An Experimental Comparison of RDF Data Management Approaches in a SPARQL Benchmark Scenario

7th International Semantic Web Conference
Karlsruhe, Germany
October 28, 2008

Speaker: *Michael Schmidt*
joint work with T. Hornung, N. Küchlin, G. Lausen, and C. Pinkel

Motivation

- Efficient evaluation of SPARQL is a non-trivial task
  - SPARQL evaluation is `PSPACE`-complete
  - Homogeneous data format poses potential for severe bottlenecks (as we will discuss later)
  - Several optimization approaches have been made, but use their own, user-defined experimental setting for verification

Contributions

Part I

SPARQL Performance Benchmark *SP²Bench*

- Data Generator + Benchmark Queries
- Queries pose various challenges to SPARQL engines
- Allows us to compare optimization approaches
- Available online at

http://dbis.informatik.uni-freiburg/index.php?project=SP2B

Contributions

Part II

- Evaluation of existing RDF management approaches
  - Focus on translations into relational context
  - Comparison to native engine, relational setting

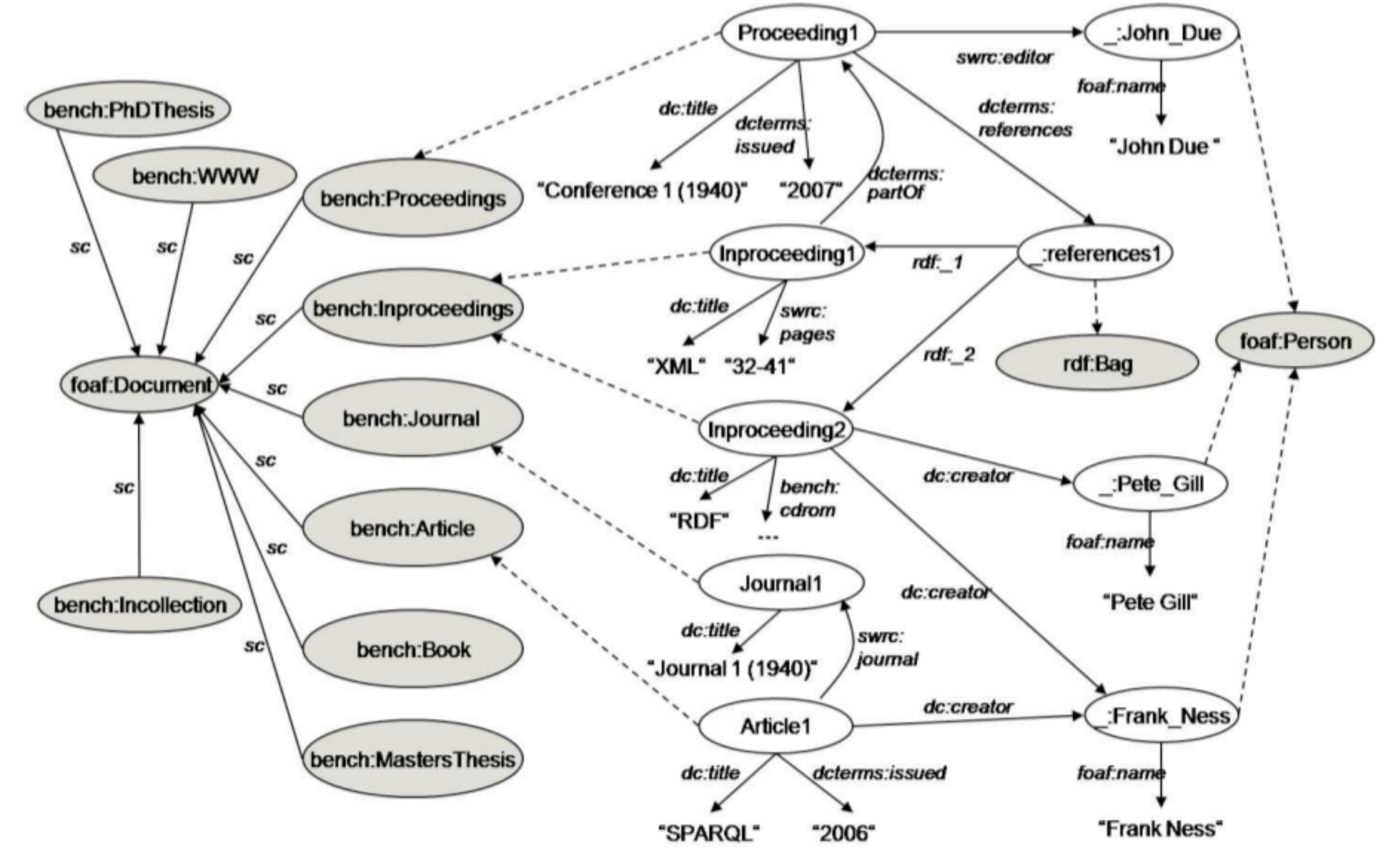
Part III

- Several new findings
  - Limitations of existing evaluation approaches
  - Severe gap to native relational data processing

