



**HAL**  
open science

## ViP2P: XML Warehousing in P2P Networks

Ioana Manolescu, Spyros Zoupanos

► **To cite this version:**

Ioana Manolescu, Spyros Zoupanos. ViP2P: XML Warehousing in P2P Networks. école thématique BDA, May 2010, Les Houches, France. inria-00543940

**HAL Id: inria-00543940**

**<https://inria.hal.science/inria-00543940>**

Submitted on 6 Dec 2010

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# ViP2P: XML Warehousing in P2P Networks

Ioana Manolescu-Goujot

Equipe Leo

INRIA Saclay-Île-de-France & LRI, Université de Paris Sud-11



Supported by: *ANR-08-DEFIS-004 CODEX, INRIA ADT ViP2P & ViRDF*

## Leo and friends in front of our lab



# Part I

## Introduction

## Motivation

Distributed data management: old goal (1970)

## Motivation

Distributed data management: old goal (1970)

- distributed versions of industrial-strength DBMSs
- map/reduce style systems for massively parallel computations

## Motivation

Distributed data management: old goal (1970)

- distributed versions of industrial-strength DBMSs
- map/reduce style systems for massively parallel computations

Still missing: the **flexible federation**

- high independence of the sites: when to be in, what to store
- **data distribution transparency**
- ... with the usual performance requirements

## Distributed warehouses of Web content

### Web content

structured documents, schemas, annotations, concepts, mappings,  
Web services, inter-document links

### Web content warehouse

Distributed database of selected content, whose users may:

- publish resources
- connect (annotate, map, link...) existing resources
- update resources
- **enhance** resources by combining them

In the style of the RNTL WebContent project (2005-2009)

