Distributed RDF Query Processing and Reasoning in Peer-to-Peer Networks

Zoi Kaoudi
Postdoctoral researcher

Work done in the context of my PhD thesis at the National and Kapodistrian University of Athens

Outline

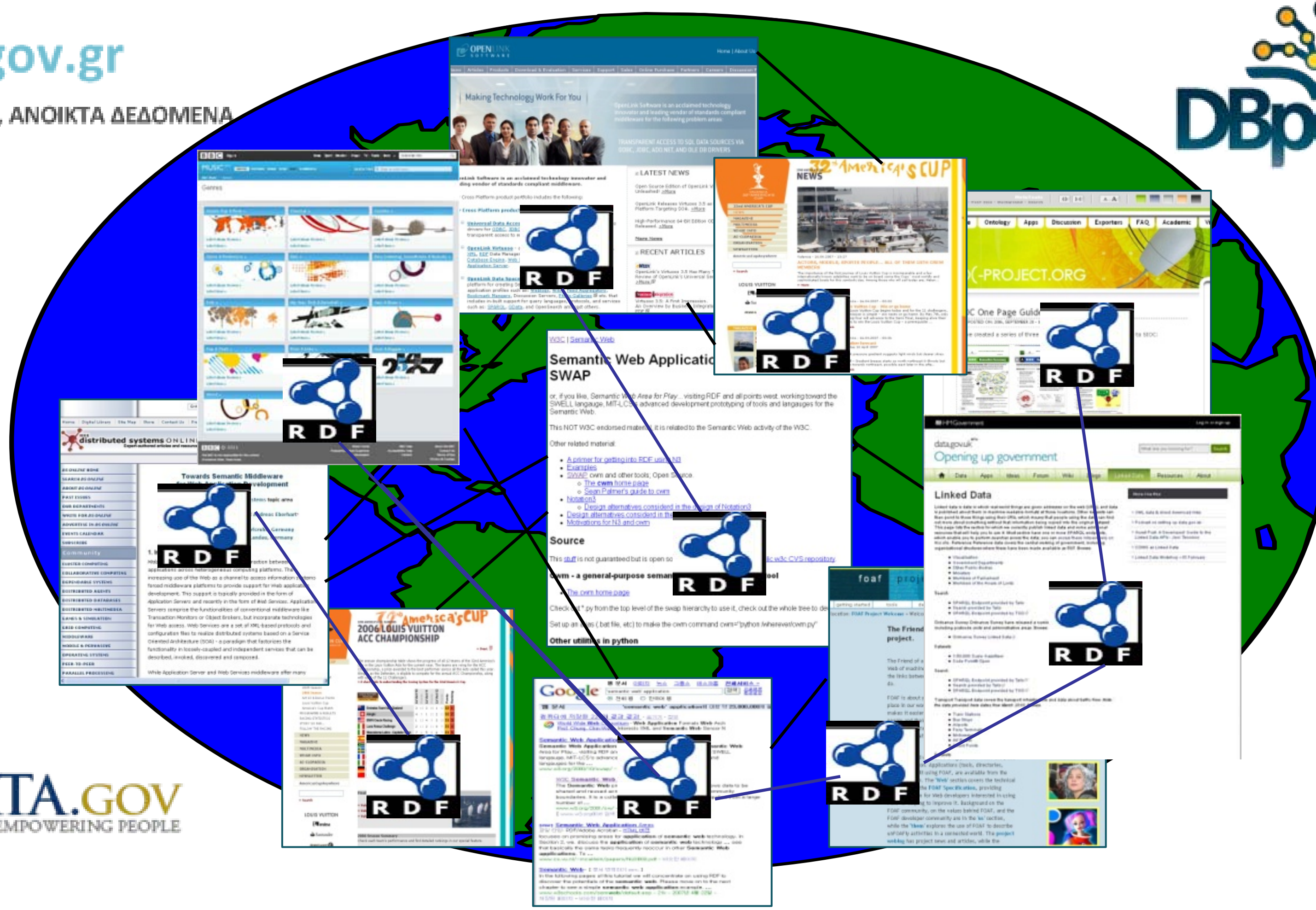
- Introduction
- The system Atlas
- RDFS Reasoning in P2P networks
- SPARQL Query Processing and Optimization
- Conclusions

Web of Documents/Web of Data

geodata.gov.gr

ΔΗΜΟΣΙΑ ΔΕΔΟΜΕΝΑ, ΑΝΟΙΚΤΑ ΔΕΔΟΜΕΝΑ

DBpedia



DATA.GOV
EMPOWERING PEOPLE

data.gov.uk BETA

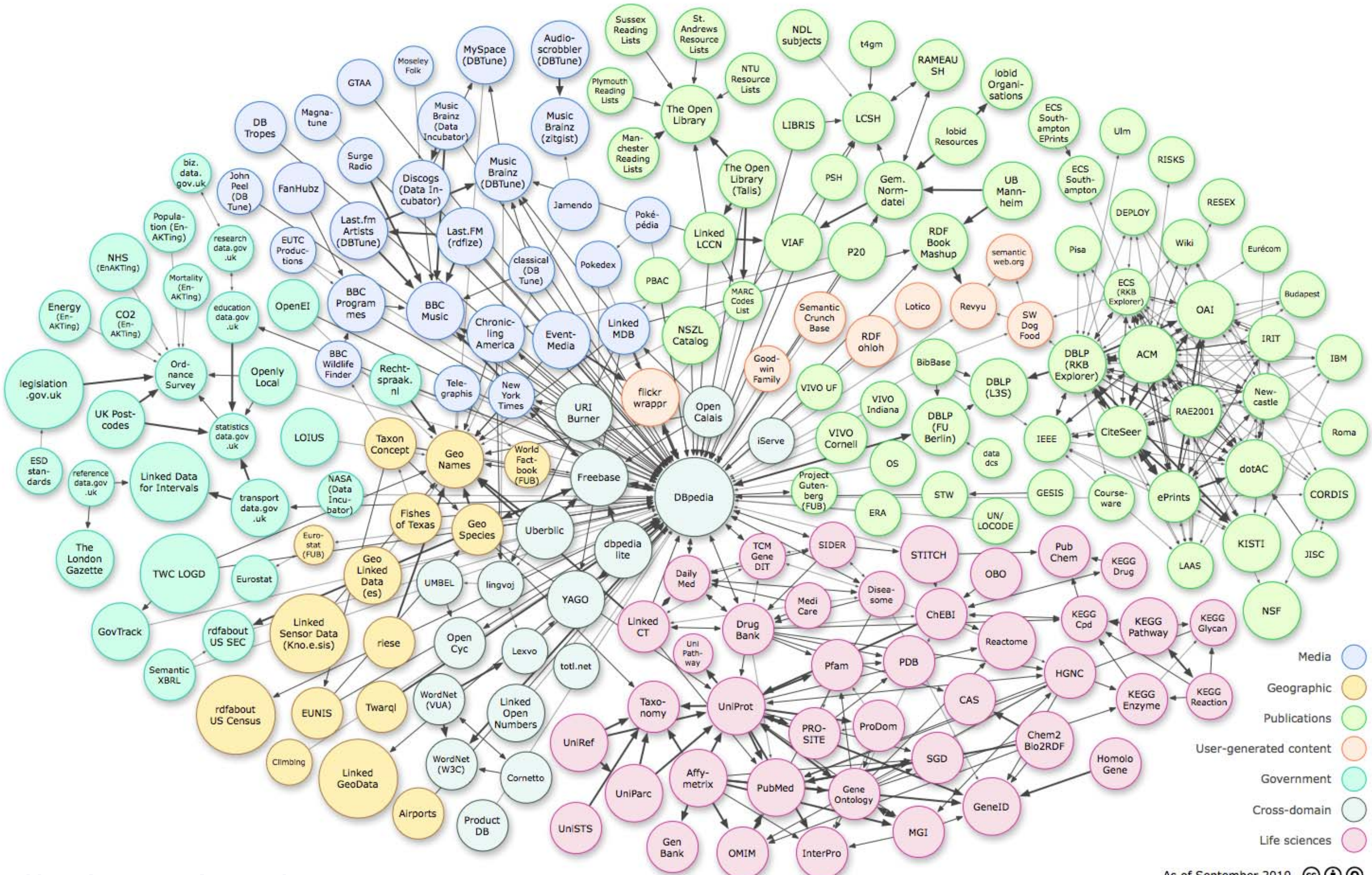
Resource Description Framework

- RDF provides a simple and abstract knowledge representation for resources on the Web which are uniquely identified by *Universal Resource Identifiers* (URIs)

- RDF Schema (RDFS) gives meaning to RDF terms and groups them to concepts

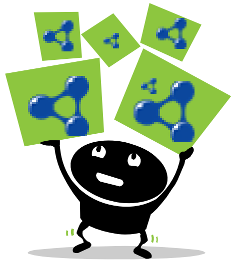
- SPARQL: official W3C recommendation language for querying RDF data

Linked Open Data Cloud



http://lod-cloud.net/

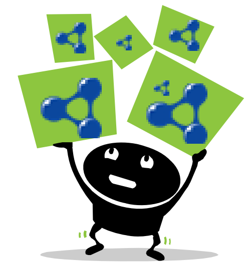
RDF Data Management



- With the vast amount of available RDF data sources on the Web increasing rapidly, there is an urgent need for RDF data management

- RDF storage, query processing and reasoning have been at the center of attention during the last years

RDF Data Management



- Centralized solutions
  - Jena, Sesame, RSSDB, Oracle, RDF-3X, etc.
- Parallel and distributed solutions
  - P2P systems
    - RDFPeers, BabelPeers, GridVine, Edutella etc.
  - Powerful clusters
    - Virtuoso, YARS2, MaRVIN
  - Cloud computing
    - Mika08, Urbani09, Stein10

Peer-to-peer (P2P) systems

- P2P systems provide nice features for Internet-scale applications (e.g., content sharing, distributed digital libraries)

- Distributed Hash Tables (DHTs)
  - answer exact match queries efficiently

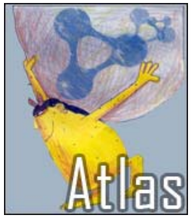
Challenges

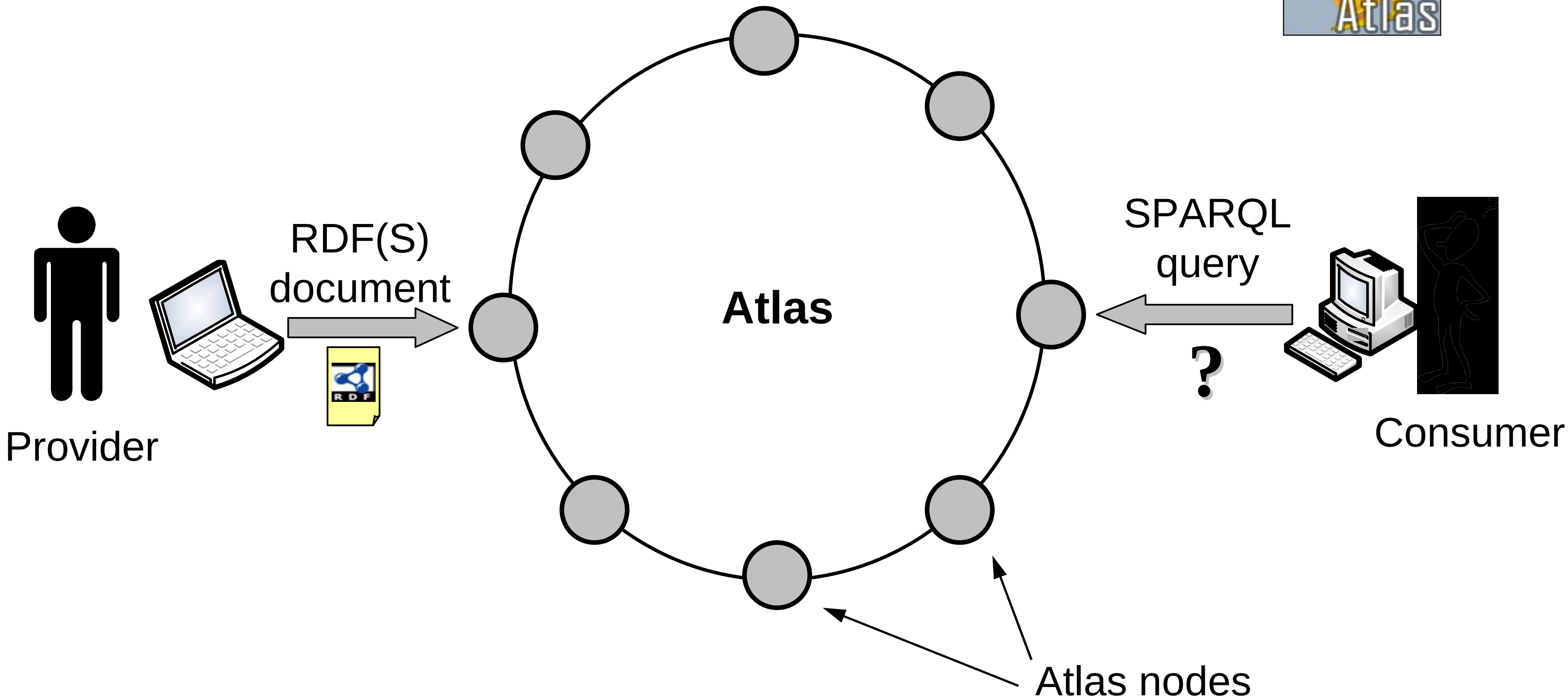
- How to index RDF data and RDFS ontologies in a DHT?
  - ***Triple indexing scheme and handle RDF and RDFS uniformly***
  - ***Distributed mapping dictionary***
- How to answer SPARQL queries efficiently? What kinds of query optimization techniques to use?
  - ***Greedy optimization algorithms to minimize the size of the intermediate results***
  - ***Selectivity estimation techniques***
  - ***RDF statistics in a DHT system***
- How to provide reasoning mechanisms for RDFS in a DHT?
  - ***Distributed forward chaining (FC)***
  - ***Distributed backward chaining (BC)***
  - ***Distributed magic sets transformation (MS)***
  - ***Comparative study (analytically and experimentally)***
  - ***Theoretical proofs of correctness***

Outline

- Introduction
- **The system Atlas**
- RDFS Reasoning in P2P networks
- SPARQL Query Processing and Optimization
- Conclusions

RDF Data Management in P2P networks: the system Atlas





The system Atlas

- **Atlas**: a full-blown open source P2P system for the distributed processing of RDF and RDFS data stored on top of DHTs (http://atlas.di.uoa.gr)

- Atlas has been used in EU projects OntoGrid and SemsorGrid4Env as a distributed registry of metadata