SP²Bench Scenario

- Domain: DBLP bibliographic data
- Contains bilbliographic entities such as articles, journals, proceedings, inproceedings…
- DBLP fits „RDF philosophy“
  - RDF designed for representing meta data
  - Many social-world distributions found in DBLP

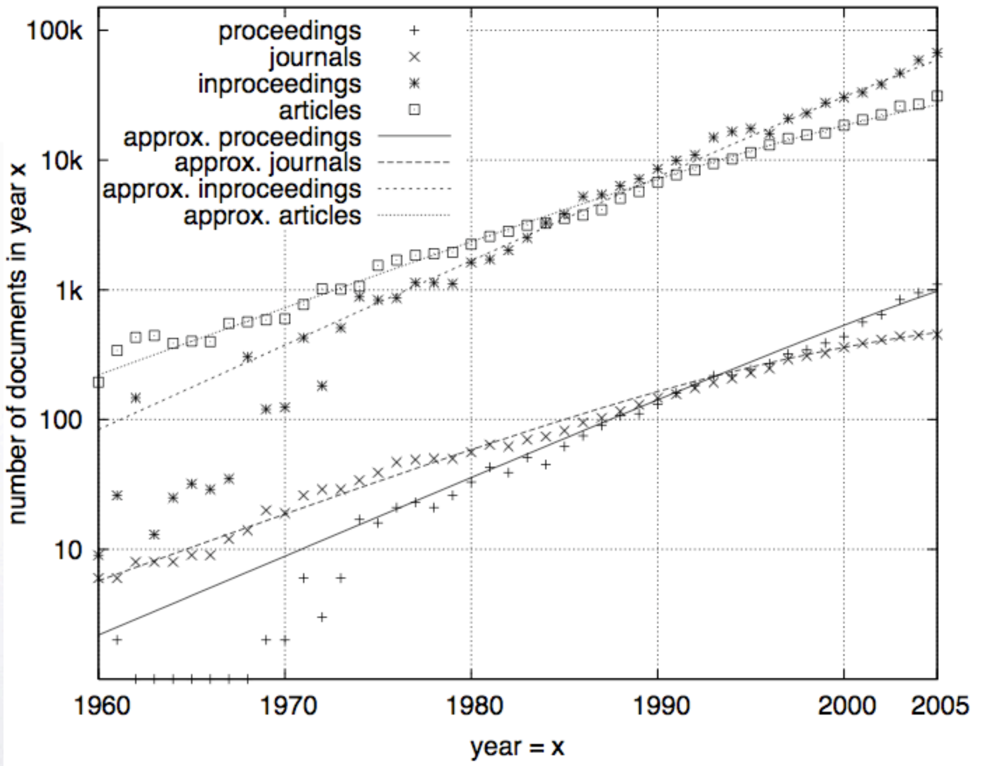
M. Schmidt, T. Hornung, G. Lausen, C. Pinkel. SP²Bench - A SPARQL Performance Benchmark. In ICDE'09.
Ley, M.: DBLP Database. http://www.informatik.uni- trier.de/~ ley/db/.


Proceeding1
_:John_Due
swrc:editor
foaf:name
"John Due"
dc:title
dcterms:issued
dcterms:references
"Conference 1 (1940)"
"2007"
dcterms:partOf
bench:PhDThesis
bench:WWW
bench:Proceedings
Inproceeding1
_:references1
rdf:_1
bench:Inproceedings
swrc:pages
"XML"
"32-41"
rdf:_2
rdf:Bag
foaf:Person
foaf:Document
bench:Journal
Inproceeding2
dc:creator
_:Pete_Gill
"RDF"
bench:cdrom
bench:Article
"Pete Gill"
bench:Incollection
Journal1
swrc:journal
bench:Book
"Journal 1 (1940)"
Article1
_:Frank_Ness
bench:MastersThesis
"SPARQL"
"2006"
"Frank Ness"
sc

Data with Real-world Characteristics

#instances per year for each document type

real DBLP
vs.
approx. in SP²Bench



Data with Real-world Characteristics

- Other characteristics that we consider
  - Citation system
    - Incoming citations per publication (follows a power law distribution)
    - Outgoing citations per publication
  - Structure of documents
  - ...