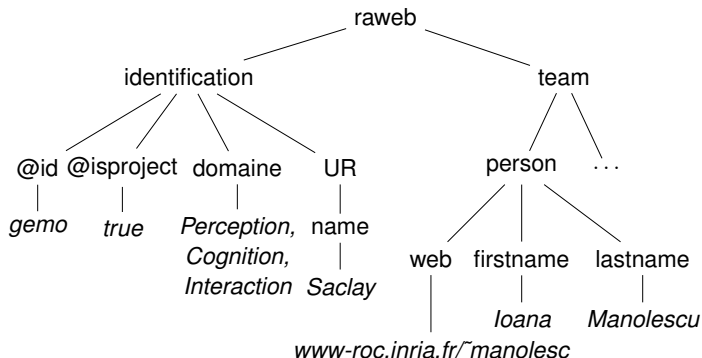


## INRIA annual activity report in XML

<https://irabot.inrialpes.fr/RA2009/gemo.xml>

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE raweb PUBLIC "raweb2.dtd">
<raweb year="2009">
  <identification id="gemo" isproject="true">
    <shortname>gemo</shortname>
    <domaine>Perception, Cognition, Interaction</domaine>
    <UR name="Saclay"/>
  </identification>
  <team>
    <person>
      <web>http://www-roc.inria.fr/~manolesc</web>
      <firstname>Ioana</firstname>
      <lastname>Manolescu</lastname>
    </person>...
  </team>...
</raweb>
```

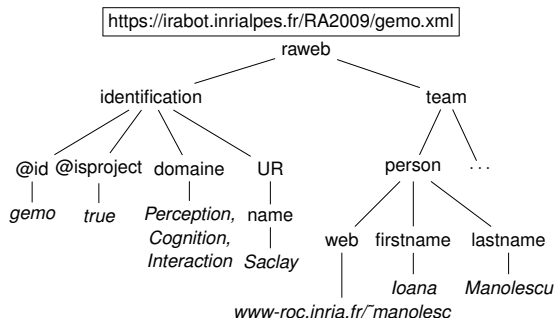
# INRIA annual activity report in XML



- (Very) popular data exchange format
- Many W3C standards + efficient tools

# XML and the Semantic Web

## XML



## RDF

`http://gemo.saclay.inria.fr`  
*wroteReport*

`https://irabot.inrialpes.fr/RA2009/gemo.xml`

---

`http://gemo.saclay.inria.fr`  
*coordinates*

`http://codex.saclay.inria.fr`

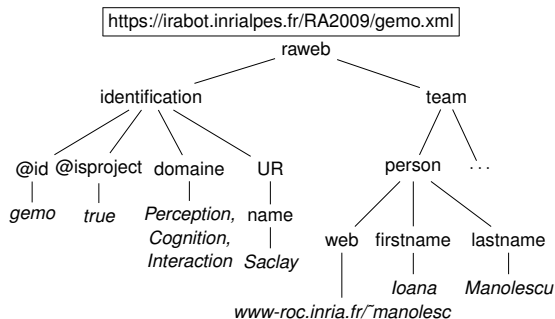
---

`http://codex.saclay.inria.fr`  
*hasType*

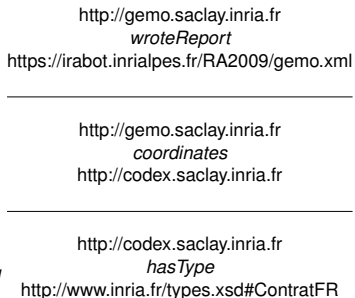
`http://www.inria.fr/types.xsd#ContratFR`

# XML and the Semantic Web

## XML



## RDF

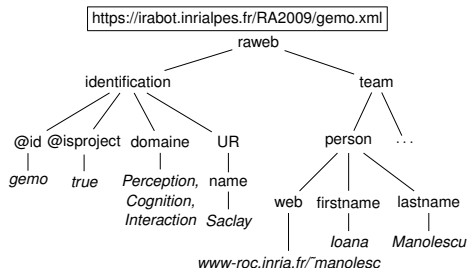


## Possible queries

- European projects involving at least two INRIA teams?
- Most frequent keywords for each INRIA theme?
- PhD students co-supervised with a foreign University?

# Distributed warehouses of Web content

## Distributed XML



<http://gemo.saclay.inria.fr>

<http://codex.saclay.inria.fr>



## Distributed RDF

Serv. Inf.  
Sci. Tech.  
INRIA

<http://pierre.senellart.com>  
*collaboratesWith*  
<https://gemo.saclay.inria.fr>

Indiana  
Univ.

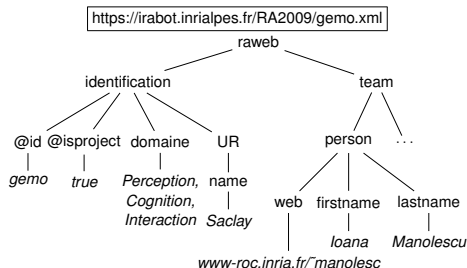
<http://pierre.senellart.com>  
*coordinates*  
[dbweb.enst.fr/events/sigmod10contest/](http://dbweb.enst.fr/events/sigmod10contest/)

Splitted  
Desktop

<http://gemo.saclay.inria.fr>  
*possibleCollaborator*  
<http://www.splitted-desktop.com/fr/>

# Distributed warehouses of Web content

## Distributed XML



<http://gemo.saclay.inria.fr>



<http://codex.saclay.inria.fr>



## Distributed RDF

Serv. Inf.  
Sci. Tech.  
INRIA

<http://pierre.senellart.com>  
*collaboratesWith*  
<https://gemo.saclay.inria.fr>

Indiana  
Univ.

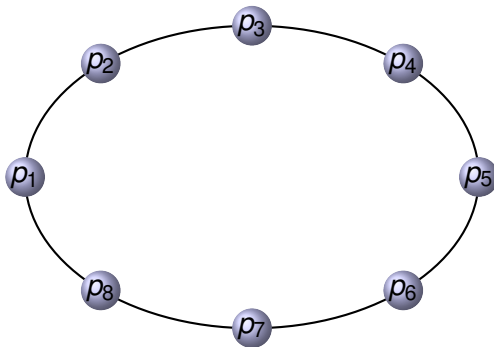
<http://pierre.senellart.com>  
*coordinates*  
<http://dbweb.enst.fr/events/sigmod10contest/>

Splitted  
Desktop

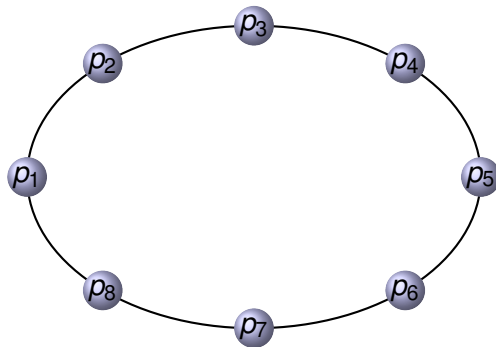
<http://gemo.saclay.inria.fr>  
*possibleCollaborator*  
<http://www.splitted-desktop.com/fr/>

- French researchers involved in ACM conference organization?
- Possible collaborators for a data management project in the clouds?

## Distributed hash tables

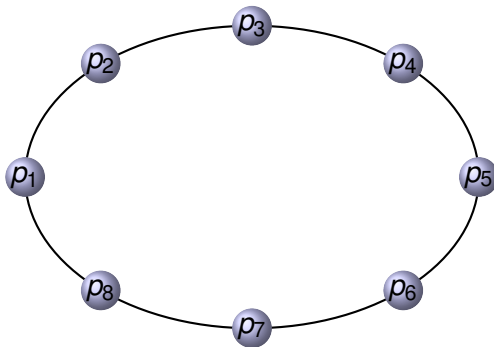


## Distributed hash tables

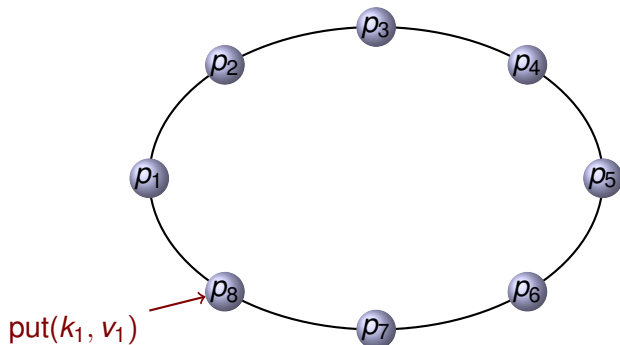




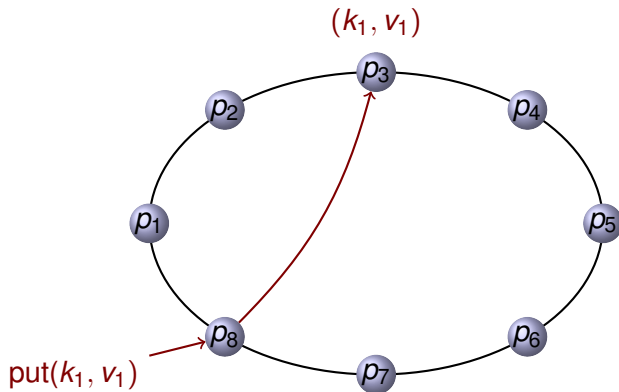
## Distributed hash tables



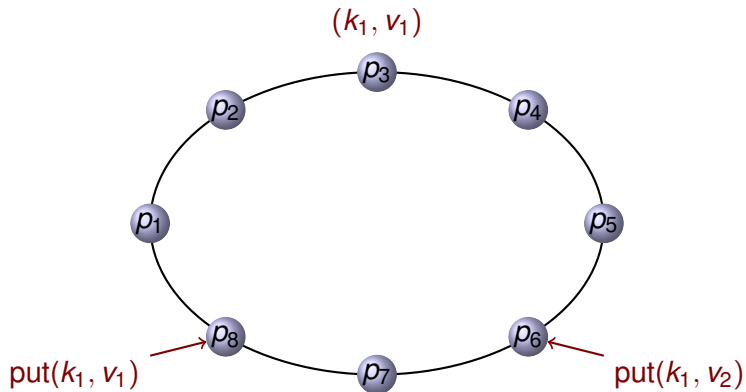
## Distributed hash tables



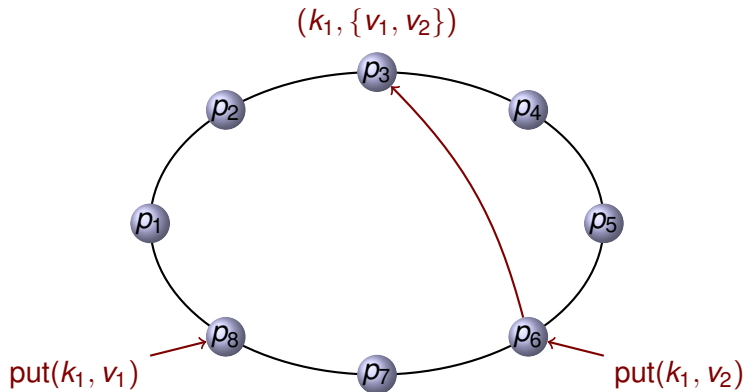
# Distributed hash tables



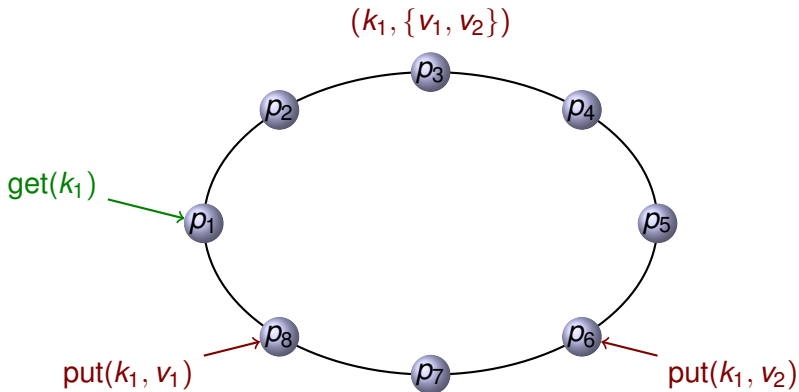