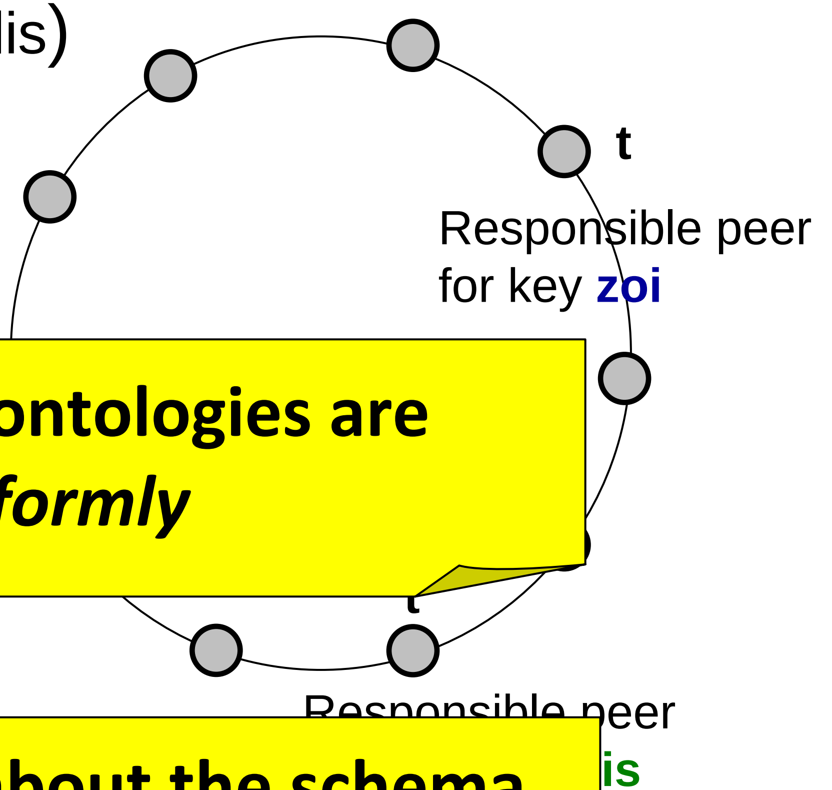
Indexing in Atlas

RDF triple t = (zoi, advisor, manolis)

Index identifier = (Hash(**zoi**))

Index identifier = (Hash(**advisor**))

Index i



RDF data and RDFS ontologies are handled *uniformly*

triple pattern tp = (?x, advisor, ?y)

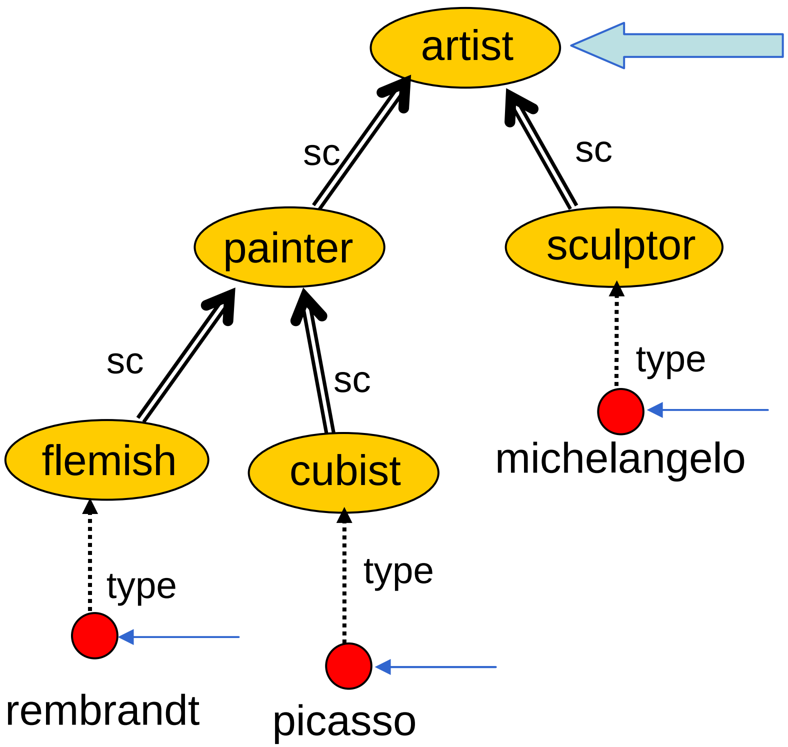
Index identifier = (Hash(**advisor**))

No global knowledge about the schema

Outline

- Introduction
- The system Atlas
- **RDFS Reasoning in P2P networks**
- SPARQL Query Processing and Optimization
- Conclusions

RDFS Reasoning



Query: Find all artists
q = (?x, rdf:type, artist)

RDFS entailment rules

Answer

?x
rembrandt
picasso
michelangelo

RDFS reasoning techniques

- Bottom-up approach
  - compute all inferences a priori (RDFS closure)
- Top-down approach
  - compute inferences at query run time
- Optimized bottom-up approach
  - compute inferences a priori given a specific query

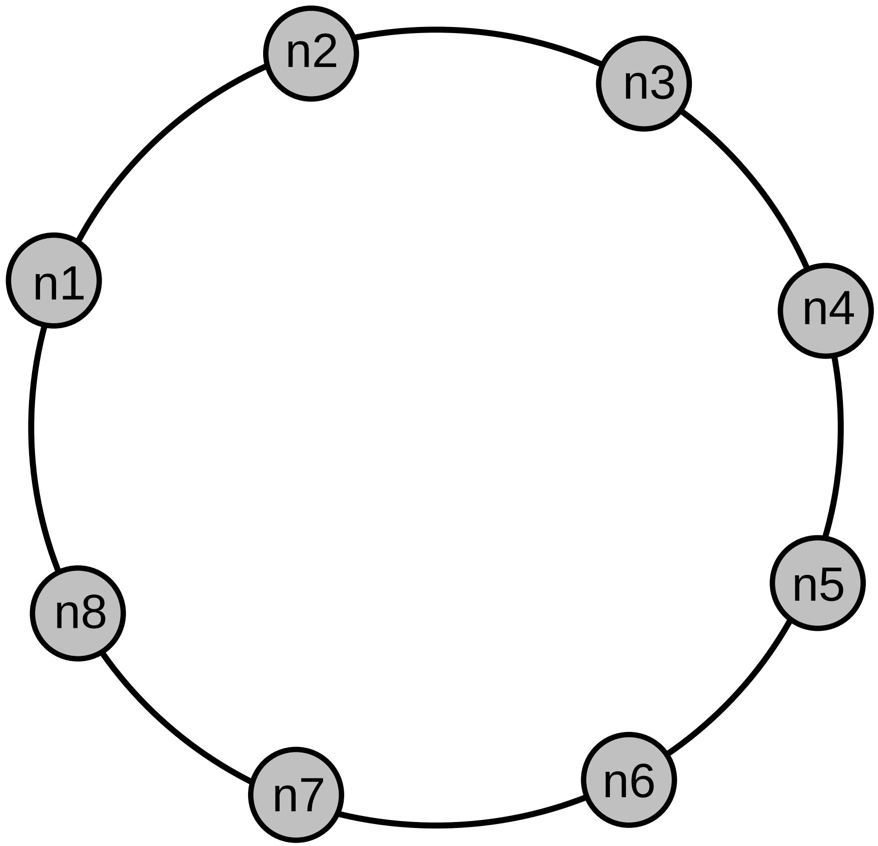
Outline

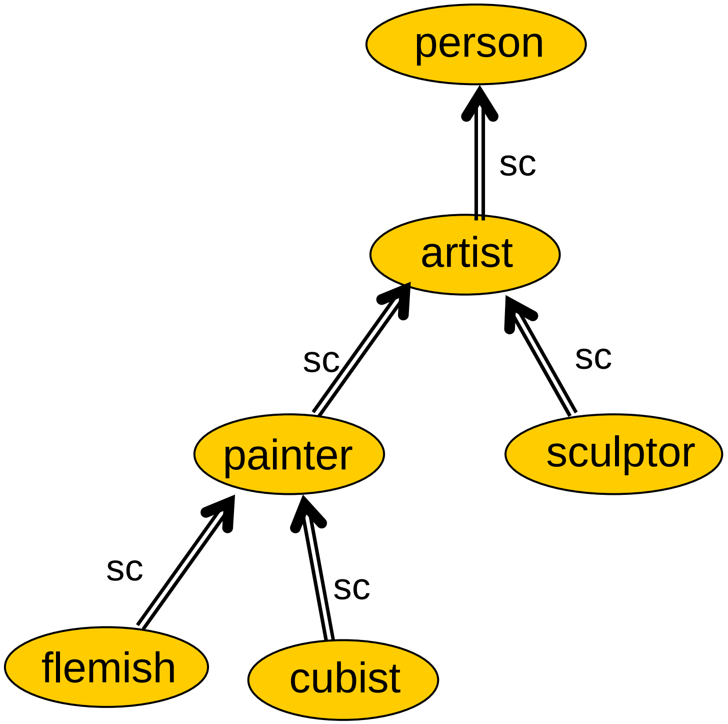
- Introduction
- The system Atlas
- **RDFS Reasoning in P2P networks**
  - **Bottom-up approach**
  - Top-down approach
  - Optimized bottom-up approach
  - Evaluation
- SPARQL Query Processing and Optimization
- Conclusions

Bottom-up approach

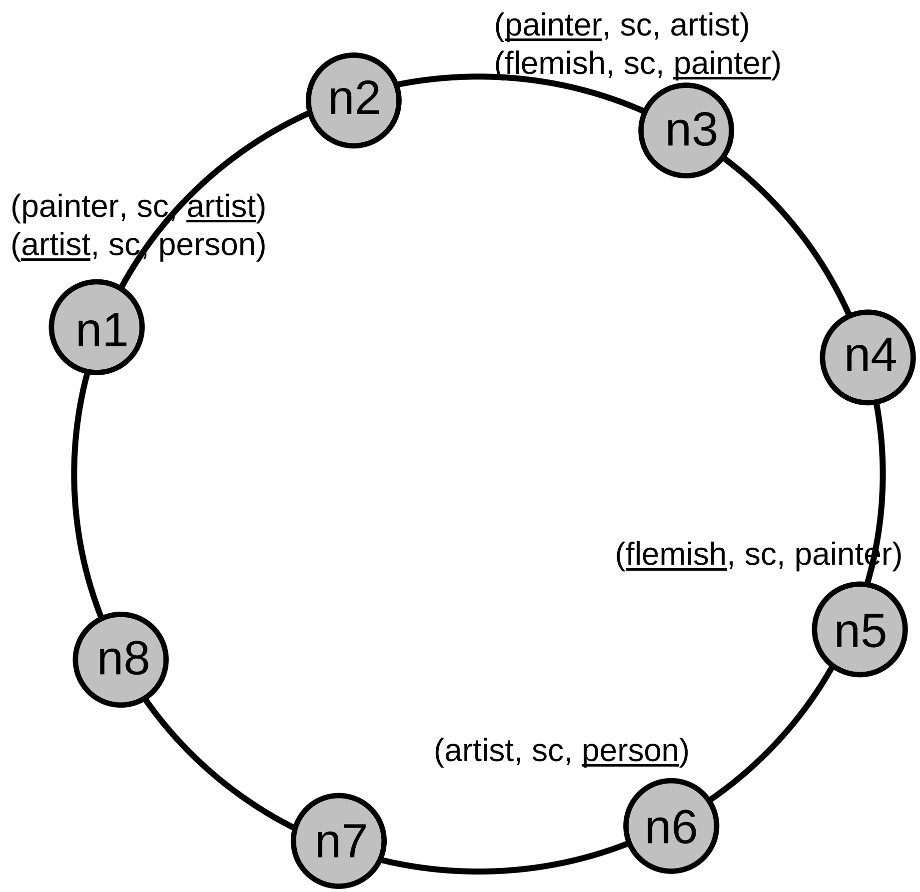
- Distributed *forward chaining* algorithm
  - compute all inferences a priori (RDFS closure)
  - many redundant triples

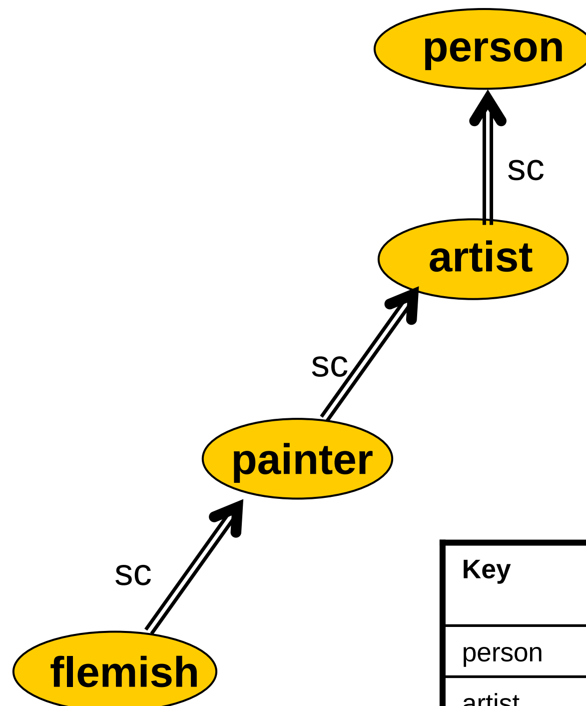
Distributed Forward Chaining (FC)