SP²Bench SPARQL Queries

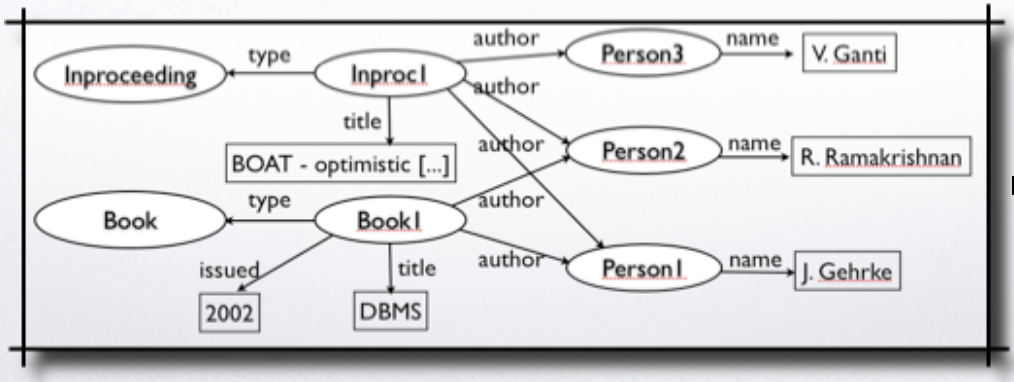
- Meaningful requests on top of the data
- Vary in a broad range of characteristics
  - Different operator constellation, RDF access patterns, and complexity
  - Result size (small, large, linear, ...)
  - Number of variables
  - ...

Storage Schemes for RDF

- Focus of this work: translation into relational context and evaluation of queries with conventional SQL database systems
- We consider two different approaches
  - Simple Triple Table Approach
  - Vertical Partitioning

Triple Table Approach

- Simple and straightforward storage scheme for RDF data
- All data stored in a single relation `Triples`(*subject*, *predicate*, *object*)



`Triples`

subject	*predicate*	*object*
Book1	type	Book
Book1	title	"DBMS"
Book1	issued	"2002"
Book1	author	Person1
Book1	author	Person2
Person1	name	"J. Gehrke"
...	...	...

Triple Table Approach

- Systematic SPARQL-to-SQL rewriting to evaluate SPARQL queries on top of the triples table

```
SELECT ?author
WHERE {
   ?book type Book.
   ?book author ?author.
}
```

"Select all book authors"

SPARQL-to-SQL translation (Triple Table) →

```
SELECT T2.object AS author
FROM Triples T1,
     Triples T2
WHERE
  T1.predicate="type" AND
  T1.object="Book" AND
  T2.predicate="author" AND
  T1.subject=T2.subject
```

Triple Table Approach

- Main disadvantage: resulting queries typically contain self-joins over table `Triples`

```
SELECT ?author
WHERE {
    ?book type Book.
    ?book author ?author.
}
```

"Select all book authors"

SPARQL-to-SQL translation (Triple Table)

```
SELECT T2.object AS author
FROM Triples T1,
     Triples T2,
WHERE
  T1.predicate="type" AND
  T1.object="Book" AND
  T2.predicate="author" AND
  T1.subject=T2.subject
```

Dictionary Encoding

Triples

subject	*predicate*	*object*
Book1	type	Book
Book1	title	"DBMS"
Book1	issued	"2002"
Book1	author	Person1
Book1	author	Person2
Person1	name	"J. Gehrke"
...	...	...

Dictionary encoding →

Triples

subject	*predicate*	*object*
1	2	3
1	4	5
1	6	7
1	8	9
1	8	10
9	11	12
...	...	...