## Distributed hash tables



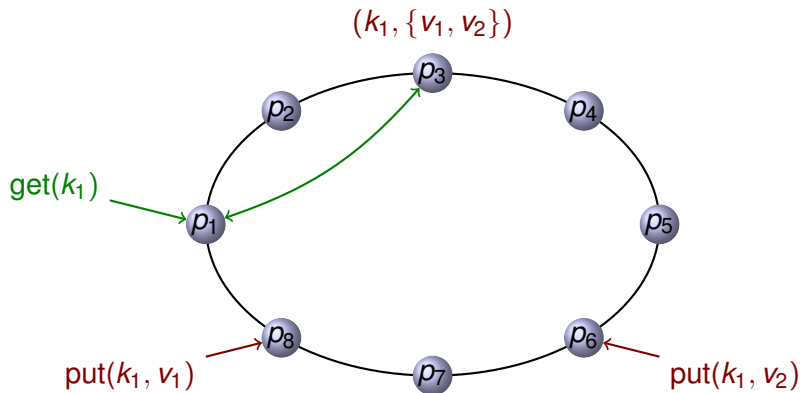
## Distributed hash tables



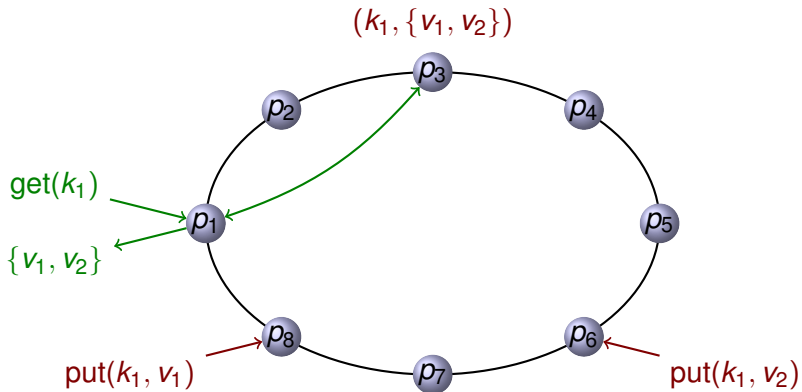
## Distributed hash tables



# Distributed hash tables



# Distributed hash tables





## What distributed hash tables provide

### Dynamic peer networks

- each peer is assigned an **id**  $\Rightarrow$  **address range**
- bound of  $\log_2(N)$  hops to route a message to peer
- **network re-adjustment** when peers join or leave: peers' address spaces stretch and contract

## What distributed hash tables provide

### Dynamic peer networks

- each peer is assigned an **id**  $\Rightarrow$  **address range**
- bound of  $\log_2(N)$  hops to route a message to peer
- **network re-adjustment** when peers join or leave: peers' address spaces stretch and contract

(Key, value) stores = basis for content sharing

- **index** the resources by **keys**
- **look up** resources by **keys**

## What distributed hash tables provide

### Dynamic peer networks

- each peer is assigned an **id**  $\Rightarrow$  **address range**
- bound of  $\log_2(N)$  hops to route a message to peer
- **network re-adjustment** when peers join or leave: peers' address spaces stretch and contract

(Key, value) stores = basis for content sharing

- **index** the resources by **keys**
- **look up** resources by **keys**

### Main DHT functionality

**Small** shared memory  $\Rightarrow$  resource catalog

## From DHTs to distributed data management

### Functionalities to add

- data indexing
- **storage** for application data and even DHT index data
- local query processing
- **distributed query processing**: operators, including **data transfers**, optimization . . .

## From DHTs to distributed data management

### Functionalities to add

- data indexing
- **storage** for application data and even DHT index data
- local query processing
- **distributed query processing**: operators, including **data transfers**, optimization . . .

### Reliability

- provided** a peer will eventually answer at each address  
some (key, value) replication to handle peer failures  
(broadcast to  $k$  replicas)
- to do** resilience of all added functionalities to peer joining/leaving/failing

## Querying distributed data on a DHT

Two types of operations:

- 1 Look up useful data sources in the DHT
  - Documents matching the query
  - Documents which *might* match the query

## Querying distributed data on a DHT

Two types of operations:

- 1 Look up useful data sources in the DHT
  - Documents matching the query
  - Documents which *might* match the query

DHT indexing is important here

## Querying distributed data on a DHT

Two types of operations:

- 1 Look up useful data sources in the DHT
  - Documents matching the query
  - Documents which *might* match the query

DHT indexing is important here

Index building and maintenance



## Querying distributed data on a DHT

Two types of operations:

- 1 Look up useful data sources in the DHT
  - Documents matching the query
  - Documents which *might* match the query

DHT indexing is important here

Index building and maintenance

