PhD Defense

On the foundations for the compilation of web data queries: optimization and distributed evaluation of SPARQL.

Date: 13 September, 2018
Defendant: Louis JACHIET
Directors: Nabil LAYAÏDA
Pierre GENEVÈS
Reviewers: Dario COLAZZO
Ioana MANOLESCU
Jury: Jérôme EUZENAT
Patrick VALDURIEZ
PhD Defense

On the foundations for the compilation of web data queries: optimization and distributed evaluation of SPARQL.

Date: 13 September, 2018
Defendant: Louis JACHIET
Directors: Nabil LAYAÏDA
Pierre GENEVÈS
Reviewers: Dario COLAZZO
Ioana MANOLESCU
Jury: Jérôme EUZENAT
Patrick VALDURIEZ
What are the museums in Grenoble?

Who are they exposing?
Musée de Grenoble

Le musée de Grenoble, créé en 1798, est le principal musée d'art et d'Antiquités de la Ville de Grenoble, en Isère. Situé place Lavalette, à l'emplacement d'un ancien couvent des Franciscains édifié en 1218 et dont le site deviendra militaire à la fin du xviᵉ siècle, il fait partie des premiers musées d'art français et conserve l'une des plus belles collections d'art ancien.

Toutes les tendances et mouvements de la peinture sont présentés, tels le fauvisme avec des tableaux de Henri Matisse (8 peintures), André Derain, Albert Marquet, Raoul Dufy, Maurice de Vlaminck, Emile Othon Friesz (6 peintures), Jean Puy, Charles Camoin (Nu à la chemise mauve, acquis en 2012) Kees van Dongen, le cubisme avec Georges Braque, Albert Gleizes, André Lhote, Fernand Léger et Le Corbusier, l'école de Paris, représentée par Amedeo Modigliani, Pinchus Krémègne, Chaïm Soutine, Maurice Utrillo ainsi que Marc Chagall. Quatre peintures
The goals of a data standard are to:

* Encode data in a machine readable & processable way
* Allow exchange of data on the web
* Facilitate querying
RDF: Resource Description Framework [RCM14]

In RDF, data is represented as entities and as statements expressing relationships between these entities.
RDF: Resource Description Framework [RCM14]

In RDF, data is represented as entities and as statements expressing relationships between these entities.

In N-Triples format:

<table>
<thead>
<tr>
<th>subject</th>
<th>predicate</th>
<th>object</th>
</tr>
</thead>
<tbody>
<tr>
<td>:Musée_de_Grenoble</td>
<td>:isA</td>
<td>:Museum</td>
</tr>
<tr>
<td>:Musée_de_Grenoble</td>
<td>:locatedIn</td>
<td>:Grenoble</td>
</tr>
<tr>
<td>:Musée_de_Grenoble</td>
<td>:exhibits</td>
<td>:Chagall</td>
</tr>
<tr>
<td>:Musée_de_Grenoble</td>
<td>:exhibits</td>
<td>:Fantin-Latour</td>
</tr>
<tr>
<td>:Perret_Tower</td>
<td>:locatedIn</td>
<td>:Grenoble</td>
</tr>
<tr>
<td>:Louvre</td>
<td>:isA</td>
<td>:Museum</td>
</tr>
</tbody>
</table>
RDF: Resource Description Framework [RCM14]

In RDF, data is represented as entities and as statements expressing relationships between these entities.

Viewed as graphs:
Triple Pattern (TP)

A Triple Path is a triple \((s, p, o)\) where \(s\), \(p\), \(o\) are constants or variables.

In SPARQL:

```
```

Solution:

\((?\text{What} \rightarrow \text{Musée_de_Grenoble})\)
\((?\text{What} \rightarrow \text{Louvre})\)
Triple Pattern (TP)

A Triple Path is a triple \((s, p, o)\) where \(s, p, o\) are constants or variables.

“What are the museums?”
Triple Pattern (TP)

A Triple Path is a triple \((s, p, o)\) where \(s, p, o\) are constants or variables.

“What are the museums?”

In SPARQL:

```
```
Triple Pattern (TP)

A Triple Path is a triple \((s, p, o)\) where \(s\), \(p\), \(o\) are constants or variables.

“What are the museums?”