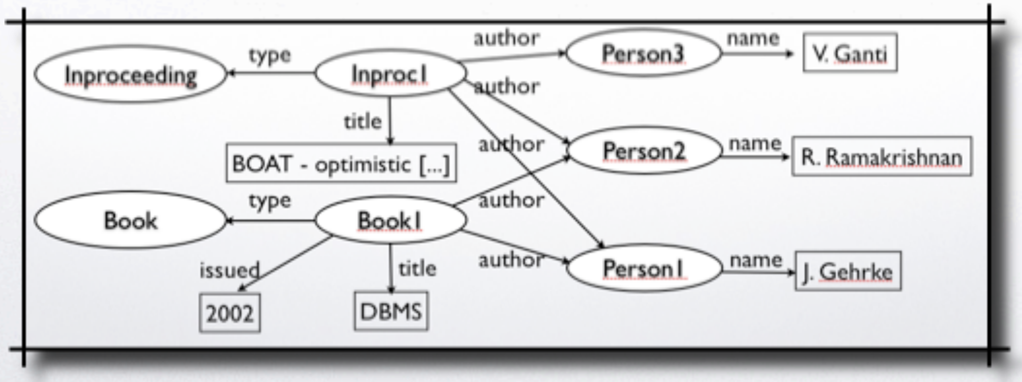
+

Dictionary

ID	*val*
1	Book1
2	type
3	Book
4	title
5	"DBMS"
6	issued
7	2002
8	author
9	Person1
10	Person2
11	name
12	J. Gehrke
...	...

Vertical Partitioning

- Set up one table for each distinct property (predicate) in the data
- Per table, store all tuples with the respective predicate



type

subject	*object*
Book1	Book
Inproc1	Inproceeding

author

subject	*object*
Book1	Person1
Book1	Person2
Inproc1	Person1
Inproc1	Person2
Inproc1	Person3

name

subject	*object*
Person1	"J. Gehrke"
Person2	"R. Ramakrishnan"
Person3	"V. Ganti"

...

Theoharis, Y., Christophides, V., Karvounarakis, G.: Benchmarking RDF Representations of RDF/S Stores. In ISWC'05.
D.J. Abadi et al.: Scalable Semantic Web Data Management Using Vertical Partitioning. In *VLDB'07*.

Vertical Partitioning

- Systematic SPARQL-to-SQL rewriting to evaluate SPARQL queries on top of the predicate tables, similar to the Triple Table approach

```
SELECT ?author
WHERE {
    ?book type Book.
    ?book author ?author.
}
```

"Select all book authors"

SPARQL-to-SQL translation (Vert. Part.)

```
SELECT au.object AS author
FROM type ty,
     author au
WHERE
   ty.object="Book" AND
   ty.subject=au.subject
```

Merge Joins (Vertical Partitioning)

```
SELECT au.object AS author
FROM type ty,
     author au
WHERE
  ty.object="Book" AND
  ty.subject=au.subject
```

"Select all book authors"

type

subject	*object*
Book1	Book
Book2	Book
Book3	Book
...	...

author

subject	*object*
Book1	Person1
Book1	Person2
Book2	Person2
Book3	Person4
Book3	Person5
Book3	Person6
...	...

D.J. Abadi et al.. Scalable Semantic Web Data Management Using Vertical Partitioning. In *VLDB*'07.

Merge Joins (Vertical Partitioning)

```
SELECT au.object AS author
FROM type ty,
     author au
WHERE
   ty.object="Book" AND
   ty.subject=au.subject
```

"Select all book authors"

type

subject	*object*
Book1	Book
Book2	Book
Book3	Book
...	...

author

subject	*object*
Book1	Person1
Book1	Person2
Book2	Person2
Book3	Person4
Book3	Person5
Book3	Person6
...	...

Efficient evaluation by merging subject columns when data physically sorted by (subject,object)!

D.J. Abadi et al.. Scalable Semantic Web Data Management Using Vertical Partitioning. In *VLDB*'07.

Merge Joins (Triple Table)

Finding: merge joins also possible in Triple Table scenario when physically sorting data by (predicate,subject,object)!

Vertical Partitioning

author

subject	*object*
Book1	Person1
Book1	Person2
Book2	Person2
Book3	Person4
Book3	Person5
Book3	Person6
...	...

type

subject	*object*
Book1	Book
Book2	Book
Book3	Book
...	...

Triple Table Approach

Triples

	subject	*predicate*	*object*
author-block	Book1	author	Person1
	Book1	author	Person2
	Book2	author	Person2
	Book3	author	Person4
	Book3	author	Person5
	Book3	author	Person6
	...	...	...
type-block	Book1	type	Book
	Book2	type	Book
	Book3	type	Book
	...	...	...

see also: *L. Sidirourgos, R. Gocalves, M. Kerstin, N. Nes, and S. Manegold*: Column-store Support for RDF Data Management: not all swans are white. In VLDB'08.