- 2 Remaining query evaluation steps
  - **evaluate** sub-queries to extract partial results
  - **combine** partial results
  - **transfer** results to query peer

## Querying distributed data on a DHT

Two types of operations:

- 1 Look up useful data sources in the DHT
  - Documents matching the query
  - Documents which *might* match the query

DHT indexing is important here

Index building and maintenance

- 2 Remaining query evaluation steps
  - **evaluate** sub-queries to extract partial results
  - **combine** partial results
  - **transfer** results to query peer

Distributed query processing “as usual”

## Trade-offs in DHT indexing and query processing

### Level of detail of resource indexing

- lookup precision ↗ ⇒ execution time ↘
- data publication time ↗, possibly execution time ↗

### Data re-placement or clustering

- fewer peers contacted for a query (message no. ↘, execution time ?)
- data transfers in the absence of queries (message no. ↗, total message size ↗)

## Part II

# ViP2P: materialized views on DHTs

5 Credits

6 ViP2P overview

7 ViP2P views

8 ViP2P rewritings

9 The distributed platform

## Joint work with

### Students

- Konstantinos Karanasos, Asterios Katsifodimos, Spyros Zoupanos (PhD 2009)
- Martin Goodfellow (U. Strathclyde), Silviu Julean (UVT, Romania), Varunesh Mishra (IIT Kanpur), Alexandra Roatis (UVT, Romania)

### Engineers

Jesús Camacho-Rodriguez, Julien Leblay, Alin Tilea

### Friends from abroad

Angela Bonifati (CNR Italy), Vasilis Vassalos (AUER, Greece)

## Web data sharing in ViP2P

Web data:

- XML documents
- RDF annotations

## Web data sharing in ViP2P

Web data:

- XML documents
- RDF annotations

Choices:



## Web data sharing in ViP2P

Web data:

- XML documents
- RDF annotations

Choices:

- peers retain control over the data they store/publish
  - no global schema
  - **documents** published independently
  - **annotations** (triples) can freely connect content
- content is re-distributed: peers may accumulate results of long-running queries
- peers share their accumulated results with others

## Behind the scene: distributed materialized views

## Behind the scene: distributed materialized views

Long-running query = subscription = view

Declarative specification of results that the peer is interested in

- tree pattern  $\Rightarrow$  matches an individual document
- tree patterns with value joins  $\Rightarrow$  matches across distinct documents
- corresponding XQuery dialect

## Behind the scene: distributed materialized views

Long-running query = subscription = view

Declarative specification of results that the peer is interested in

- tree pattern  $\Rightarrow$  matches an individual document
- tree patterns with value joins  $\Rightarrow$  matches across distinct documents
- corresponding XQuery dialect
- flexibility

## Behind the scene: distributed materialized views

Long-running query = subscription = view

Declarative specification of results that the peer is interested in

- tree pattern  $\Rightarrow$  matches an individual document
- tree patterns with value joins  $\Rightarrow$  matches across distinct documents