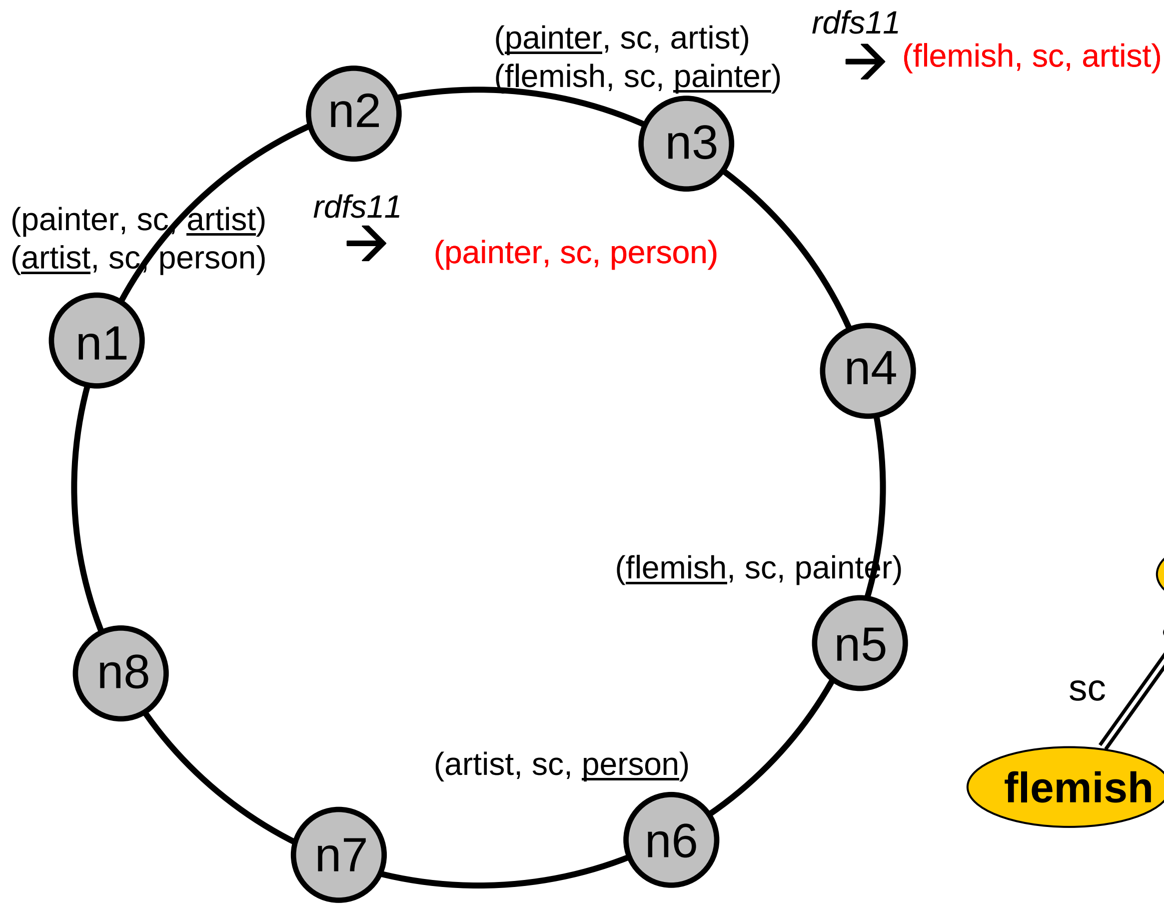
Distributed Forward Chaining (FC)

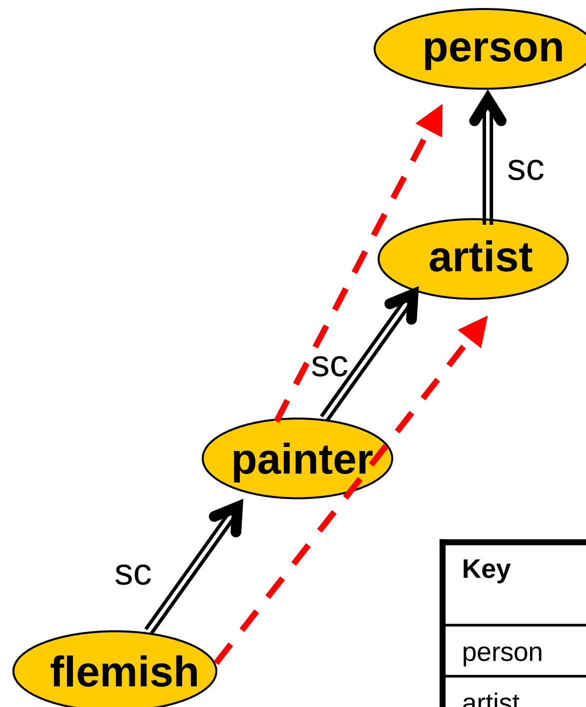




Key	Responsible peer
person	n6
artist	n1
painter	n3
flemish	n5

Distributed Forward Chaining (FC)



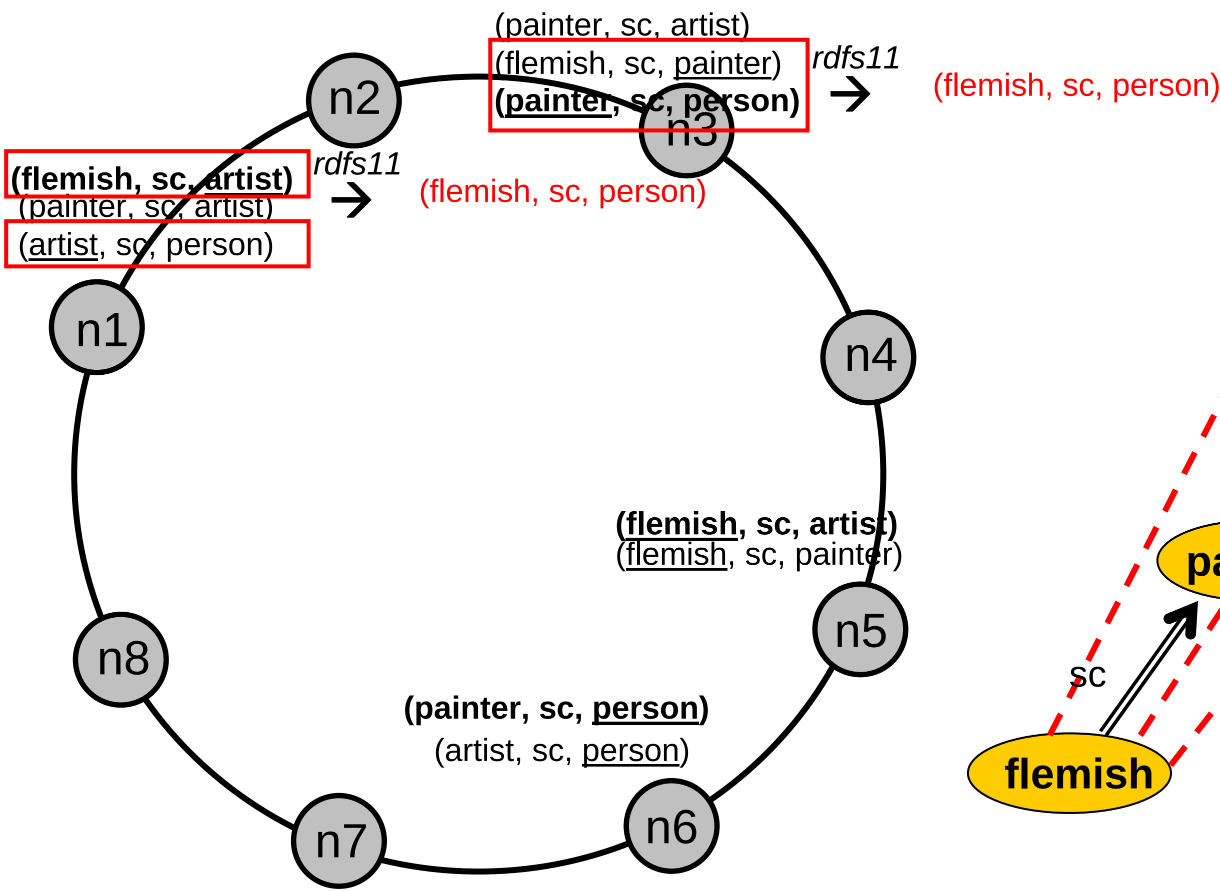


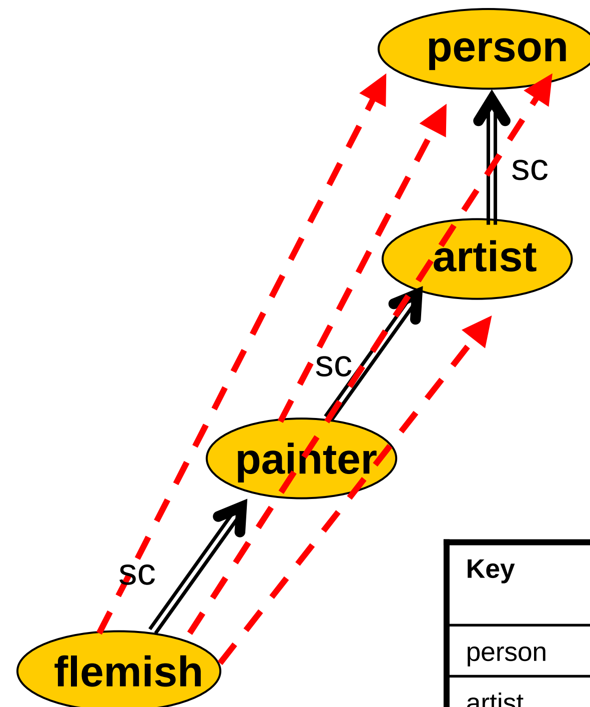
Key	Responsible peer
person	n6
artist	n1
painter	n3
flemish	n5

rdfs11

uuu rdfs:subClassOf vvv **vvv rdfs:subClassOf xxx** →	**uuu rdfs:subClassOf xxx**

Distributed Forward Chaining (FC)



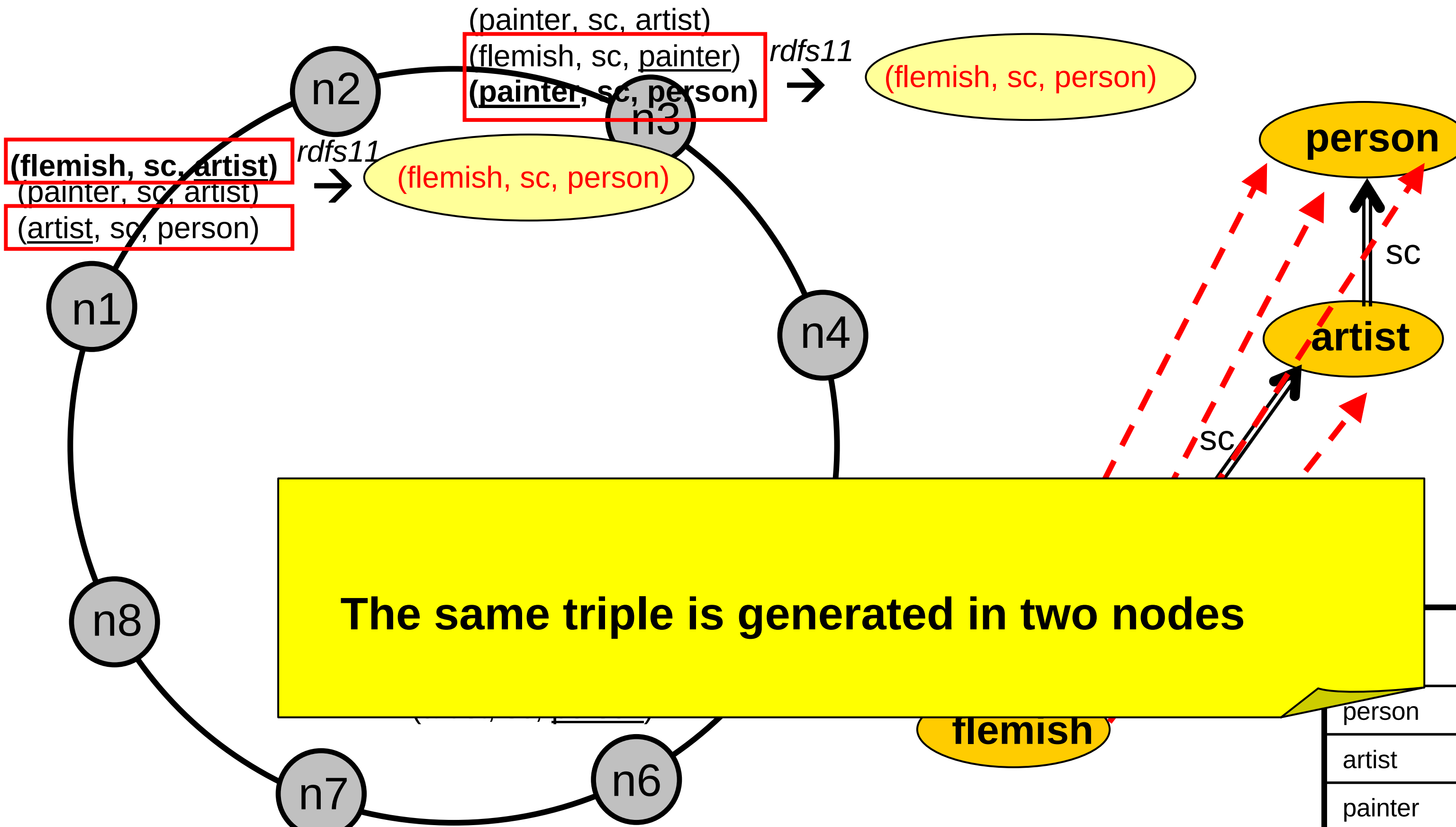


Key	Responsible peer
person	n6
artist	n1
painter	n3
flemish	n5

rdfs11

uuu rdfs:subClassOf vvv **vvv rdfs:subClassOf xxx**	→	**uuu rdfs:subClassOf xxx**

Distributed Forward Chaining (FC)



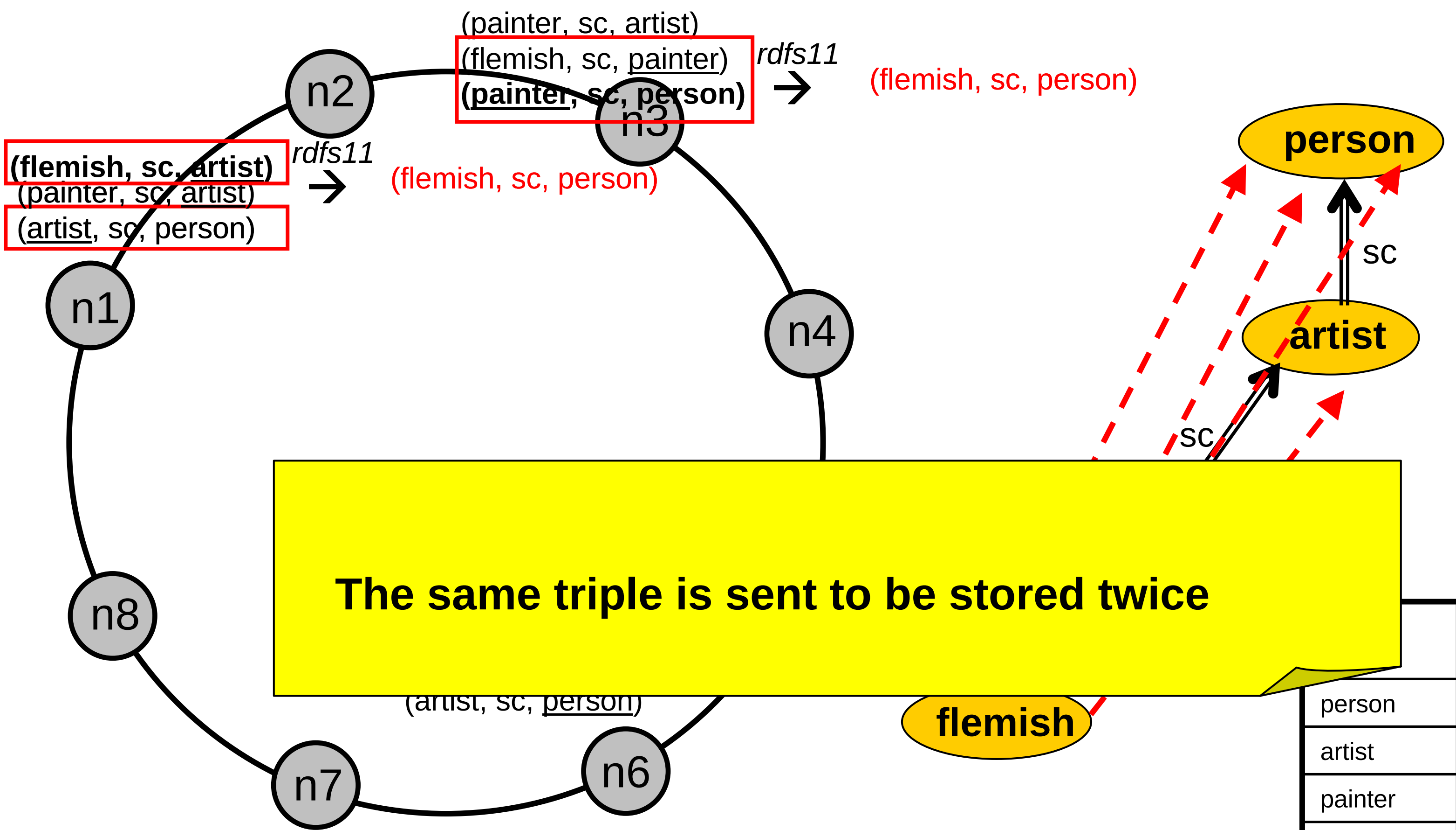
The same triple is generated in two nodes