In SPARQL:

Triple Pattern (TP)

A Triple Path is a triple \((s,p,o)\) where \(s\), \(p\), \(o\) are constants or variables.

“What are the museums?”

In SPARQL:

\[
?\text{what} :\text{isA} :\text{Museum} .
\]

Solution:

\[
(?\text{What} \rightarrow :\text{Musée_de_Grenoble})
\]

\[
(?\text{What} \rightarrow :\text{Louvre})
\]
Basic Graph Patterns (BGP)

Basic Graph Patterns are conjunctions of Triple Patterns.
Basic Graph Patterns (BGP)

Basic Graph Patterns are conjunctions of Triple Patterns.

“What are the museums in Grenoble exposing Chagall?”
Basic Graph Patterns (BGP)

Basic Graph Patterns are conjunctions of Triple Patterns.

“What are the museums in Grenoble exposing Chagall?”

In SPARQL:

```sparql
```
Basic Graph Patterns (BGP)

Basic Graph Patterns are conjunctions of Triple Patterns.

“What are the museums in Grenoble exposing Chagall?”

In SPARQL:

```sparql
```
Basic Graph Patterns (BGP)

Basic Graph Patterns are conjunctions of Triple Patterns.

“What are the museums in Grenoble exposing Chagall?”

In SPARQL:

```
```

Solution:

```
(?What → :Musée_de_Grenoble)
```
“What are the museums in Grenoble and who are they exposing?”
“What are the museums in Grenoble and who are they exposing?”

In SPARQL:

```sparql
```
“What are the museums in Grenoble and who are they exposing?”

In SPARQL:

```
```

Solution:

```
(?What → :Musée_de_Grenoble; ?artist → :Chagall)
```
Other graph patterns in SPARQL 1.0 [PS+08]

* Triple Patterns

* Conjunction

Basic Graph Patterns
Other graph patterns in SPARQL 1.0 [PS+08]

* Triple Patterns

* Conjunction

* Disjunction

Museums or Trekking paths
Other graph patterns in SPARQL 1.0 [PS⁺08]

* Triple Patterns

* Conjunction

* Disjunction

* Filters

Museums with artists born before 1900
Other graph patterns in SPARQL 1.0 [PS⁺08]

* Triple Patterns

* Conjunction

* Disjunction

* Filters

* Conditional optionals

Missing information
Other graph patterns in SPARQL 1.0 [PS+08]

- Triple Patterns
- Conjunction
- Disjunction
- Filters
- Conditional optionals
- Changing graphs, etc.
Novelties of SPARQL 1.1 [HSP13]

* Expressions

Compute the age of monuments
Novelties of SPARQL 1.1 [HSP13]

* Expressions

* Minus, Exists

CITIES WITH MUSEUMS BUT WITHOUT OPERAS
Novelties of SPARQL 1.1 [HSP13]

* Expressions

* Minus, Exists

* Group by & Aggregation

Count the number of museums per city
Novelties of SPARQL 1.1 [HSP13]

* Expressions

* Minus, Exists

* Group by & Aggregation

  Count the number of museums per city

* Property Paths
“What are the museums in France?”
"What are the museums in France?"

![Diagram of museum locations in France]
"What are the museums in France?"

What :isA Museum :locatedIn France

Musée_de_Grenoble :locatedIn Grenoble :locatedIn Isère :isA Museum :locatedIn France

ARA :locatedIn France
“What are the museums in France?”

Property Path (PP)

A Property Path is a triple \((s, r, o)\) where \(r\) is a path expression.
“What are the museums in Grenoble and who are they exposing?”
“What are the museums in Grenoble and who are they exposing?”
"What are the museums in Grenoble and who are they exposing?"
“What are the museums in Grenoble and who are they exposing?”

(?What → :Musée_de_Grenoble; ?artist → :Chagall)
"What are the museums in Grenoble and who are they exposing?"