Experimental Setting

- Scenario ***TR***: Simple Triple Table approach
  - Data physically sorted by (*predicate*, *subject*, *object*)
  - Secondary index for remaining permutations of *subj.*, *pred.*, *obj.*
  - Combined with Dictionary Encoding
- Scenario ***VP***: Vertical Partitioning
  - Data physically sorted by *(subject, object)*
  - *Secondary Index for (object, subject)*
  - Combined with Dictionary Encoding

Experimental Setting

- Scenario ***SP***: Sesame native SPARQL engine
  - No RDF/SPARQL-to-SQL translation necessary
  - Provided Sesame all possible indices on RDF data
- Scenario ***RS***: Purely relational model of the scenario
  - Encoding designed using ERM DB modeling techniques
  - Using flat tables for publications, venues, persons, etc.
  - Queries: semantically equivalent SQL queries on top of the relational model

Settings Summary

- **TR**: Triple Table Approach
- **VP**: Vertical Partitioning
- **RS**: Purely Relational Schema

} MonetDB mserver v5.5.0, using the new algebra frontend

- **SP**: SPARQL Engine Sesame

} Sesame v2.0 coupled with its native SAIL

- Intel2 DuoCore 2.13GHz CPU, 3GB DDR2 RAM, Ubuntu v7.10 gutsy
- Generated Documents: 10k, 50k, 250k, 1M, 5M, and 25M triples
- 30min/query timeout, 2GB main memory limit, report on avg. over 3 runs

Experimental Results Q1

```
SELECT ?yr
WHERE {
  ?journal rdf:type bench:Journal.
  ?journal dc:title "Journal 1 (1940)".
  ?journal dcterms:issued ?yr
}
```

SPARQL (original benchmark query)

Return the year of publication of the journal with the title 'Journal 1 (1940)'.

```
SELECT T3.object AS yr
FROM type ty, title ti, issued is
WHERE ty.object="bench:Journal" AND
      ti.object="Journal 1 (1940)" AND
      ty.subject=ti.subject AND
      ti.subject=is.subject
```

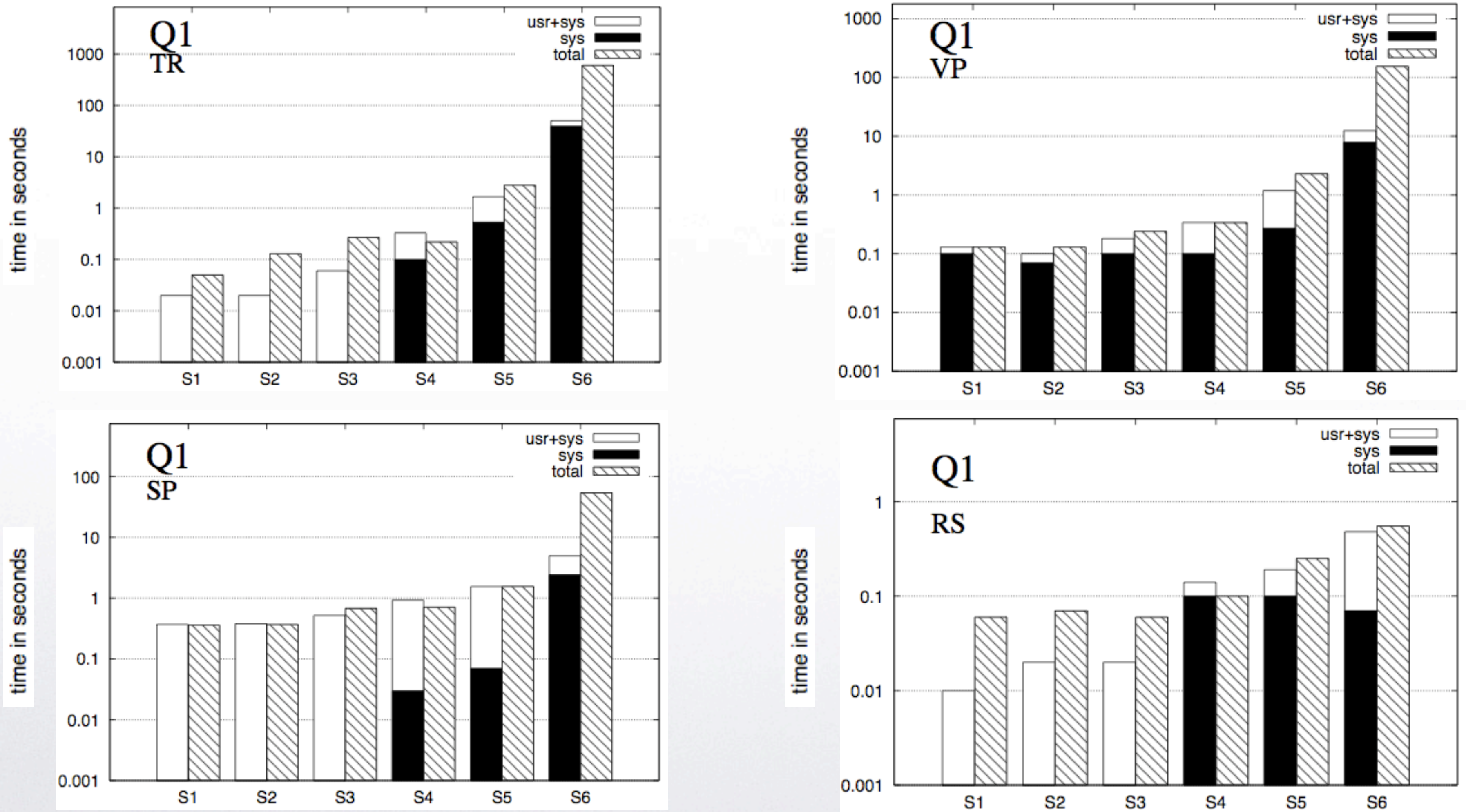
SQL/VP query without dictionary encoding (marginally modified)

```
SELECT T3.object AS yr
FROM Triples T1, Triples T2, Triples T3
WHERE T1.predicate="rdf:type" AND
      T1.object="bench:Journal" AND
      T2.predicate="dc:title" AND
      T2.object="Journal 1 (1940)" AND
      T3.predicate="dcterns:issued" AND
      T1.subject=T2.subject AND
      T1.subject=T3.subject
```