	Responsible peer
person	n6
artist	n1
painter	n3
flemish	n5

rdfs11

uuu rdfs:subClassOf vvv **vvv rdfs:subClassOf xxx**	→	**uuu rdfs:subClassOf xxx**

Distributed Forward Chaining (FC)



The same triple is sent to be stored twice

	Responsible peer
person	n6
artist	n1
painter	n3
flemish	n5

rdfs11

uuu rdfs:subClassOf vvv **vvv rdfs:subClassOf xxx**	→	**uuu rdfs:subClassOf xxx**

RDFS Entailment Rules

(Only the RDFS rules proposed in the minimal deductive system **mrdf** of [Munoz et al.2009])

- subClass(X,Y) :- triple(X, rdfs:subClassOf, Y).
- subClass(X,Y) :- triple(X, rdfs:subClassOf, Z), subClass(Z, Y).

edb relation: triple

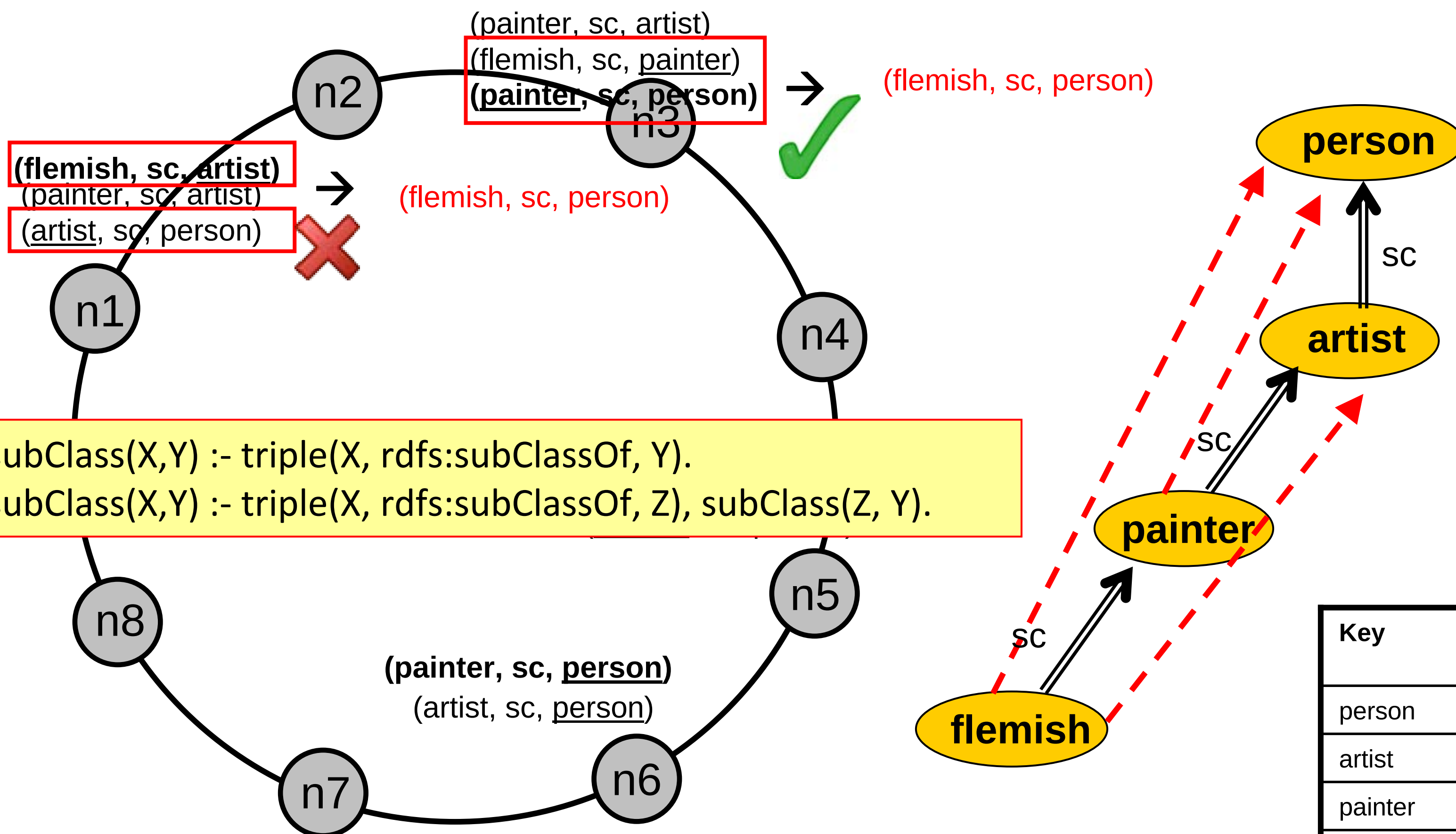
- subProperty(X, Y) :- triple(X, rdfs:subPropertyOf, Y).
- subProperty(X, Y) :- triple(X, rdfs:subPropertyOf, Z), subProperty(Z, Y).

idb relations: subClass, subProperty, newTriple, type

- newTriple(X, P, Y) :- triple(X, P, Y).
- newTriple(X, P, Y) :- triple(X, P1, Z), subProperty(P1, P).

- type(**Safe** , Y).
- type((Z, rdfs:subC **Linear**
- type(X, Y) :- triple(X, P, Z), triple(P, rdfs:domain, Y).
- type(X, Y) :- triple(Z, P, X), triple(P, rdfs:range, Y).

Distributed Forward Chaining (FC*)



subClass(X,Y) :- triple(X, rdfs:subClassOf, Y).
subClass(X,Y) :- triple(X, rdfs:subClassOf, Z), subClass(Z, Y).

Key	Responsible peer
person	n6
artist	n1
painter	n3
flemish	n5

Outline

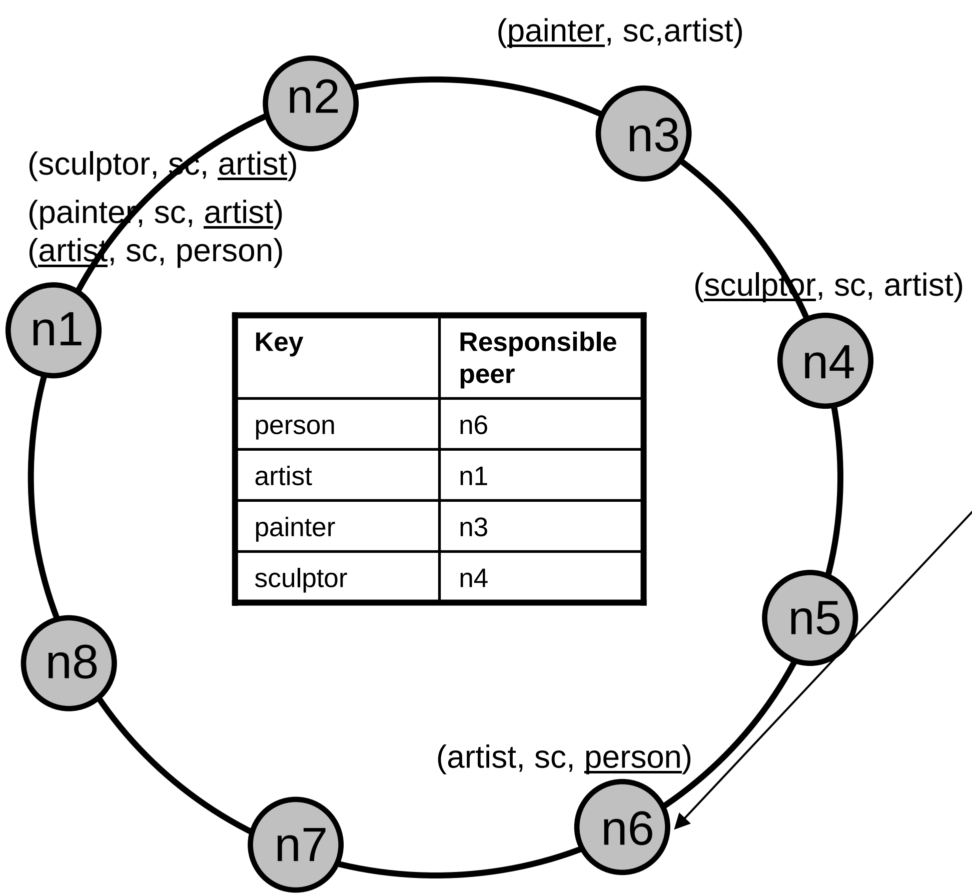
- Introduction
- The system Atlas
- **RDFS Reasoning in P2P networks**
  - Bottom-up approach
  - **Top-down approach**
  - Optimized bottom-up approach
  - Evaluation
- SPARQL Query Processing and Optimization
- Conclusions

Top-down approach

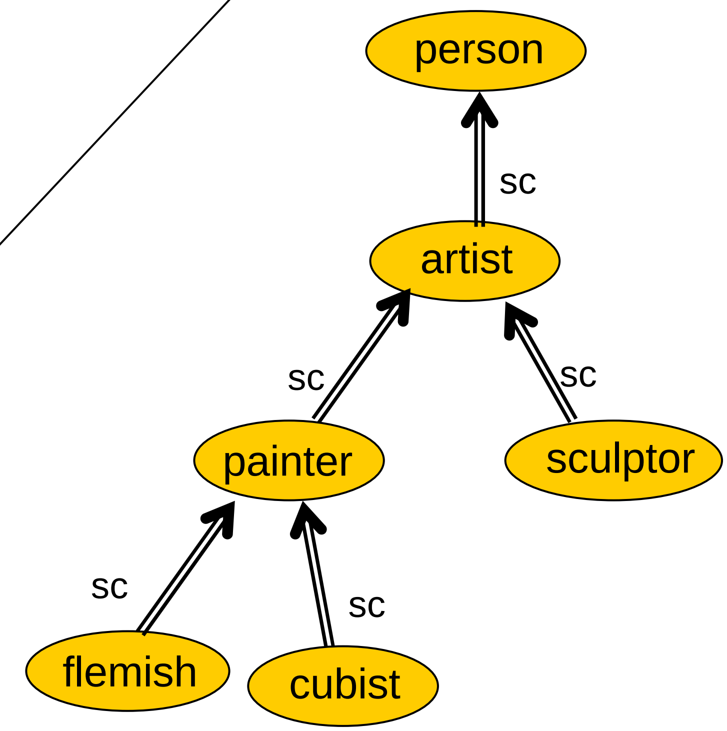
- Distributed *backward chaining* algorithm
  - store only the given triples
  - compute necessary inferences at query run time

Distributed Backward Chaining (BC)

Query: Find all subclasses of person
q = (X, sc, person)

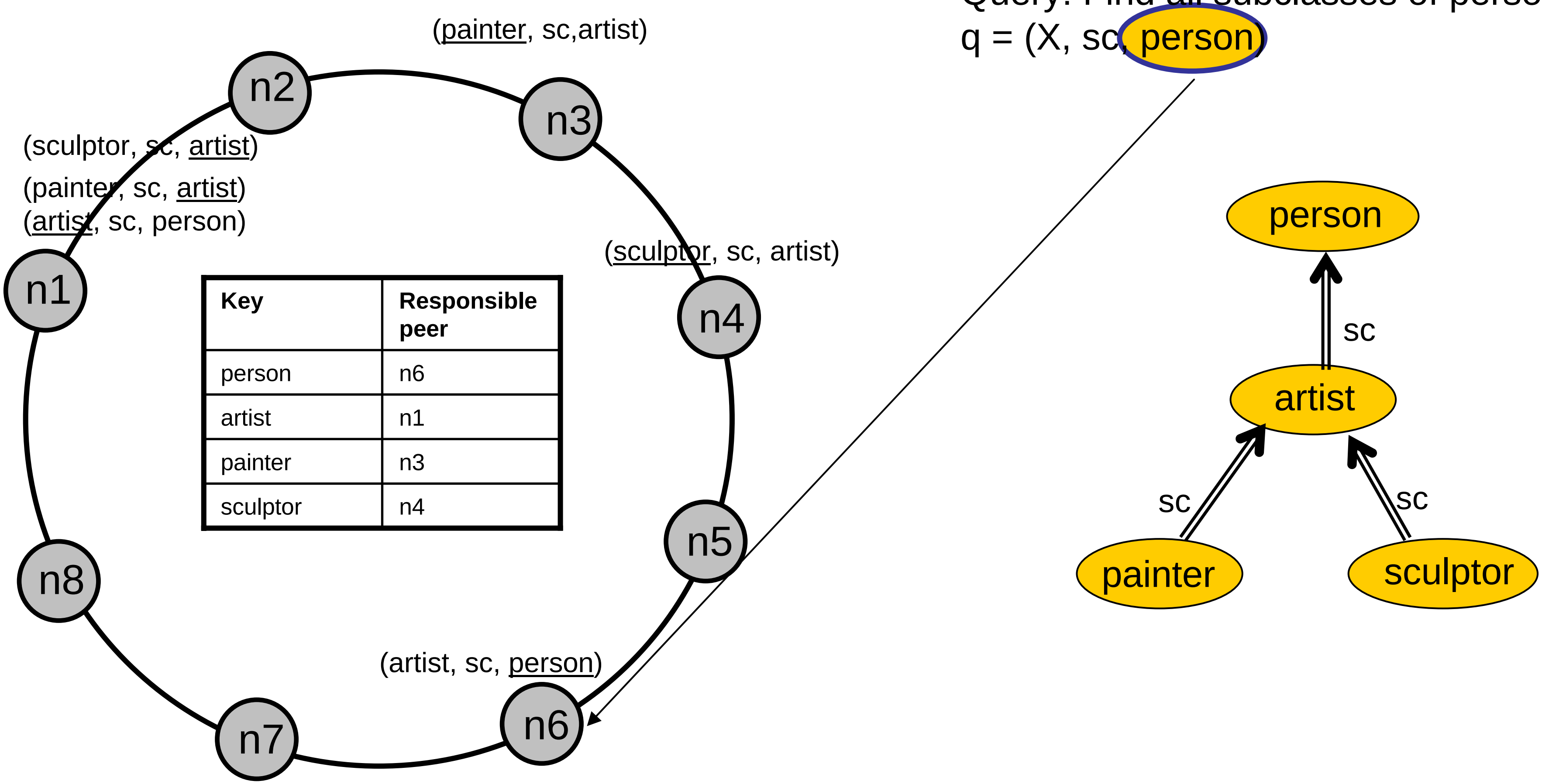


Key	Responsible peer
person	n6
artist	n1
painter	n3
sculptor	n4



Distributed Backward Chaining (BC)

Query: Find all subclasses of person
q = (X, sc, person)



Key	Responsible peer
person	n6
artist	n1
painter	n3
sculptor	n4

Distributed Backward Chaining (BC)

subClass (X, person)

n6

Which predicate should we evaluate first?

subClass (X,Y) :- triple (X, rdfs:subClassOf, Y).
subClass (X,Y) :- triple(X, rdfs:subClassOf, Z), subClass(Z, Y).