(?What → :Musée_de_Grenoble; ?artist → :Chagall)

\[O\left(\#\text{nodes}^{\#\text{variables}}\right)\] checks!
Compute solution to each individual TP

Combine individual solutions

\[(TP1 \bowtie TP2) \bowtie TP3\]
Compute solution to each individual TP

Combine individual solutions

\((TP1 \bowtie TP2) \bowtie TP3\)

\(O(\#\text{edges}) \text{ per TP}\)

\(O(|A| + |B| + |A \bowtie B|) \text{ per } A \bowtie B\)
(BGP, \bowtie) as an algebraic structure:

* \bowtie is associative

\[ TP_1 \bowtie (TP_2 \bowtie TP_3) = (TP_1 \bowtie TP_2) \bowtie TP_3 \]

* \bowtie is commutative

\[ TP_1 \bowtie TP_2 = TP_2 \bowtie TP_1 \]

An optimization method:

Generate all the equivalent terms, run the most efficient.
The relational algebra [Cod70]

* Base relations
* Combined through a set of operators (⋈, ∪, σ, etc.)
The relational algebra [Cod70]

* Base relations
* Combined through a set of operators (⋈, ∪, σ, etc.)

Optimization of relational languages

1. Generate equivalent terms
2. Select an estimated most efficient
3. Execute it
The relational algebra [Cod70]

* Base relations
* Combined through a set of operators (⋈, ∪, σ, etc.)

Optimization of relational languages

1. Generate equivalent terms
2. Select an estimated most efficient
3. Execute it

Problems:

* Not specifically for graphs
The relational algebra [Cod70]

* Base relations
* Combined through a set of operators (⋈, ∪, σ, etc.)

Optimization of relational languages

1. Generate equivalent terms
2. Select an estimated most efficient
3. Execute it

Problems:

* Not specifically for graphs
* Mismatchs in the semantics
The relational algebra [Cod70]

* Base relations
* Combined through a set of operators (⋈, ∪, σ, etc.)

Optimization of relational languages

1. Generate equivalent terms
2. Select an estimated most efficient
3. Execute it

Problems:

* Not specifically for graphs
* Mismatches in the semantics
* Poor optimization of Property Paths
Introduction / Query evaluation / Relational algebra

The relational algebra [Cod70]

* Base relations
* Combined through a set of operators (⋈, ∪, σ, etc.)

Optimization of relational languages

1. Generate equivalent terms
2. Select an estimated most efficient
3. Execute it

Problems:

* Not specifically for graphs
* Mismatches in the semantics
* Poor optimization of Property Paths
1. Generate equivalent terms
2. Select an estimated most efficient
3. Execute it
Contributions / The $\mu$-algebra / A new algebra

The $\mu$-algebra:

* a variation of the relational algebra
* equipped with fixpoints
* matches the SPARQL semantics
Contributions / The $\mu$-algebra / Syntax

$$\varphi ::= \begin{array}{l}
\varphi_1 \cup \varphi_2 \\
\varphi_1 \setminus \varphi_2 \\
\varphi_1 - \varphi_2 \\
\varphi_1 \setminus \varphi_2 \\
\varphi_1 \ni \varphi_2 \\
\varphi_1 \ni \varphi_2 \\
\rho^b_a (\varphi) \\
\pi_a (\varphi) \\
\beta^b_a (\varphi) \\
\theta (\varphi, g : C \to D) \\
\Theta (\varphi, g, C, D) \\
\sigma_{filter} (\varphi) \\
\mu (X = \varphi) \\
\text{let } (X = \varphi) \text{ in } \psi \\
X \\
\emptyset \\
|c_1 \to v_1, \ldots, c_n \to v_n|
\end{array}$$

- $\varphi$ : ::= formula
- $\cup$ : union
- $\setminus$ : normal minus
- $-$ : set minus
- $\setminus$ : strict minus
- $\ni$ : left-join
- $\ni$ : join
- $\rho^b_a (\varphi)$ : column exchange (or rename)
- $\pi_a (\varphi)$ : projection
- $\beta^b_a (\varphi)$ : column multiplying
- $\theta (\varphi, g : C \to D)$ : apply a function to mappings
- $\Theta (\varphi, g, C, D)$ : reduce
- $\sigma_{filter} (\varphi)$ : row filtering
- $\mu (X = \varphi)$ : fixpoint
- $\text{let } (X = \varphi) \text{ in } \psi$ : let-binder
- $X$ : variable
- $\emptyset$ : no mapping
- $|c_1 \to v_1, \ldots, c_n \to v_n|$ : a mapping
\[ \varphi ::= \]