- corresponding XQuery dialect
- flexibility

Data publication = incremental update of the view

Easier for tree pattern views

## Behind the scene: distributed materialized views

Long-running query = subscription = view

Declarative specification of results that the peer is interested in

- tree pattern  $\Rightarrow$  matches an individual document
- tree patterns with value joins  $\Rightarrow$  matches across distinct documents
- corresponding XQuery dialect
- flexibility

Data publication = incremental update of the view

Easier for tree pattern views

Query planning = rewriting queries using views

Find relevant views  $\Rightarrow$  rewrite the query  $\Rightarrow$  optimize the rewriting

## Behind the scene: distributed materialized views

Long-running query = subscription = view

Declarative specification of results that the peer is interested in

- tree pattern  $\Rightarrow$  matches an individual document
- tree patterns with value joins  $\Rightarrow$  matches across distinct documents
- corresponding XQuery dialect
- flexibility

Data publication = incremental update of the view

Easier for tree pattern views

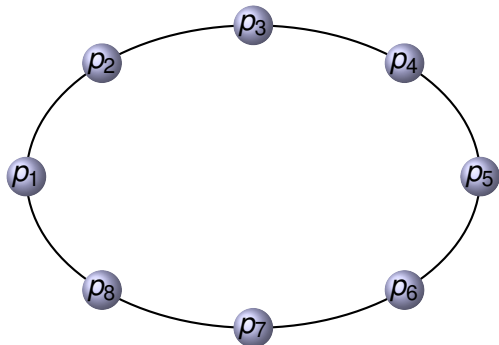
Query planning = rewriting queries using views

Find relevant views  $\Rightarrow$  rewrite the query  $\Rightarrow$  optimize the rewriting

Query execution = distributed query processing

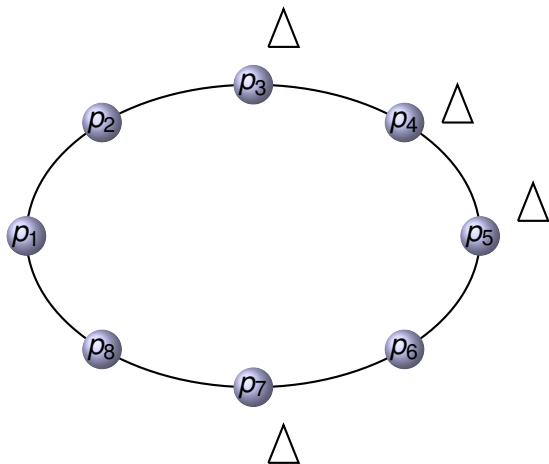
With XML- and RDF-specific bells & whistles

## Publishing and querying in ViP2P





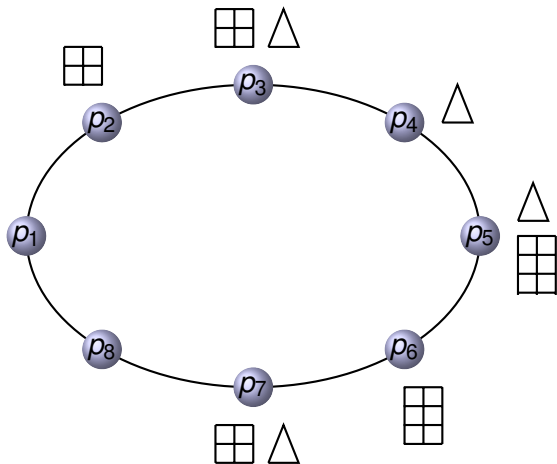
## Publishing and querying in ViP2P



The peers may store:

- documents, annotations

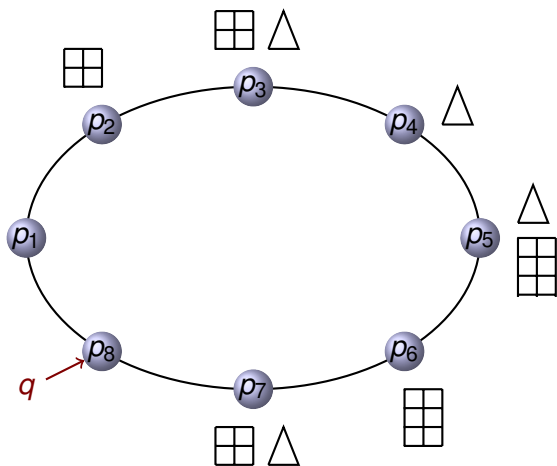
## Publishing and querying in ViP2P



The peers may store:

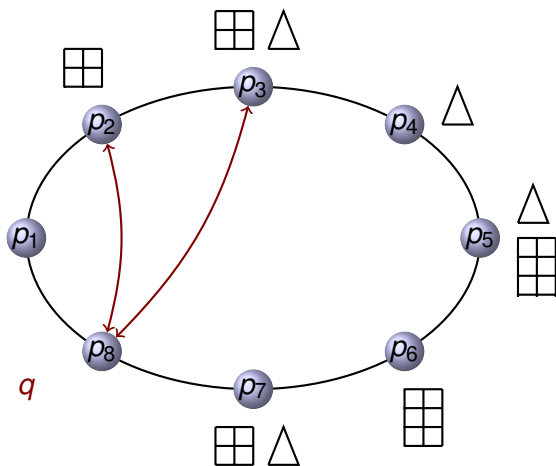
- documents, annotations
- views

## Publishing and querying in ViP2P



When  $q$  arrives:

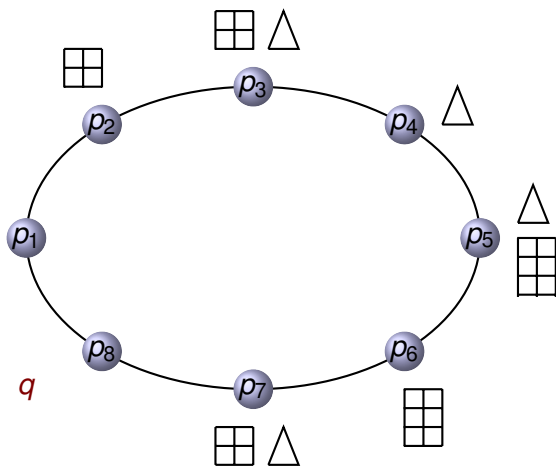
## Publishing and querying in ViP2P



When  $q$  arrives:

- view definition lookup

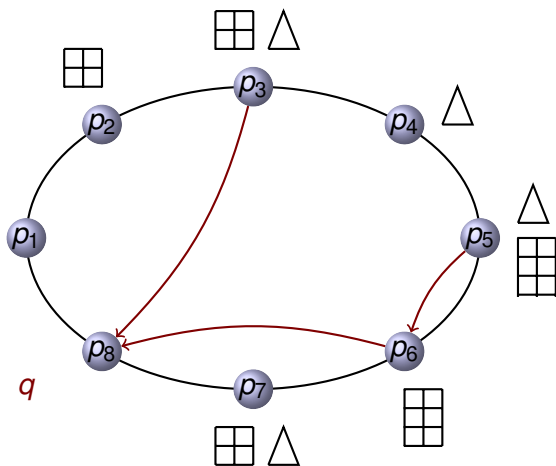
## Publishing and querying in ViP2P



When  $q$  arrives:

- view definition lookup
- rewriting

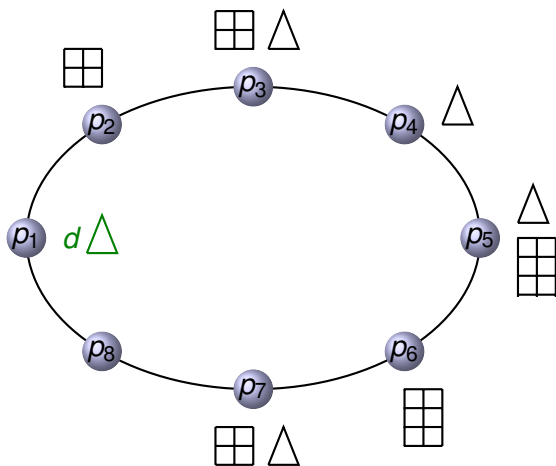
## Publishing and querying in ViP2P



When  $q$  arrives:

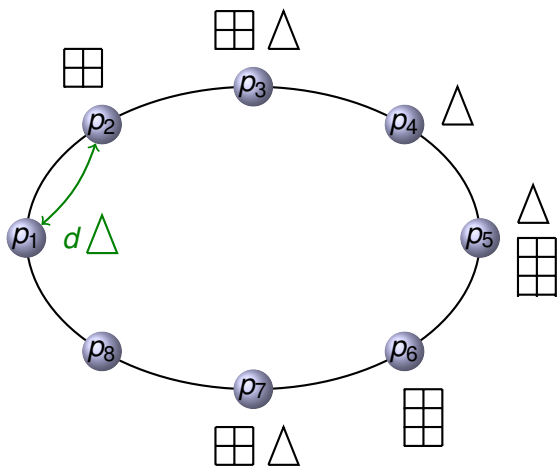
- view definition lookup
- rewriting
- execution of physical plan

## Publishing and querying in ViP2P



When  $d$  arrives:

## Publishing and querying in ViP2P

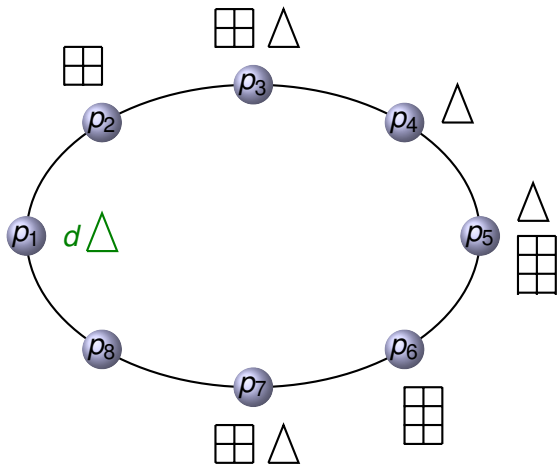


When  $d$  arrives:

- search view definitions for which  $v_i(d) \neq \emptyset$



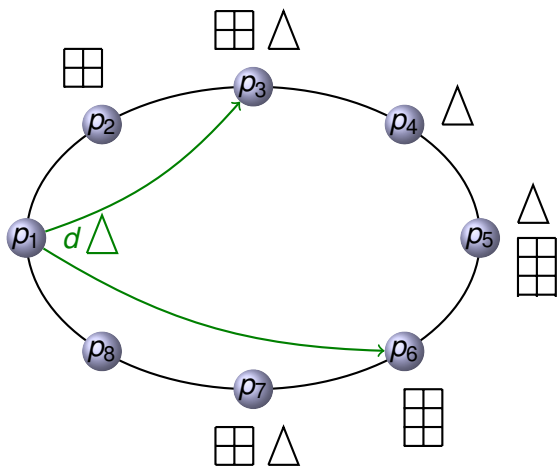
## Publishing and querying in ViP2P



When  $d$  arrives:

- search view definitions for which  $v_i(d) \neq \emptyset$
- compute  $v_i(d)$

# Publishing and querying in ViP2P

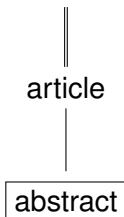


When  $d$  arrives:

- search view definitions for which  $v_i(d) \neq \emptyset$
- compute  $v_i(d)$
- send results

## Sample views

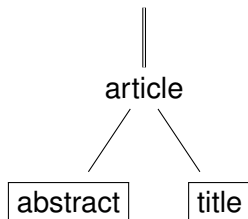
All article abstracts



URI	abstract
<a href="http://peer1.com/doc1.xml">http://peer1.com/doc1.xml</a>	Databases...