Evaluate first the predicate that can be evaluated locally!

RDFS Entailment Rules - revisited

- Recursive rules
- Rule adornment from recursive query processing
  - Good orderings for evaluating predicates
  - eg. subClass(X, artist) → subClass^{fb} (X,Y)
- Extended adornment
  - Ordered string of *f, b,* ***k***
    - ***k*** **: an argument that is bound and the <u>key</u>**
    - *b* : bound argument (not the key)
    - *f* : free argument
  - eg. At node responsible for key artist:
    triple(X, rdf:type, artist) → triple^{fbk} (X, rdf:type, Y).

  - Good ordering for evaluating predicates in a distributed environment

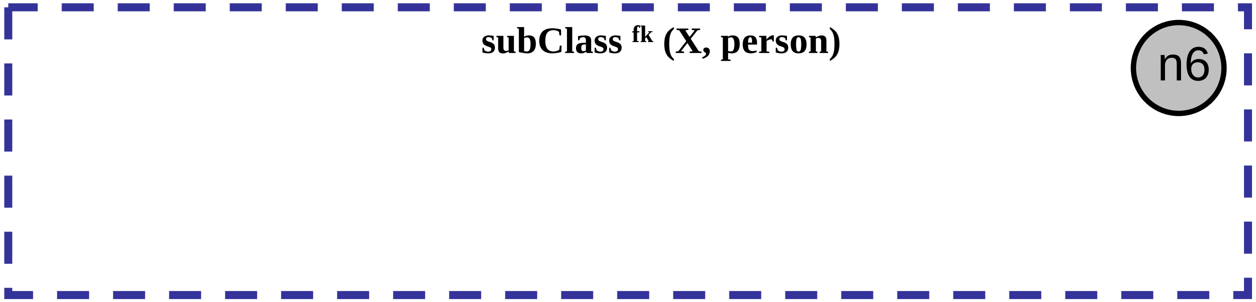
RDFS Entailment Rules - revisited

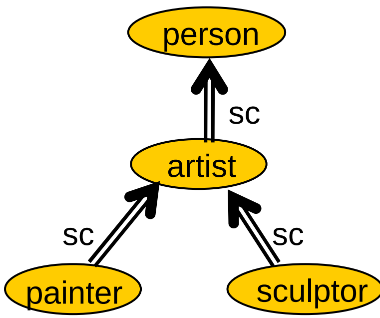
- $subClass^{kf}$ (X,Y) :- $triple^{\mathbf{k}bf}$ (X, rdfs:subClassOf, Y).
- $subClass^{kf}$ (X,Y) :- $triple^{\mathbf{k}bf}$ (X, rdfs:subClassOf, Z), $subClass^{ff}$ (Z, Y).
- $subClass^{fk}$ (X,Y) :- $triple^{fb\mathbf{k}}$ (X, rdfs:subClassOf, Y).
- $subClass^{fk}$ (X,Y) :- $subClass^{ff}$(X, Z), $triple^{fb\mathbf{k}}$ (Z, rdfs:subClassOf, Y).

- $subProperty^{kf}$ (X,Y) :- $triple^{\mathbf{k}bf}$ (X, rdfs:subPropertyOf, Y).
- $subProperty^{kf}$ (X,Y) :- $triple^{\mathbf{k}bf}$ (X, rdfs:subPropertyOf, Z), $subProperty^{ff}$ (Z, Y).
- $subProperty^{fk}$ (X,Y) :- $triple^{fb\mathbf{k}}$ (X, rdfs:subPropertyOf, Y).
- $subProperty^{fk}$ (X,Y) :- $subProperty^{ff}$(X, Z), $triple^{fb\mathbf{k}}$ (Z, rdfs:subPropertyOf, Y).

- $type^{kf}$ (X, Y) :- $triple^{\mathbf{k}bf}$ (X, rdf:type, Y).
- $type^{kf}$ (X, Y) :- $triple^{\mathbf{k}ff}$ (X, P, Z), $triple^{fbf}$ (P, rdfs:domain, Y).
- $type^{kf}$ (X, Y) :- $triple^{ff\mathbf{k}}$ (Z, P, X), $triple^{fbf}$ (P, rdfs:range, Y).
- $type^{kf}$ (X, Y) :- $triple^{\mathbf{k}bf}$ (X, rdf:type, Z), $subClass^{ff}$ (Z, Y).
- $type^{fk}$ (X, Y) :- $triple^{fb\mathbf{k}}$ (X, rdf:type, Y).
- $type^{fk}$ (X, Y) :- $triple^{fff}$ (X, P, Z), $triple^{fb\mathbf{k}}$ (P, rdfs:domain, Y).
- $type^{fk}$ (X, Y) :- $triple^{fff}$ (Z, P, X), $triple^{fb\mathbf{k}}$ (P, rdfs:range, Y).
- $type^{fk}$ (X, Y) :- $type^{ff}$ (X, Z), $triple^{fb\mathbf{k}}$ (Z, rdfs:subClassOf, Y).

Distributed Backward Chaining (BC)





Key	Responsible peer
person	n6
artist	n1
painter	n3
sculptor	n4

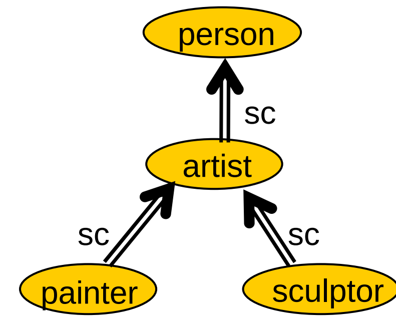
Distributed Backward Chaining (BC)

subClass ^{fk} (X, person)

r1: triple ^{fbk} (X, rdfs:subClassOf, person)

r2: triple ^{fbk} (Z, rdfs:subClassOf, person)
subClass ^{ff} (X, Z)

n6



subClass^{fk} (X,Y) :- triple^{fbk} (X, rdfs:subClassOf, Y). **(r1)**
subClass^{fk} (X,Y) :- subClass^{ff}(X, Z), triple^{fbk} (Z, rdfs:subClassOf, Y). **(r2)**

Key	**Responsible peer**
person	n6
artist	n1
painter	n3
sculptor	n4

Distributed Backward Chaining (BC)

subClass ^{fk} (X, person)

n6

r1

r2

triple ^{fbk} (X, rdfs:subClassOf, person)

triple ^{fbk} (Z, rdfs:subClassOf, person)
subClass ^{ff} (X, Z)

Z / artist

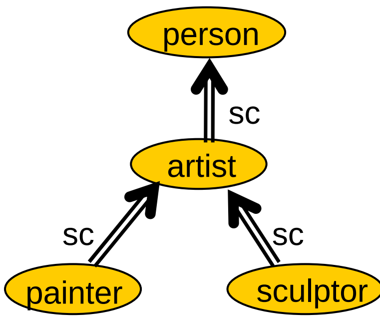
subClass^{fk} (X,Y) :- triple^{fbk} (X, rdfs:subClassOf, Y). **(r1)**
subClass^{fk} (X,Y) :- subClass^{ff}(X, Z), triple^{fbk} (Z, rdfs:subClassOf, Y). **(r2)**

triple ^{fbk} (X, rdfs:subClassOf, artist)

triple ^{fbk} (Z, rdfs:subClassOf, artist)
subClass ^{ff} (X, Z)

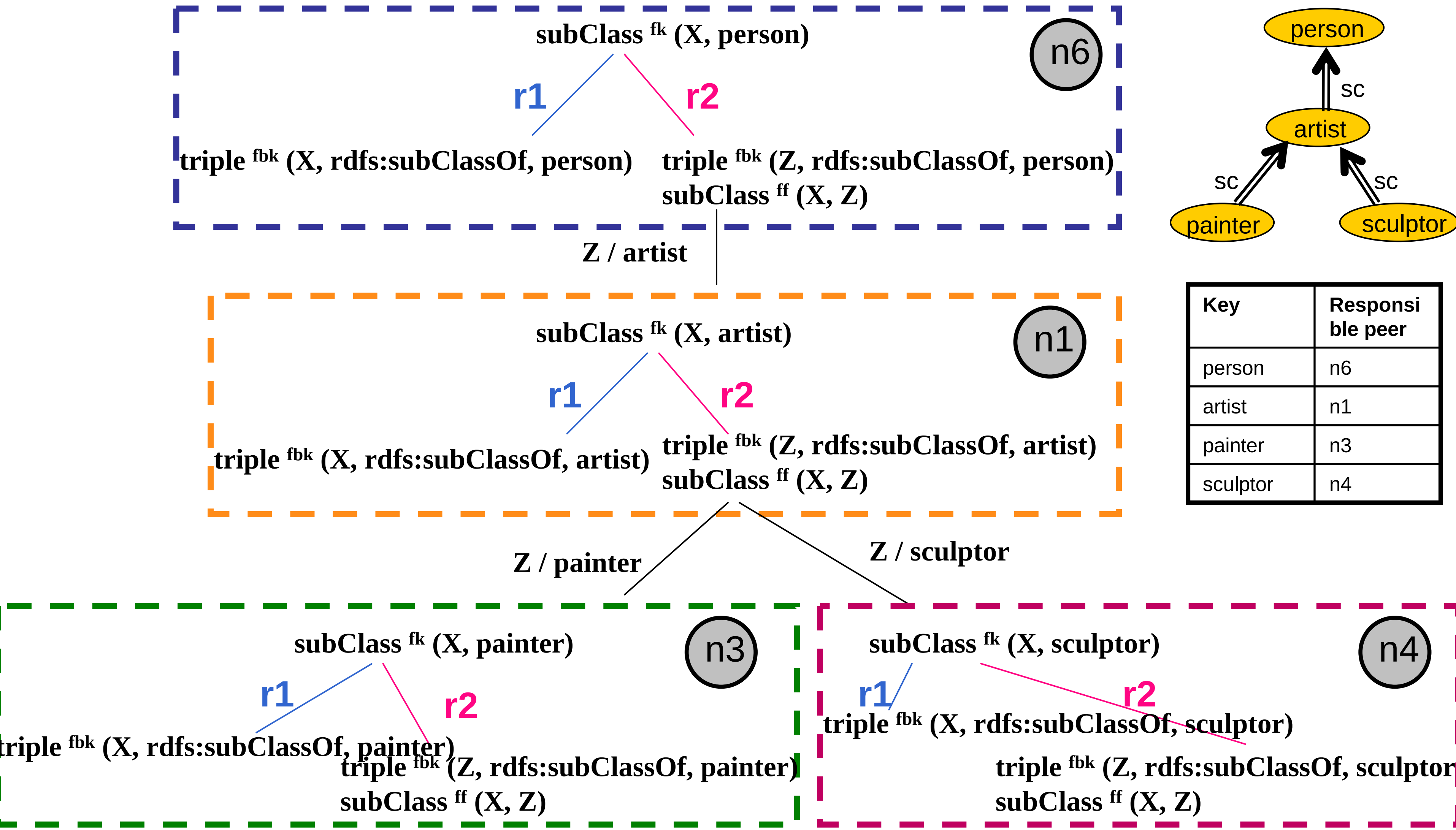
Z / painter

Z / sculptor



Key	Responsible peer
person	n6
artist	n1
painter	n3
sculptor	n4

Distributed Backward Chaining (BC)



Key	Responsible peer
person	n6
artist	n1
painter	n3
sculptor	n4

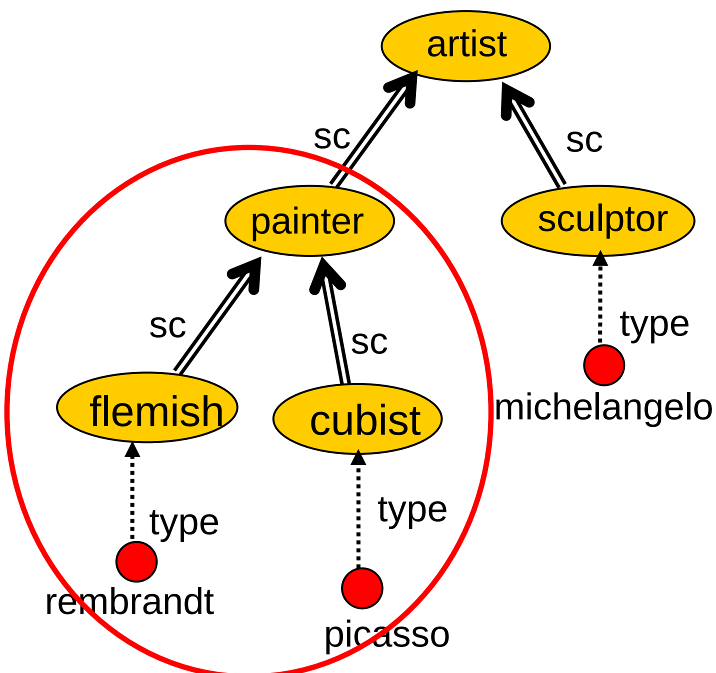
Outline

- Introduction
- The system Atlas
- **RDFS Reasoning in P2P networks**
  - Bottom-up approach
  - Top-down approach
  - **Optimized bottom-up approach**
  - Evaluation
- SPARQL Query Processing and Optimization
- Conclusions

Forward chaining for magic rules (MS)