\[
\begin{align*}
\varphi_1 \cup \varphi_2 \\
\varphi_1 \setminus \varphi_2 \\
\varphi_1 - \varphi_2 \\
\varphi_1 \setminus \varphi_2 \\
\varphi_1 \times \varphi_2 \\
\varphi_1 \varpi \varphi_2 \\
\rho^b_a(\varphi) \\
\pi_a(\varphi) \\
\beta^b_a(\varphi) \\
\theta(\varphi, g : C \rightarrow D) \\
\Theta(\varphi, g, C, D) \\
\sigma_{\text{filter}}(\varphi) \\
\mu(X = \varphi) \\
\text{let } (X = \varphi) \text{ in } \psi \\
X \\
\emptyset \\
|c_1 \rightarrow v_1, \ldots, c_n \rightarrow v_n|
\end{align*}
\]
The $\mu$-algebra / Translation

\[ \text{Path} \]

?s \rightarrow ?o
\[
\text{Tr}(A / B) = \text{Tr}(A) \text{mod} \left( \text{Tr}(B) \right)
\]
\[ Tr(A/B) = \pi_m \left( \rho_o^m(Tr(A)) \otimes \rho_s^m(Tr(B)) \right) \]
$A^* = \text{Empty Path or Path of } A^*/A$
\[ A^* = \text{Empty Path or Path of } A^*/A \]

\[ Tr(A^*) = \text{EmptyPath} \cup Tr(A^*)/A \]
\[ A^* = \text{Empty Path or Path of } A^*/A \]

\[
\begin{align*}
\text{Tr}(A^*) &= \text{EmptyPath} \cup \text{Tr}(A^*)/A \\
&= \mu \left( X = \text{EmptyPath} \cup X/A \right)
\end{align*}
\]
A^* = Empty Path or Path of A^*/A

\[ Tr(A^*) = \mu\left( X = \text{EmptyPath} \cup \frac{Tr(A^*)}{A} \right) \]

\[ = \mu\left( X = \beta_s^o(AllNodes) \cup \pi_m\left( \rho_o^m(X) \bowtie \rho_s^m(Tr(A)) \right) \right) \]
Contributions / The $\mu$-algebra / An example

$\mathbb{N} : \text{Red}/: \text{Yellow}^*$ ?o
Contributions / The $\mu$-algebra / An example

:N :Red/:Yellow* ?o
Contributions / The $\mu$-algebra / An example

$N$ :Red/:Yellow$^*$ $?o$
Contributions / The $\mu$-algebra / An example

$N : \text{Red} / : \text{Yellow}^*$ ?o
Contributions / The $\mu$-algebra / An example

$N : \text{Red} / : \text{Yellow}^* \ ? o$
Contributions / The $\mu$-algebra / An example

$N$ :Red/:Yellow$^*$ ?o
Contributions / The $\mu$-algebra / An example

$N$ : Red / : Yellow* ?o
Contributions / The $\mu$-algebra / An example

N : Red / : Yellow

The diagram shows a network of nodes connected by lines, with the node labeled 'N' highlighted in blue.
Contributions / The $\mu$-algebra / An example

$\mathbb{N} : \text{Red} / : \text{Yellow}^*$ *o
Contributions / The \( \mu \)-algebra / An example

\[ \mathbb{N} : \text{Red} / : \text{Yellow}^* \]
Contributions / The $\mu$-algebra / An example

$\mathbb{N} : \text{Red} / : \text{Yellow}^* \ ?_o$
Contributions / The $\mu$-algebra / An example

:N :Red/:Yellow* ?o
Contributions / The $\mu$-algebra / An example

$:N :\text{Red}://\text{Yellow}^* \: ?o$
>>> Contributions / The $\mu$-algebra / An example

:\( N \) :Red/:Yellow* ?o
Contributions / The $\mu$-algebra / An example

: $\mathbb{N}$ : Red / : Yellow * ? o
Rewrite rules for fixpoints
Rewrite rules for fixpoints

* pushing filters?

\[ \sigma_{\text{filter}} (\mu(X = \varphi)) = \mu(X = \sigma_{\text{filter}}(\varphi)) \]
Rewrite rules for fixpoints

* pushing filters?

\[ \sigma_{\text{filter}} (\mu(X = \varphi)) = \mu(X = \sigma_{\text{filter}} (\varphi)) \]

* return fixpoints?
Rewrite rules for fixpoints

* pushing filters?

\[ \sigma_{\text{filter}}(\mu(X = \varphi)) = \mu(X = \sigma_{\text{filter}}(\varphi)) \]

* return fixpoints?