SQL/TR query without dictionary encoding (marginally modified)

All translations and SP²Bench data generator available online at
http://dbis.informatik.uni-freiburg/index.php?project=SP2B

#Triples: S1=10k / S2=50k / S3=250k / S4=1M / S5=5M / S6=25M



Experimental Results Q4

Select the names of all distinct pairs of article authors that have published in the same journal.

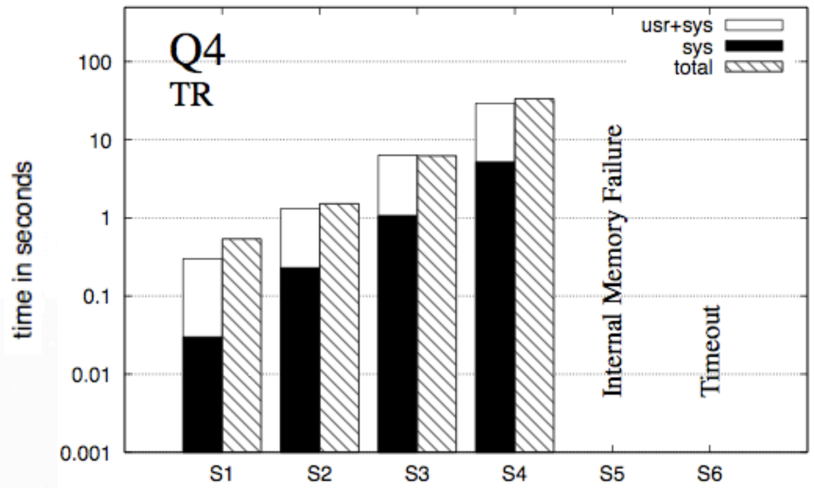
```
SELECT DISTINCT ?name1 ?name2
WHERE {
   ?article1 rdf:type bench:Article.
   ?article2 rdf:type bench:Article.
   ?article1 dc:creator ?author1.
   ?author1 foaf:name ?name1.
   ?article2 dc:creator ?author2.
   ?author2 foaf:name ?name2.
   ?article1 swrc:journal ?journal.
   ?article2 swrc:journal ?journal.
   FILTER (?name1<?name2)
}
```

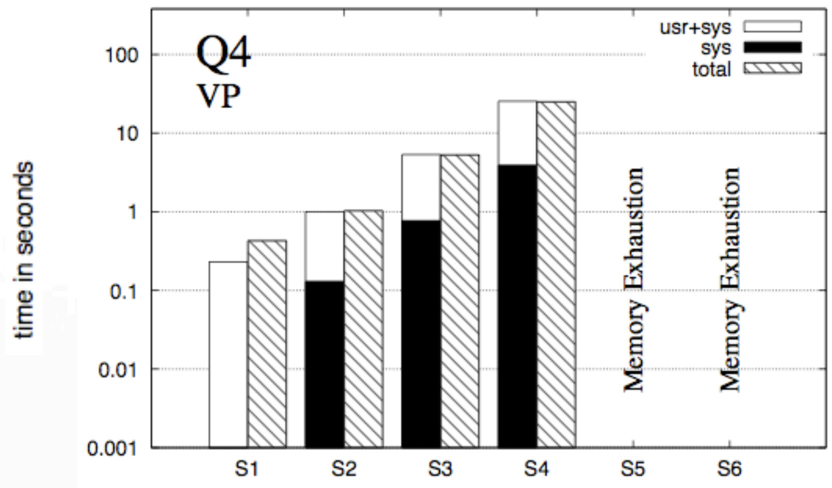
SPARQL (original benchmark query)