<a href="http://peer2.com/f2.xml">http://peer2.com/f2.xml</a>	XML processing...
...	...

## Sample views

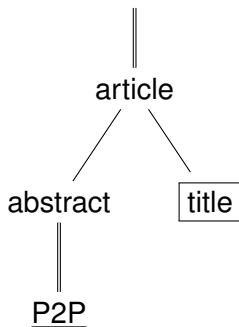
Titles and abstracts of all articles



URI	abstract	title
...	...	...

## Sample views

Titles of all articles having “P2P” in the abstract



URI	title
...	...

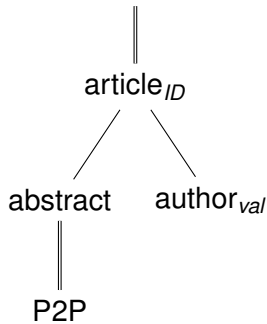
## Sample views

URIs of all documents containing  
articles having “P2P” in the abstract



## Sample views

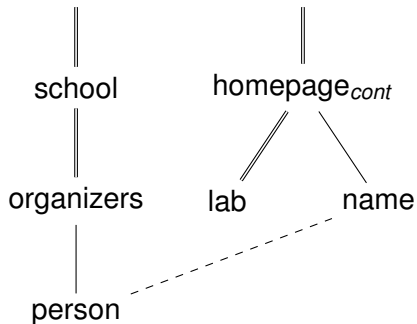
IDs and authors of all articles having  
“P2P” in the abstract



URI	article.ID	author.val
...	...	...

## Sample views

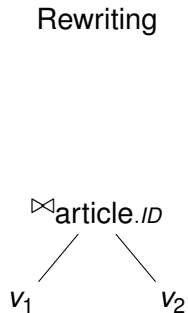
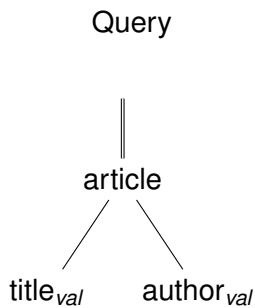
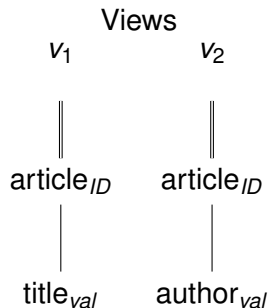
Homepages of all WebDam summer school organizers



URI1	URI2	homepage.cont
...	...	...

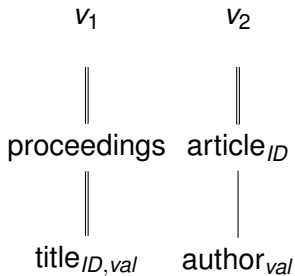


## Sample rewritings

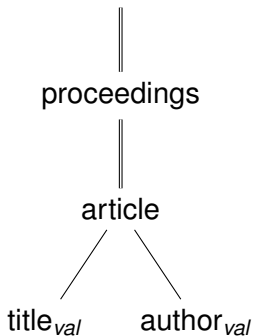


## Sample rewritings

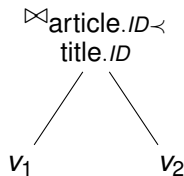
Views



Query

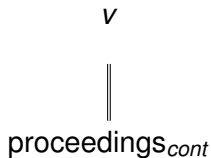


Rewriting

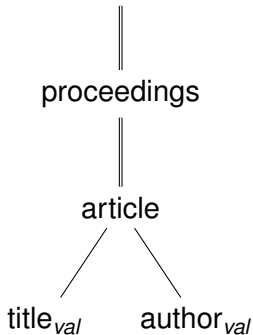


## Sample rewritings

Views



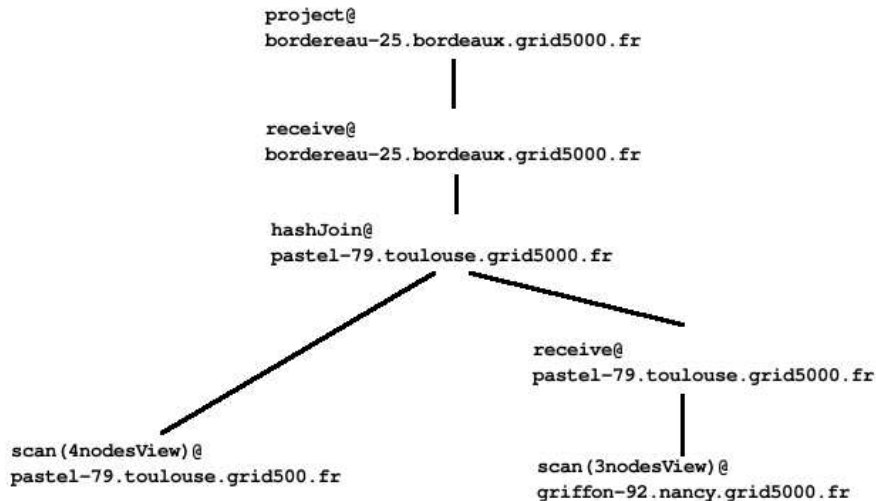
Query



Rewriting



## Sample physical rewriting plan



## ViP2P platform

- 70.000 Java lines (330 classes) and decreasing :)
- Uses Berkeley DB (version 3.3.75) to store view data
- Uses FreePastry (version 2.1) as our DHT network
- Experiments carried on *Grid5000* using **250 machines**
- Deployed on **1000 ViP2P peers**

## ViP2P platform

- 70.000 Java lines (330 classes) and decreasing :)
- Uses Berkeley DB (version 3.3.75) to store view data
- Uses FreePastry (version 2.1) as our DHT network
- Experiments carried on *Grid5000* using **250 machines**
- Deployed on **1000 ViP2P peers**

Algorithms inside:

- SAX-based custom processor for computing  $v(d)$  + identifiers
- Iterator-based execution engine, (nested) tuples
- Send/receive for distributed execution + URI compression
- Many query optimization strategies

## Indexing views for view materialization

### View indexing

For each view  $v$  and node label  $l$  in  $v$ , put  $(l, v)$

For each view  $v$  and constant  $k$  in  $v$ , put  $(k, v)$

## Indexing views for view materialization

### View indexing

For each view  $v$  and node label  $l$  in  $v$ , put  $(l, v)$

For each view  $v$  and constant  $k$  in  $v$ , put  $(k, v)$

### View lookup

For each document  $d$  and node label  $l$  in  $d$ , get( $l$ )



## Indexing views for view materialization

### View indexing

For each view  $v$  and node label  $l$  in  $v$ , put  $(l, v)$

For each view  $v$  and constant  $k$  in  $v$ , put  $(k, v)$

### View lookup

For each document  $d$  and node label  $l$  in  $d$ , get( $l$ )

**Filter** views thus obtained:

**if**  $v$  uses a tag/keyword not present in  $d$   
**then** discard  $v$

## Indexing views for view materialization

### View indexing

For each view  $v$  and node label  $l$  in  $v$ , put  $(l, v)$

For each view  $v$  and constant  $k$  in  $v$ , put  $(k, v)$