- Magic sets transformation technique
  - Rewrite rules given a specific query
  - Bottom-up evaluation (forward chaining)
  - No unnecessary information is inferred

Rule	Head	Body
1	m_newTriple(P)	sup_{61}(P, Y)
2	m_newTriple(P)	sup_{71}(P, Y)
3	m_subProperty(Z)	sup_{21}(Z,Y)
4	m_subProperty(P)	m_newTriple(P)
5	m_subClass(Z)	sup_{41}(Z,Y)
6	m_type(Z)	sup_{51}(Z,Y)
7	sup_{21}(Z, Y)	m_subProperty(Y), triple(Z, sp, Y)
8	sup_{31}(P1, P)	m_newTriple(P), subProperty(P1, P)
9	sup_{41}(Z, Y)	m_subClass (Y), triple(Z, sc, Y)
10	sup_{51}(Z, Y)	m_type(Y), triple(Z, sc, Y)
11	sup_{61}(P, Y)	m_type(Y), triple(P, dom, Y)
12	sup_{71}(P, Y)	m_type(Y), triple(P, range, Y)
13	newTriple(X, P ,Y)	m_newTriple(P), triple(X, P, Y)
14	subProperty(X, Y)	m_subProperty(Y), triple(X, sp, Y)
15	subProperty(X, Y)	sup_{21}(Z,Y), subProperty(X, Z)
16	newTriple(X, P, Y)	sup_{31}(P1,P), triple(X,P1,Y)
17	subClass(X, Y)	m_subClass(Y), triple(X, sc, Y)
18	subClass(X, Y)	sup_{41}(Z,Y), subClass(X, Z)
19	type(X, Y)	m_type(Y), triple(X, type, Y)
20	type(X, Y)	sup_{51}(Z, Y), type(X, Z)
21	type(X, Y)	sup_{61}(P, Y), newTriple(X, P, Z)
22	type(X, Y)	sup_{71}(P, Y), newTriple(Z, P, X)
23	m_type(a)	



Query: Find all painters
q = (?x, rdf:type, painter)

Outline

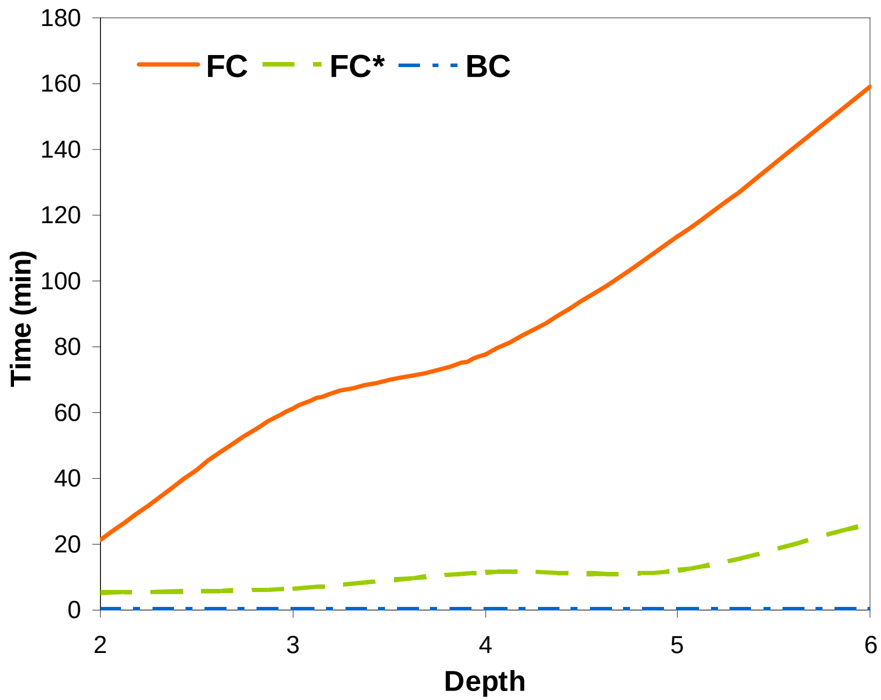
- Introduction
- The system Atlas
- **RDFS Reasoning in P2P networks**
  - Bottom-up approach
  - Top-down approach
  - Optimized bottom-up approach
  - **Evaluation**
- SPARQL Query Processing and Optimization
- Conclusions

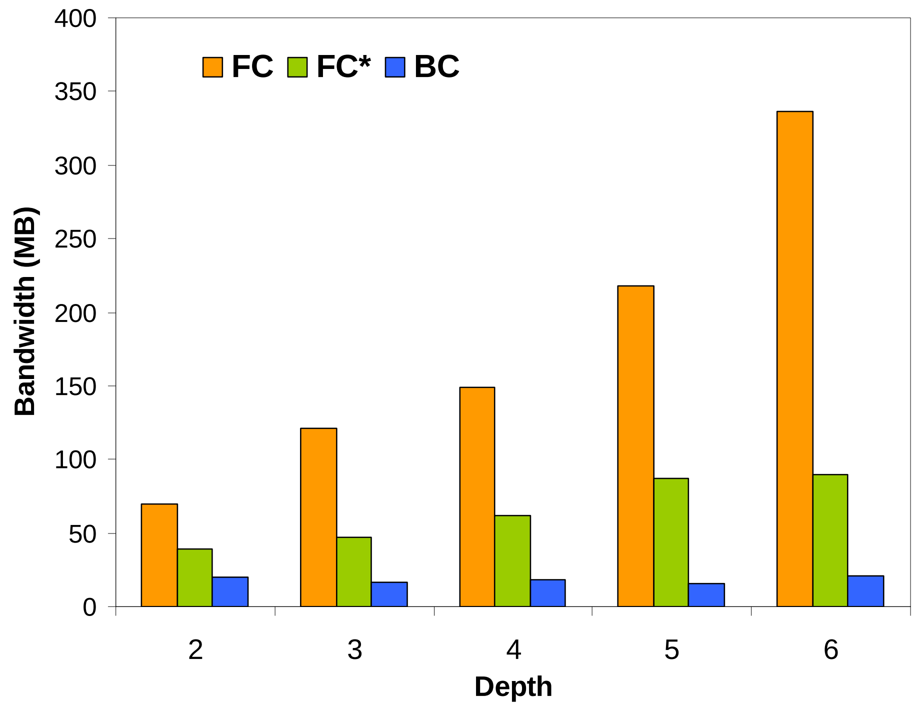
Experimental setup

- Algorithms have been implemented in Atlas system using Bamboo DHT [Rhea04]
- Testbeds
  - **PlanetLab** (~210 nodes available at the time of the experiments)
  - **local cluster** (30 machines – 4 Atlas nodes per machine)
- Datasets
  - Synthetic data from RBench generator [Theoharis05]
    - Number of instances: 10,000 to 1M
    - RDFS class hierarchy tree depth: 2-6 (7 to 128 classes)
    - Query: Give me the instances of the root class
  - LUBM benchmark
    - Number of triples: ~110,000 to ~2,700,00
    - Query: Give me the instances of a class

Store time and network traffic

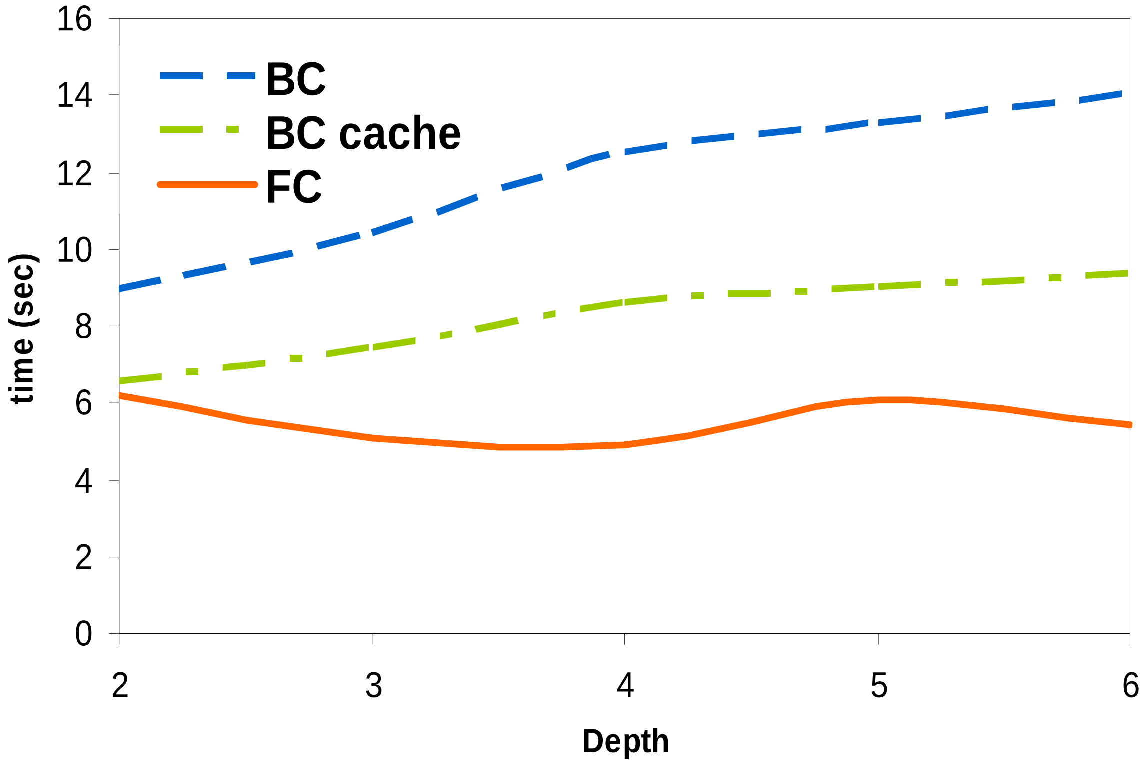
PlanetLab, RBench with 10,000 instances





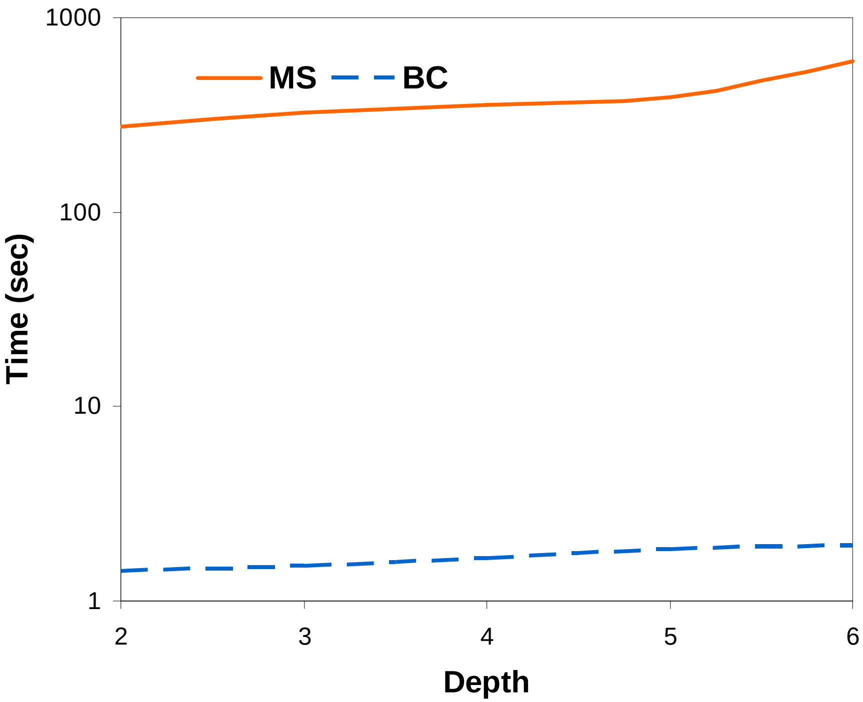
Query response time

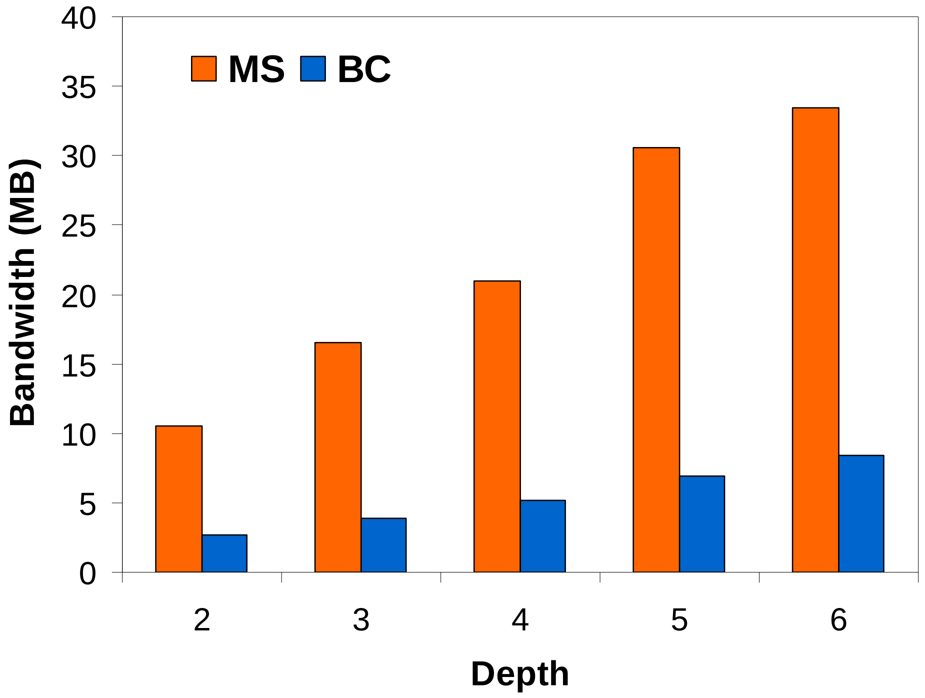
PlanetLab, RBench with 10,000 instances



Backward chaining vs. magic sets

Cluster: RBench with 1M instances





Outline

- Introduction
- The system Atlas
- RDFS Reasoning in P2P networks
- **SPARQL Query Processing and Optimization**
- Conclusions

SPARQL

- SPARQL queries of basic graph patterns

```
SELECT ?x ?y ?z
WHERE {
   ?x rdf:type ub:Student .  (tp1)
   ?x ub:takesCourse ?z .    (tp2)
   ?x ub:advisor ?y .        (tp3)
   ?y ub:teacherOf ?z .      (tp4)
 }
```