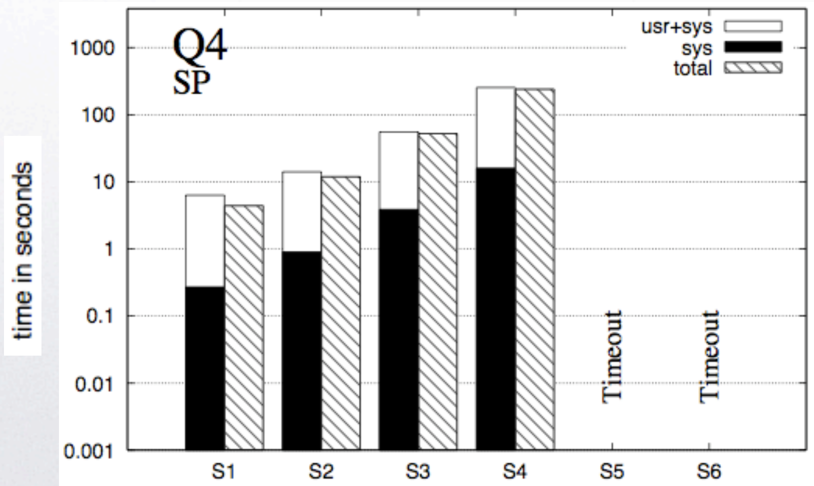
```
SELECT DISTINCT
  T4.object AS name1, T6.object AS name2
FROM Triples T1, Triples T2, ...,  Triples T8
WHERE
  T1.predicate="rdf:type" AND
  T1.object="bench:Article" AND
  T2.predicate="rdf:type" AND
  T2.object="bench:Article" AND
  T3.predicate="dc:creator" AND
  T4.predicate="foaf:name" AND
  T5.predicate="dc:creator" AND
  T6.predicat="foaf:name" AND
  T7.predicate="swrc:journal" AND
  T8.predicate="swrc:journal" AND
  T1.subject=T3.subject AND
  T1.subject=T7.subject AND
  T2.subject=T5.subject AND
  T2.subject=T8.subject AND
  T3.object=T4.subject AND
  T5.object=T6.subject AND
  T7.object=T8.object AND
  T4.object<T6.object
```

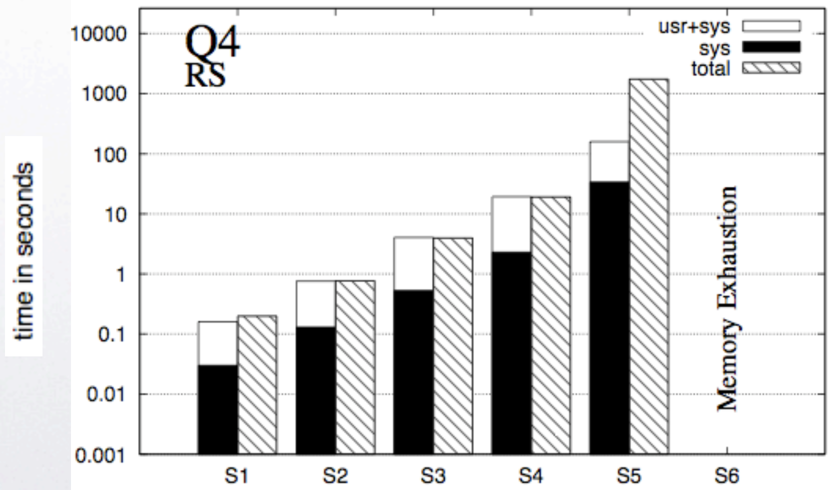
SQL/Triple Table without dictionary encoding (marginally modified)