### View lookup

For each document  $d$  and node label  $l$  in  $d$ , get( $l$ )

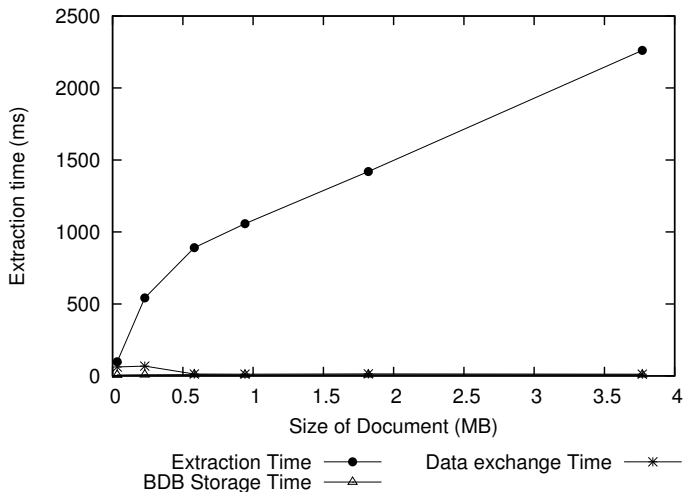
**Filter** views thus obtained:

**if**  $v$  uses a tag/keyword not present in  $d$   
**then** discard  $v$

Evaluate the filtered view set on  $d$

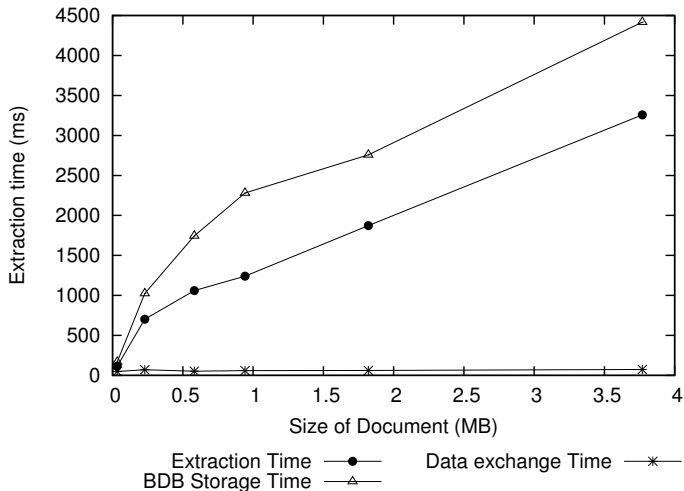
## View materialization

View + documents = few results



## View materialization

View + documents = many results



## Indexing views for query rewriting

### View indexing

For each view  $v$  and node label  $l$  in  $v$ , put  $(l, v)$

For each view  $v$  and constant  $k$  in  $v$ , put  $(k, v)$

## Indexing views for query rewriting

### View indexing

For each view  $v$  and node label  $l$  in  $v$ , put  $(l, v)$

For each view  $v$  and constant  $k$  in  $v$ , put  $(k, v)$

### View lookup

For each query  $q$  and label  $l$  in  $q$ , get( $l$ )

For each query  $q$  and constant  $k$  in  $q$ , get( $k$ )

## Indexing views for query rewriting

### View indexing

For each view  $v$  and node label  $l$  in  $v$ , put  $(l, v)$

For each view  $v$  and constant  $k$  in  $v$ , put  $(k, v)$

### View lookup

For each query  $q$  and label  $l$  in  $q$ , get( $l$ )

For each query  $q$  and constant  $k$  in  $q$ , get( $k$ )

**Filter** the views thus obtained:

**if**  $v$  cannot be embedded in  $q$   
**then** discard  $v$

## Indexing views for query rewriting

### View indexing

For each view  $v$  and node label  $l$  in  $v$ , put  $(l, v)$

For each view  $v$  and constant  $k$  in  $v$ , put  $(k, v)$

### View lookup

For each query  $q$  and label  $l$  in  $q$ , get( $l$ )

For each query  $q$  and constant  $k$  in  $q$ , get( $k$ )

**Filter** the views thus obtained:

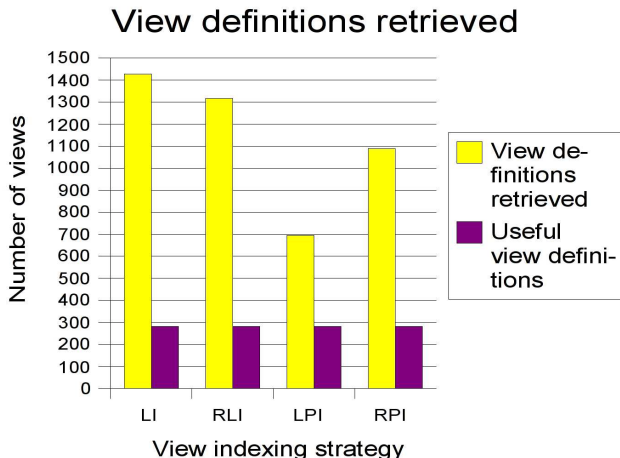
**if**  $v$  cannot be embedded in  $q$   
**then** discard  $v$

Rewrite  $q$  with the filtered view set



## View look up performance

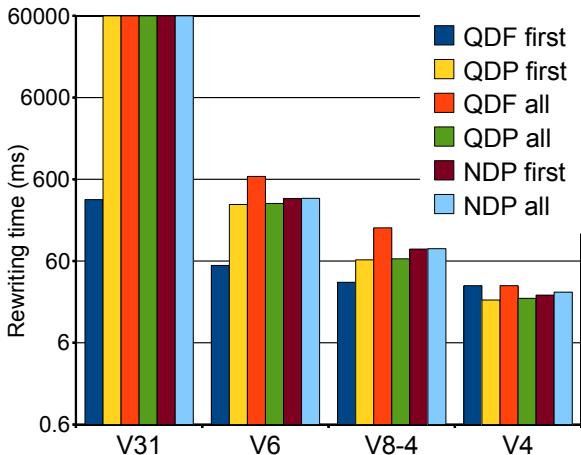
1440 views related to a 32-nodes query  $q$



## View rewriting performance

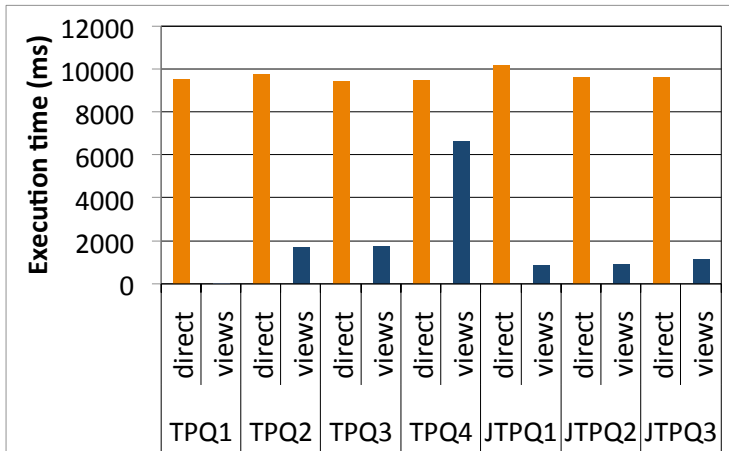
We find **minimal** rewritings

NP-hard problem; heuristics for finding some rewriting fast



## The interest of view-based rewriting

Almost-XMark queries



## Part III

# Summary

We have

A distributed engine for sharing XML data based on materialized views

We have

A distributed engine for sharing XML data based on materialized views

We are working on

- 1 XML+RDF management (K. Karanasos, F. Goasdoué)
- 2 Self-adaptive views (A. Katsifodimos, J. Camacho-Rodriguez, V. Mishra)
- 3 Incremental view maintenance (M. Goodfellow, A. Bonifati)
- 4 Query optimization (A. Roatis)

### We have

A distributed engine for sharing XML data based on materialized views

### We are working on

- 1 XML+RDF management (K. Karanasos, F. Goasdoué)
- 2 Self-adaptive views (A. Katsifodimos, J. Camacho-Rodriguez, V. Mishra)
- 3 Incremental view maintenance (M. Goodfellow, A. Bonifati)
- 4 Query optimization (A. Roatis)

### Blue sky

- 1 Pure RDF management in P2P
- 2 Moving the back-end into the clouds (EnergieNet)

## Closest related works

DHT-based sharing of relations [LHSH04]