* pushing joins?

\[ \psi \bowtie \mu(X = \varphi) = \mu(X = \psi \bowtie \varphi) \]
Rewrite rules for fixpoints

* pushing filters?

\[ \sigma_{\text{filter}}(\mu(X = \varphi)) = \mu(X = \sigma_{\text{filter}}(\varphi)) \]

* return fixpoints?

\[ \psi \bowtie \mu(X = \varphi) = \mu(X = \psi \bowtie \varphi) \]

* pushing joins?

\[ \pi_p(\mu(X = \varphi)) = \mu(X = \pi_p(\varphi)) \]

* pushing projections?
Rewrite rules for fixpoints

* pushing filters?

\[ \sigma_{\text{filter}}(\mu(X = \varphi)) = \mu(X = \sigma_{\text{filter}}(\varphi)) \]

* return fixpoints?

* pushing joins?

\[ \psi \Join \mu(X = \varphi) = \mu(X = \psi \Join \varphi) \]

* pushing projections?

\[ \pi_p(\mu(X = \varphi)) = \mu(X = \pi_p(\varphi)) \]

* combine fixpoints?

\[ \mu(X = \psi \cup \kappa) \Join \mu(X = \varphi \cup \xi) = \mu(X = \psi \Join \varphi \cup \xi \cup \kappa) \]
Contributions / The $\mu$-algebra / An example

$N : \text{Red} / : \text{Yellow}^\ast \ ?o$
Contributions / The $\mu$-algebra / An example

\[ :N :\text{Red}/:\text{Yellow}^* \circ \]

\[ \pi_s (\sigma_s = :N ( :\text{Red}/\mu(X = \beta_s^o (\text{AllNodes}) \cup X/:\text{Yellow}))) \]
\[
\pi_s(\sigma_s = \pi_s = :N \left( :\text{Red} / :\mu \left( X = \beta^o_s (\text{AllNodes}) \cup X / :\text{Yellow} \right) \right))
\]

\[
\pi_s(\sigma_s = :N \left( \mu \left( X = :\text{Red} / \beta^o_s (\text{AllNodes}) \cup X / :\text{Yellow} \right) \right))
\]
\[ \pi_{s} \left( \sigma_{s=\bar{N}} \left( \mu(X = :Red/\beta^o_s(AllNodes) \cup X/ :Yellow) \right) \right) \]

\[ \pi_{s} \left( \sigma_{s=\bar{N}} \left( \mu(X = :Red/\beta^o_s(AllNodes) \cup X/ :Yellow) \right) \right) \]

\[ \pi_{s} \left( \sigma_{s=\bar{N}} \left( \mu(X = :Red \cup X/ :Yellow) \right) \right) \]
Contributions / The $\mu$-algebra / An example

\[ \pi_s (\sigma \cdot N (\text{:Red}/\mu(X = \beta_s^o(\text{AllNodes}) \cup X/\text{:Yellow}))) \]

\[ \pi_s (\sigma \cdot N (\mu(X = \text{:Red}/\beta_s^o(\text{AllNodes}) \cup X/\text{:Yellow}))) \]

\[ \pi_s (\sigma \cdot N (\mu(X = \text{Red} \cup X/\text{:Yellow}))) \]

\[ \pi_s (\mu(X = \sigma \cdot N (\text{:Red} \cup X/\text{:Yellow}))) \]
Contributions / The $\mu$-algebra / An example

$$\pi_s(\sigma_s=-N(\text{Red}/\mu(X = \beta^o_s(\text{AllNodes}) \cup X/\text{Yellow})))$$

$$\pi_s(\sigma_s=-N(\mu(X = \text{Red}/\beta^o_s(\text{AllNodes}) \cup X/\text{Yellow})))$$

$$\pi_s(\sigma_s=-N(\mu(X = \text{Red} \cup X/\text{Yellow})))$$

$$\pi_s(\mu(X = \sigma_s=-N(\text{Red} \cup X/\text{Yellow})))$$

$$\mu(X = \pi_s(\sigma_s=-N(\text{Red}) \cup X/\text{Yellow}))$$
Methods of evaluating Property Paths:

* Ad-hoc

  Automata, Waveguide\[YGG15\]

* Fixpoints

  Datalog, Recursive SQL
Figure: Time for $N$ :Red/:Yellow* ?o on $n$ nodes
1. Generate equivalent terms
2. Select an estimated most efficient
3. Execute it
Cost model

Estimate the running time to evaluate a term.

\[ \text{Cost}(A \times B) = \text{Cost}(A) + \text{Cost}(B) + O(\text{size}(A \times B)) \]
Cost model

Estimate the running time to evaluate a term.

\[ \text{Cost}(A \otimes B) = \text{Cost}(A) + \text{Cost}(B) + O(\text{size}(A \otimes B)) \]

Cardinality estimation

Estimate the number of solutions to a term.
CardEst

A worst-case cardinality estimation with a new tool: summaries.
SPARQLGX

SPARQLGX is a distributed SPARQL query evaluator based on Apache Spark.

CardEst

A worst-case cardinality estimation with a new tool: summaries.
The hash join algorithm

\[ Map(Cogroup(A, B), \text{iter}) \]

\[ O((|A| + |B|) \times \text{shuffle} + |A \Join B|) \]

The broadcast join algorithm

\[ MapValues(A, f_B) \]

\[ O(|A| + |B| \times \#\text{workers} + |A \Join B|) \]
Figure: LUBM [GPH05] 10k (1,4 billions triples)

Figure: WatDiv [AHÖD14] 1k (140 millions triples)
1. Generate equivalent terms
2. Select an estimated most efficient
3. Execute it
Streams are one-way communication channels
Streams are one-way communication channels
Streams are one-way communication channels
Streams are one-way communication channels
Streams are one-way communication channels
Execution of $\mu$-algebra terms with streams

Execution of $\mu$-algebra terms with streams

\[
\begin{array}{c}
\text{start} \rightarrow \\
X_1 \rightarrow \\
\vdots \rightarrow \\
\vdots \rightarrow \\
X_n \rightarrow \\
\end{array}
\]
Figure: \(a \ (P1+)//(P5+) \ ?b\).
Figure: $a \ (P1+)/P2 \ ?b \ . \ ?b \ P3+ \ ?c$. 
Figure: $N0 \ P1/(P2+) \ ?a$
Figure: $a \ (P4+)/ (P5+)/ (P3+) \ b$
Conclusion / Contributions

Main contributions:

- **μ-Algebra** A new algebra with new efficient rewriting rules for fixpoints
- **muSPARQL** A prototype evaluator based on streams
- **CardEst** A new cardinality estimation technique
- **SPARQLGX** A distributed SPARQL query evaluator

Collaborations:

* Leaves enumeration technique.
* Distributed SPARQL query evaluators benchmarks.
* ProvSQL: provenance for SQL.
Conclusion / Contributions

Main contributions:

- **μ-Algebra** A new algebra with new efficient rewriting rules for fixpoints  
  BDA’17, BDA’18

- **muSPARQL** A prototype evaluator based on streams

- **CardEst** A new cardinality estimation technique

- **SPARQLGX** A distributed SPARQL query evaluator  
  ISWC’16

Collaborations:

* Leaves enumeration technique.  
  ICALP’17

* Distributed SPARQL query evaluators benchmarks.  
  BDA’17

* ProvSQL: provenance for SQL.  
  VLDB’18
Short-term perspectives

* Complete the distributed implementation of $\mu$-algebra
* Use $\mu$-algebra for SPARQL with ontology-based data access

Long-term perspectives

* Improve cardinality estimation scheme
* Port the optimization of the $\mu$-algebra to SQL and Datalog
>>> Conclusion / Questions?

Questions?
Güneş Aluç, Olaf Hartig, M Tamer Özsu, and Khuzaima Daudjee.
Diversified stress testing of rdf data management systems.


Edgar F Codd.

Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin.

Steve Harris, Andy Seaborne, and Eric Prud’hommeaux.
SPARQL query language for RDF.  
www.w3.org/TR/rdf-sparql-query/.  

David Wood Richard Cyganiak and Lanthaler Markus.  
RDF 1.1 concepts and abstract syntax, February 2014.  

Nikolay Yakovets, Parke Godfrey, and Jarek Gryz.  
Towards Query Optimization for SPARQL Property Paths.  