <sub>1</sub>	0.003	0.003	0.003	0.003
Q <sub>2</sub>	0.002	0.002	0.002	0.002
Q <sub>3</sub>	0.005	0.004	0.003	0.003
Q <sub>4</sub>	0.005	0.004	0.004	0.004
Q <sub>5</sub>	0.005	0.004	0.004	0.004
Q <sub>6</sub>	0.005	0.005	0.005	0.005
Q <sub>7</sub>	0.007	0.007	0.007	0.007
Q <sub>8</sub>	0.005	0.005	0.005	0.005
Q <sub>9</sub>	0.007	0.007	0.007	0.006

# Evaluation (DBPedia, Experiment 3)



③ P3:(?S Type ?Album)  
 (?S ?RelatedTo1 ?O1)  
 (?O1 ?RelatedTo2 ?O2)  
 (?O2 ?P3 ?O4)

Query	(B32) 32 M		DBPedia (16 M )	DBPedia (8 M )	DBPedia (4 M )
	GS	Oracle			
Q <sub>1</sub>	0.001	0.027	0.001	0.001	0.001
Q <sub>2</sub>	0.026	17.450	0.010	0.006	0.005
Q <sub>3</sub>	0.048	49.656	0.010	0.011	0.010
Q <sub>4</sub>	0.087	5348.7	0.022	0.017	0.011
Q <sub>5</sub>	0.135	-	0.036	0.032	0.011
Q <sub>6</sub>	0.185	-	0.055	0.047	0.016
Q <sub>7</sub>	0.234	-	0.076	0.061	0.023
Q <sub>8</sub>	0.265	-	0.098	0.077	0.027
Q <sub>9</sub>	0.280	-	0.110	0.092	0.034



# Conclusions

- “Query and mash up the Data Web as simple as filtering up Web Feeds” is a query formulation problem.
- End-users can navigate, query, and mash up unknown graphs.  
without knowing the schema. Data is schema-free. Multiple sources.
- MashQL is expressive as SPARQL.  
Except NAMED GRAPH.
- MashQL is not merely a SPARQL interface, or limited RDF.  
It has its own path-pattern intuition (can be similarly used for XML and DB).



## Future Work



- Reasoning, Keyword Search, Aggregation Functions, etc.
- Results Presentation (Should be tacked fundamentally).
- Firefox add-on Mashup/query editor.
- RDF summaries for SPARQL optimization.
- ..



# Thank You

Dr. Mustafa Jarrar

[mjarrar@cs.ucy.ac.cy](mailto:mjarrar@cs.ucy.ac.cy)

HPCLab, University of Cyprus