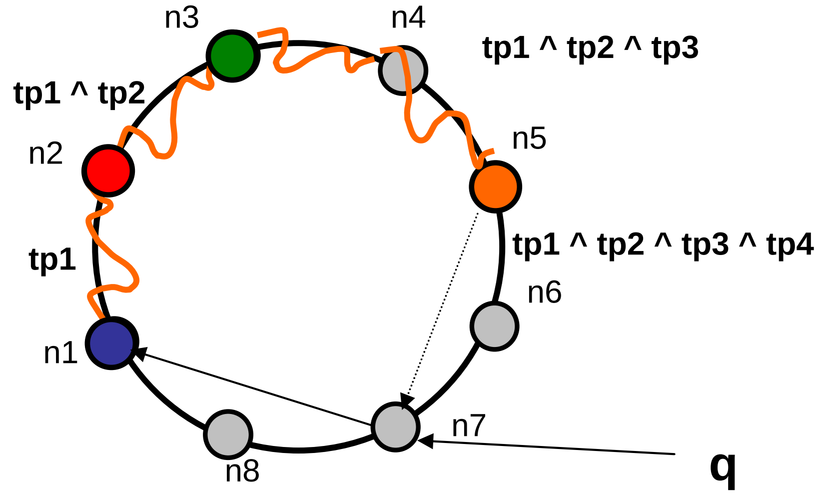
  - conjunctive triple patterns

```
tp1 ^ tp2 ^ tp3 ^ tp4
```

SPARQL query processing

- SPARQL queries of basic graph patterns

```
SELECT ?x ?y ?z
WHERE {
   ?x rdf:type ub:Student .(tp1)
   ?x ub:takesCourse ?z .  (tp2)
   ?x ub:advisor ?y .      (tp3)
   ?y ub:teacherOf ?z      (tp4)
 }
```



Query Optimization

- Find a query plan that optimizes the performance of a system with respect to a metric of interest

Query response time

Bandwidth consumption

- Find a good ordering of the triple patterns

Minimize the size of intermediate results

Lower bandwidth consumption

Joins with smaller intermediate relations

Query Optimization

- Greedy optimization algorithms
  - **selectivity-based** heuristics
  - **minimize** the size of **intermediate results**

Outline

- Introduction
- The system Atlas
- RDFS Reasoning in P2P networks
- **SPARQL Query Processing and Optimization**
  - **Selectivity estimation**
  - Statistics for RDF
  - Optimization algorithms
  - Evaluation
- Conclusions

Selectivity estimation

- Single triple pattern
  - Bound-is-easier heuristic
  - Analytical estimation: $sel(tp) = sel(s) \times sel(p) \times sel(o)$

$$sel(v) = \frac{freq_c(v)}{T}$$

frequency of value v as a component c

total triples stored in the network

- Joins of triple patterns: $sel(tp_1 \wedge tp_2) = \frac{join\_card(tp_1, tp_2)}{T^2}$

$$join\_card(tp_1, tp_2) = \frac{T_{tp_1} \times T_{tp_2}}{\max(I_{tp_1(?x)}, I_{tp_2(?x)})}$$

number of triples matching tp_1, tp_2

size of the domain of ?x in tp_i

Outline

- Introduction
- The system Atlas
- RDFS Reasoning in P2P networks
- **SPARQL Query Processing and Optimization**
  - Selectivity estimation
  - **Statistics for RDF**
  - Optimization algorithms
  - Evaluation
- Conclusions

Required statistics

- Frequency of a triple component *c* with value *v* ($\boldsymbol{freq_c(v)}$)

- Size of the domain of a variable in a triple pattern ($\boldsymbol{I_{tpi(?x)}}$)

Frequency and variable domain size - example

(?x, advisor, ?y)

frequency of advisor → # occurrences of advisor as a **predicate** (**freq$_p$(*advisor*)**)

domain size of ?x → # **distinct subject** values of **predicate** *advisor* (**ds$_p$(*advisor*)**)

domain size of ?y → # **distinct object** values of **predicate** *advisor* (**do$_p$(*advisor*)**)

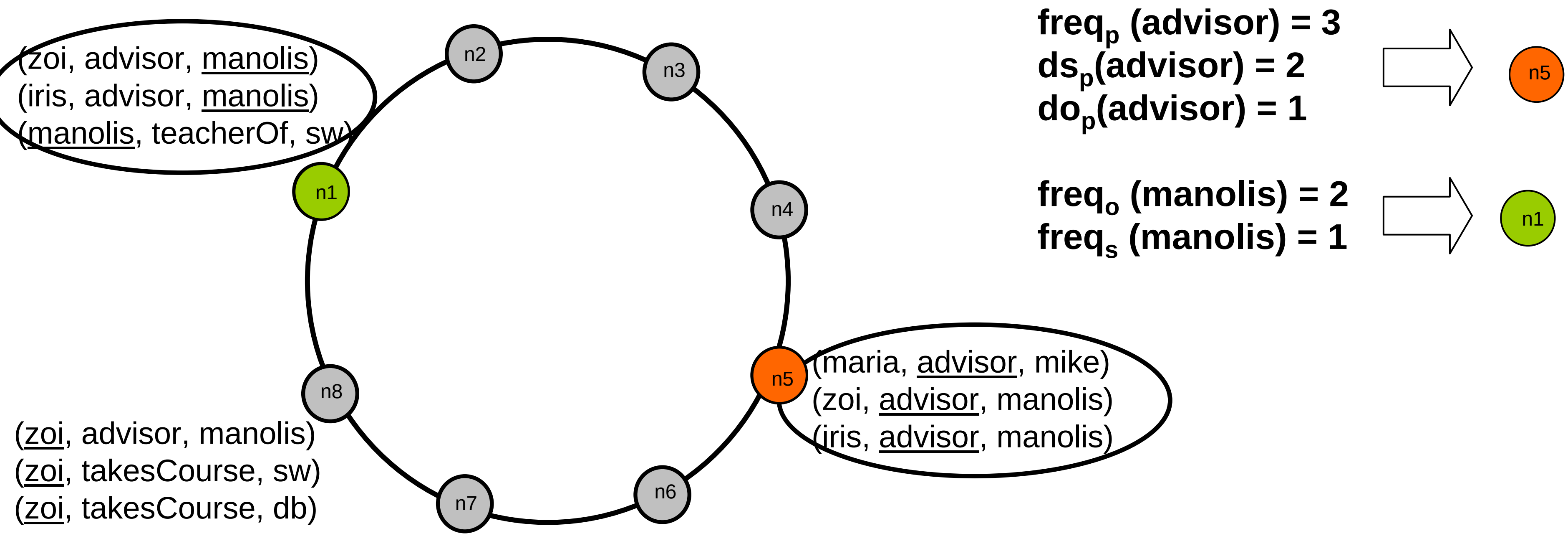
RDF triples

subject	predicate	object
zoi	*advisor*	*manolis*
iris	*advisor*	*manolis*
maria	*advisor*	*mike*
zoi	*takesCourse*	*db*
zoi	*takesCourse*	*sw*
manolis	*teacherOf*	*sw*

statistics for predicates

Value	freq$_p$	ds$_p$	do$_p$
advisor	**3**	**3**	**2**
takesCourse	2	1	2
teacherOf	1	1	1

Statistics in DHTs



Each peer is responsible for creating and maintaining statistics of its **locally** stored data and only for triple component values which are <u>**key**</u> at the specified peer

Statistics at each DHT peer

- Each peer creates statistics for each triple component separately
  - distinguish resource objects from class objects

subject	predicate	object	object-class
$freq_s$	$freq_p$	$freq_o$	$freq_c$
dp_s	ds_p	ds_o	–
do_s	dp_p	dp_o	–

- Given a space budget, each peer can keep the ***exact frequency distribution*** or an estimation by using **v-optimal *histograms***

Outline

- Introduction
- The system Atlas
- RDFS Reasoning in P2P networks
- **SPARQL Query Processing and Optimization**
  - Selectivity estimation
  - Statistics for RDF
  - **Optimization algorithms**
  - Evaluation
- Conclusions

Optimization algorithms

- **Static optimization**
  - Before the query evaluation begins, at the peer that receives the query request
    - **Naïve optimization algorithm (NA):** Orders triple patterns based on their selectivity, from the most selective to the least selective
    - **Semi-naïve optimization algorithm (SNA):** Orders triple patterns based on the join selectivity between pairs of triple patterns

Optimization algorithms

- **Dynamic optimization**
  - During the query evaluation, at each peer participating in the query processing
    - **Dynamic optimization algorithm (DA)**: join selectivity is computed at each peer participating in the query processing and estimated between the **real** intermediate results so far and a triple pattern

- All algorithms (static and dynamic) use a query graph (QG) representation to avoid Cartesian products

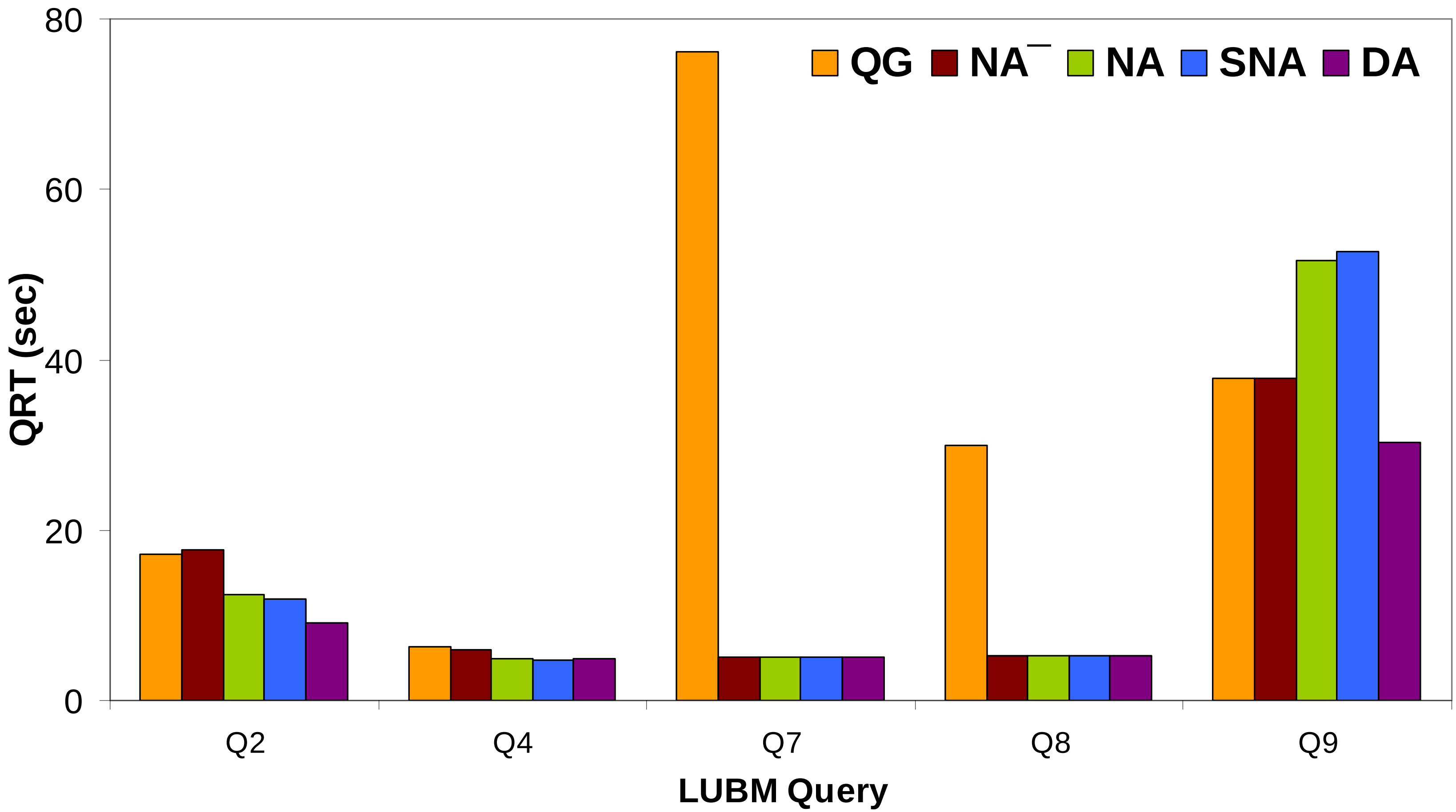
Outline

- Introduction
- The system Atlas
- RDFS Reasoning in P2P networks
- **SPARQL Query Processing and Optimization**
  - Selectivity estimation
  - Statistics for RDF
  - Optimization algorithms
  - **Evaluation**
- Conclusions

Experimental setup

- All algorithms and techniques have been implemented in Atlas
  - Mapping dictionary
- We used as a testbed PlanetLab and a **local cluster** server blade machines with two processors at 2.6GHz and 4GB memory 30 available machines – up to 4 nodes per machine → up to 120 DHT peers
- LUBM Benchmark (queries with more than 4 triple patterns)

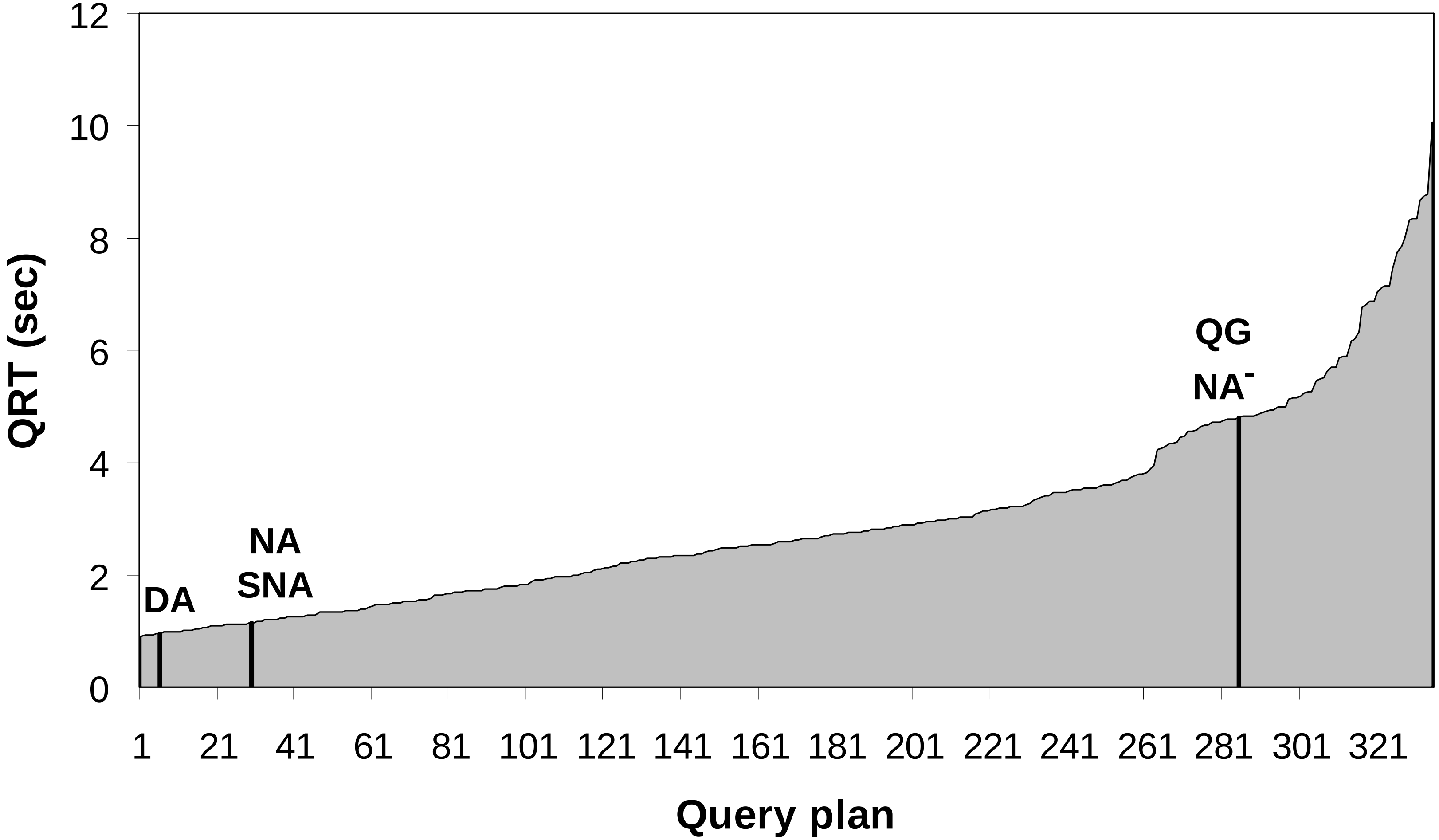
Query response time (LUBM-50)