DHT-based XML indexing [GWJD03, BC06, SHA05, AMP<sup>+</sup>08]

DHT-based shared XML caches [LP08]

XPath query rewriting [BOB<sup>+</sup>04, XO05, CDO08, TYÖ<sup>+</sup>08]

- XPath: wildcard \*, union
- Rewritings: intersection, navigations, joins

Rewriting with structural constraints [ABMP07]

- Centralized setting
- Dataguide [GW97] constraints

Layered architecture for Web content warehousing [AAC<sup>+</sup>08]

RDF querying and reasoning on DHT [KMK08, LIK06]



# Thank you!

<http://vip2p.saclay.inria.fr>

- [AAC<sup>+</sup>08] Serge Abiteboul, Tristan Allard, Philippe Chatalic, Georges Gardarin, A. Ghitescu, François Goasdoué, Ioana Manolescu, Benjamin Nguyen, M. Ouazara, A. Somani, Nicolas Travers, Gabriel Vasile, and Spyros Zoupanos.  
WebContent: efficient P2P warehousing of web data.  
*PVLDB*, 1(2):1428–1431, 2008.
- [ABMP07] Andrei Arion, Véronique Benzaken, Ioana Manolescu, and Yannis Papakonstantinou.  
Structured materialized views for XML queries.  
In *VLDB*, pages 87–98, 2007.
- [AMP<sup>+</sup>08] Serge Abiteboul, Ioana Manolescu, Neoklis Polyzotis, Nicoleta Preda, and Chong Sun.  
XML processing in DHT networks.  
In *ICDE*, pages 606–615, 2008.
- [BC06] Angela Bonifati and Alfredo Cuzzocrea.

Storing and retrieving XPath fragments in structured P2P networks.

*Data Knowl. Eng.*, 59(2), 2006.

[BOB<sup>+</sup>04] A. Balmin, F. Ozcan, K. Beyer, R. Cochrane, and H. Pirahesh.

A framework for using materialized XPath views in XML query processing.

In *VLDB*, 2004.

[CDO08] Bogdan Cautis, Alin Deutsch, and Nicola Onose.

XPath rewriting using multiple views: Achieving completeness and efficiency.

In *WebDB*, 2008.

[GW97] Roy Goldman and Jennifer Widom.

Dataguides: Enabling query formulation and optimization in semistructured databases.

In *VLDB*, 1997.

- [GWJD03] L. Galanis, Y. Wang, S.R. Jeffery, and D.J. DeWitt.  
Locating data sources in large distributed systems.  
In *VLDB*, 2003.
- [KMK08] Zoi Kaoudi, Iris Miliaraki, and Manolis Koubarakis.  
RDFS reasoning and query answering on top of DHTs.  
In *International Semantic Web Conference*, pages  
499–516, 2008.
- [LHSH04] Boon Thau Loo, Ryan Huebsch, Ion Stoica, and Joseph M.  
Hellerstein.  
The case for a hybrid P2P search infrastructure.  
In *IPTPS*, pages 141–150, 2004.
- [LIK06] Erietta Liarou, Stratos Idreos, and Manolis Koubarakis.  
Evaluating conjunctive triple pattern queries over large  
structured overlay networks.  
In *International Semantic Web Conference*, pages  
399–413, 2006.

- [LP08] Kostas Lillis and Evaggelia Pitoura.  
Cooperative XPath caching.  
In *SIGMOD Conference*, pages 327–338, 2008.
- [SHA05] Gleb Skobeltsyn, Manfred Hauswirth, and Karl Aberer.  
Efficient processing of XPath queries with structured overlay networks.  
In *OTM Conferences (2)*, 2005.
- [TYÖ<sup>+</sup>08] Nan Tang, Jeffrey Xu Yu, M. Tamer Özsu, Byron Choi, and Kam-Fai Wong.  
Multiple materialized view selection for XPath query rewriting.  
In *ICDE*, pages 873–882, 2008.
- [XO05] W. Xu and M. Ozsoyoglu.  
Rewriting XPath queries using materialized views.  
In *VLDB*, 2005.