Q7: select ?X ?Y where { ?X rdf:type ub:Student . ?Y rdf:type ub:Course . ub:AssociateProfessor147 ub:teacherOf ?Y. ?X ub:takesCourse ?Y}

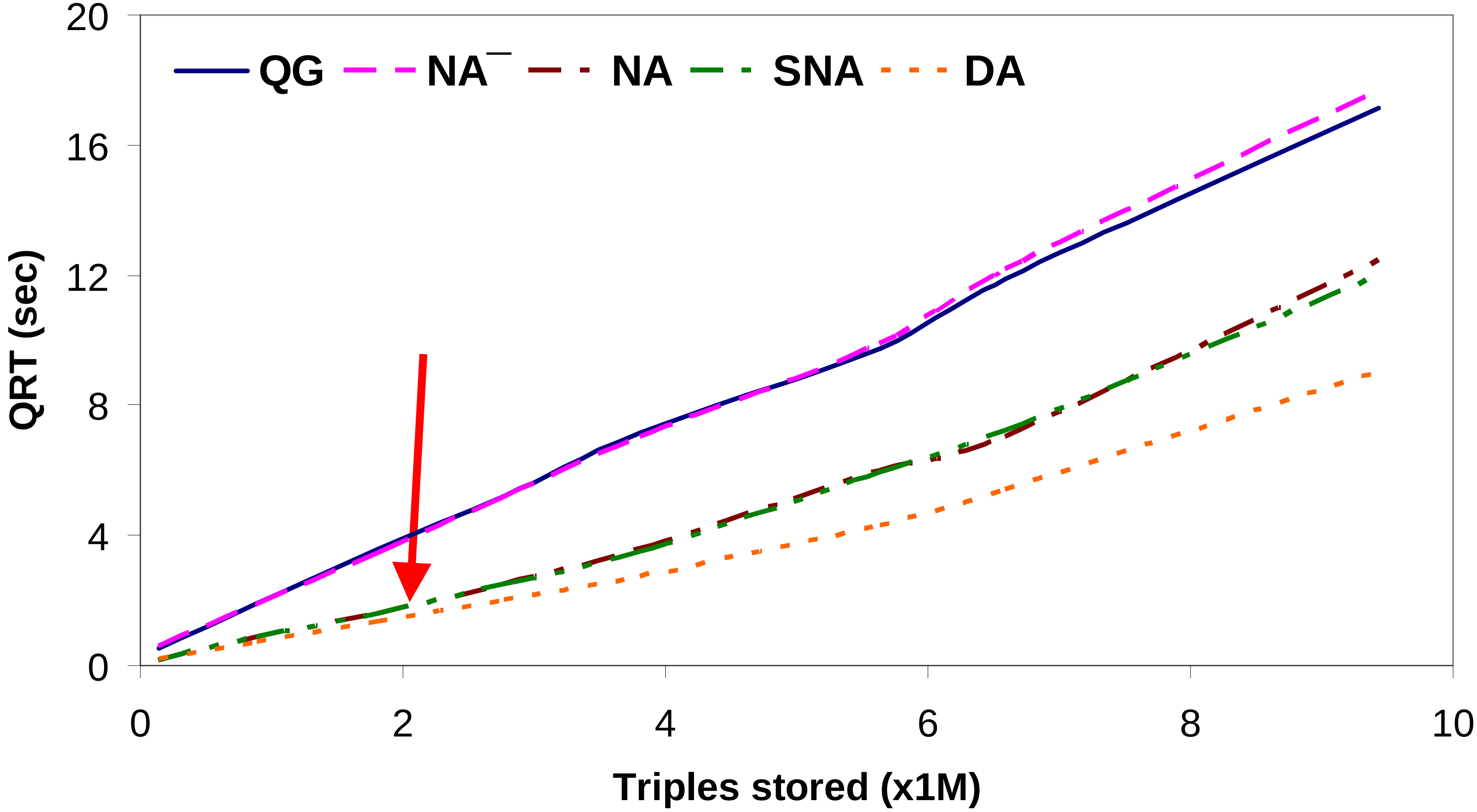
Q9: select ?X ?Y ?Z where {?X rdf:type ub:Student . ?Y rdf:type ub:Faculty . ?Z rdf:type ub:Course. ?X ub:advisor ?Y . ?X ub:takesCourse ?Z . ?Y ub:teacherOf ?Z . }

Q2 query plan space (LUBM-10)

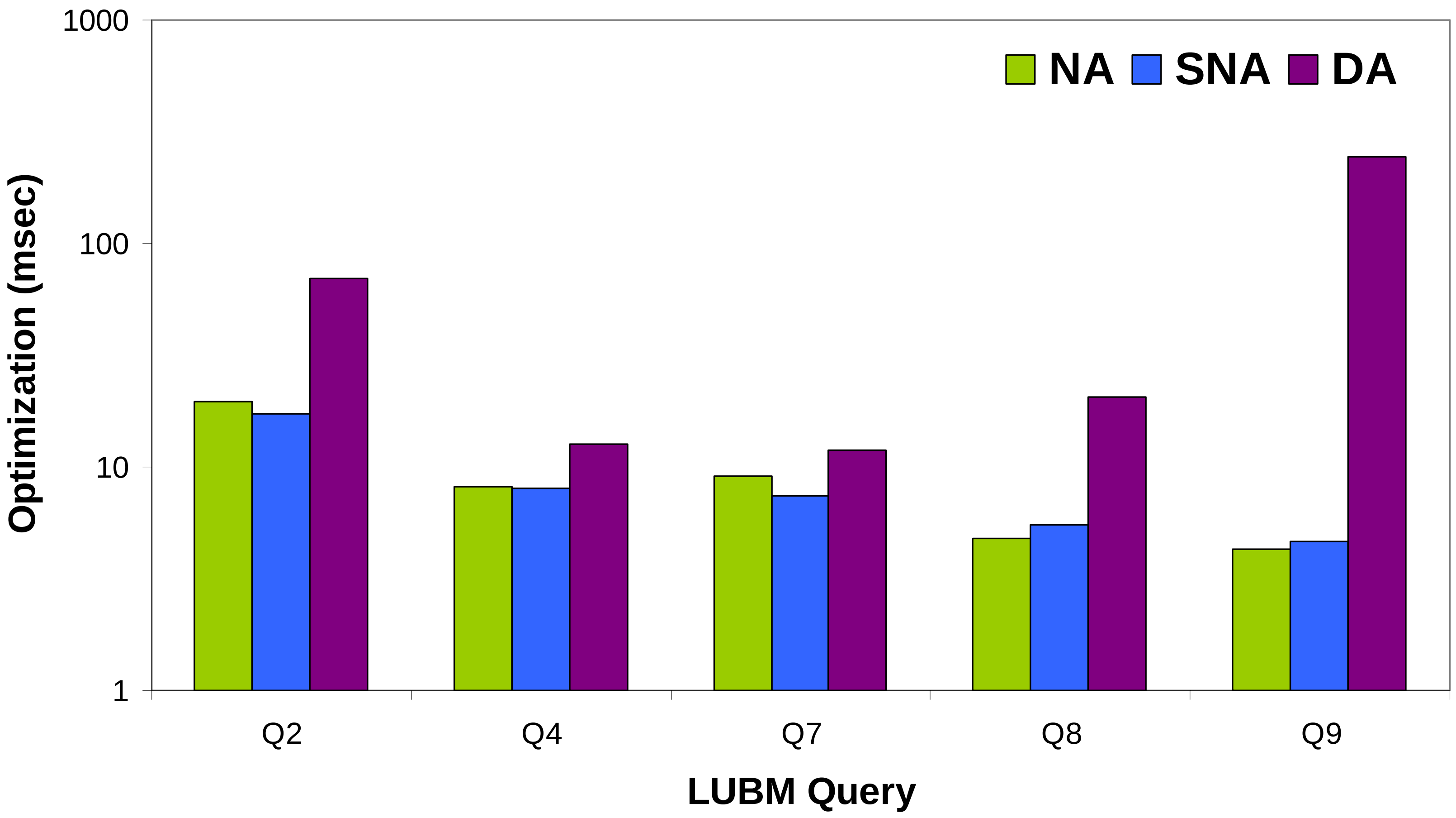


Q2: select ?X ?Y ?Z where { ?X rdf:type ub:GraduateStudent . ?X ub:undergraduateDegreeFrom ?Y . ?Y rdf:type ub:University . ?Z ub:subOrganizationOf ?Y . ?Z rdf:type ub:Department . ?X ub:memberOf ?Z . }

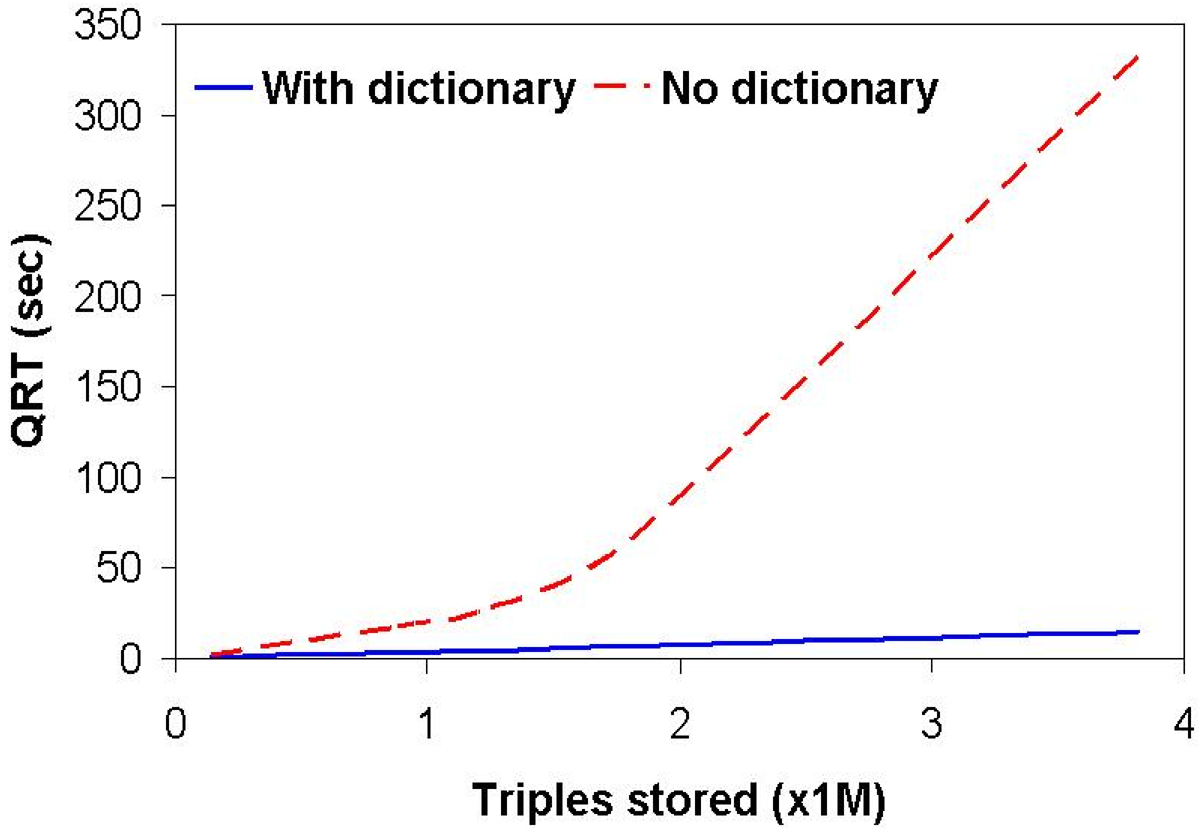
Query response time of Q2 for increasing dataset size



Optimization time (LUBM-50)



Mapping dictionary



Outline

- Introduction
- The system Atlas
- RDFS Reasoning in P2P networks
- SPARQL Query Processing and Optimization
- **Conclusions**

Conclusions

- **Atlas:** a P2P system for the distributed query processing and reasoning of RDF and RDFS data which is built on top of DHTs

- RDFS Reasoning on top of DHTs
  - Distributed forward chaining
  - Distributed backward chaining
  - Distributed magic sets transformation
  - Comparative study (analytically and experimentally)
  - Theoretical proofs of correctness

- SPARQL Query Processing and Optimization on top of DHTs
  - Selectivity estimation techniques
  - Statistics for RDF
  - Optimization algorithms
  - Distributed mapping dictionary

Future directions

- Distributed recursive queries using Datalog
- **RDF in the cloud!**

Thank you

Questions?

References 

Z. Kaoudi, K. Kyzirakos and M. Koubarakis. SPARQL Query Optimization on Top of DHTs. In *9th International Semantic Web Conference (ISWC 2010)*, Shanghai, China, November 7-11, 2010

Z. Kaoudi, M. Koubarakis, K. Kyzirakos, I. Miliaraki, M. Magiridou and A. Papadakis-Pesaresi. Atlas: Storing, Updating and Querying RDF(S) Data on Top of DHTs. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 2010

Z. Kaoudi, I. Miliaraki and M. Koubarakis**.** RDFS Reasoning and Query Answering on Top of DHTs. In *7th International Semantic Web Conference (ISWC 2008)*, Karlsruhe, Germany, October 26-30, 2008

Z. Kaoudi and M. Koubarakis. Distributed RDFS Reasoning over Structured Overlay Networks. Extended version of ISWC 2008 submitted to *ACM TWEB*