#Triples: S1=10k / S2=50k / S3=250k / S4=1M / S5=5M / S6=25M









Experimental Results Q7

```
SELECT DISTINCT ?title
WHERE {
  ?class rdfs:subClassOf foaf:Document.
  ?doc rdf:type ?class.
  ?doc dc:title ?title.
  ?bag2 ?member2 ?doc.
  ?doc2 dcterms:references ?bag2
  OPTIONAL {
    ?class3 rdfs:subClassOf foaf:Document.
    ?doc3 rdf:type ?class3.
    ?doc3 dcterms:references ?bag3.
    ?bag3 ?member3 ?doc
    OPTIONAL {
      ?class4 rdfs:subClassOf foaf:Document.
      ?doc4 rdf:type ?class4.
      ?doc4 dcterms:references ?bag4.
      ?bag4 ?member4 ?doc3
    } FILTER (!bound(?doc4))
  } FILTER (!bound(?doc3))
}
```

Return the titles of all papers that have been cited at least once, but not by any paper without citations.

Encoded as:

Return the titles of all cited papers for which none of the citing papers is not cited.

SPARQL (original benchmark query)

Experimental Results Q7

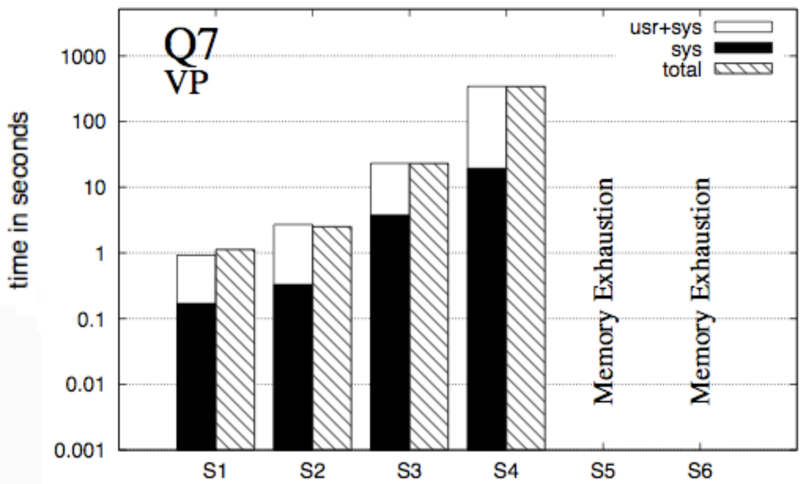
```
SELECT DISTINCT ?title
WHERE {
  ?class rdfs:subClassOf foaf:Document.
  ?doc rdf:type ?class.
  ?doc dc:title ?title.
  ?bag2 ?member2 ?doc.
  ?doc2 dcterms:references ?bag2
  OPTIONAL {
    ?class3 rdfs:subClassOf foaf:Document.
    ?doc3 rdf:type ?class3.
    ?doc3 dcterms:references ?bag3.
    ?bag3 ?member3 ?doc
    OPTIONAL {
      ?class4 rdfs:subClassOf foaf:Document.
      ?doc4 rdf:type ?class4.
      ?doc4 dcterms:references ?bag4.
      ?bag4 ?member4 ?doc3
    } FILTER (!bound(?doc4))
  } FILTER (!bound(?doc3))
}
```

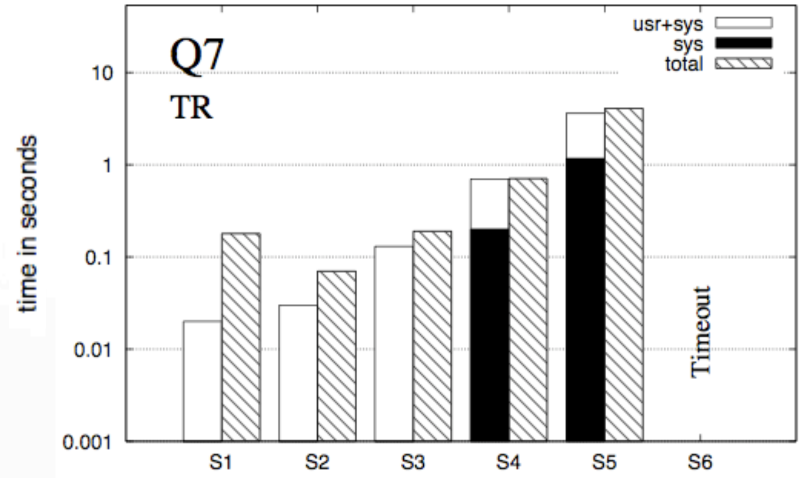
Return the titles of all papers that have been cited at least once, but not by any paper without citations.

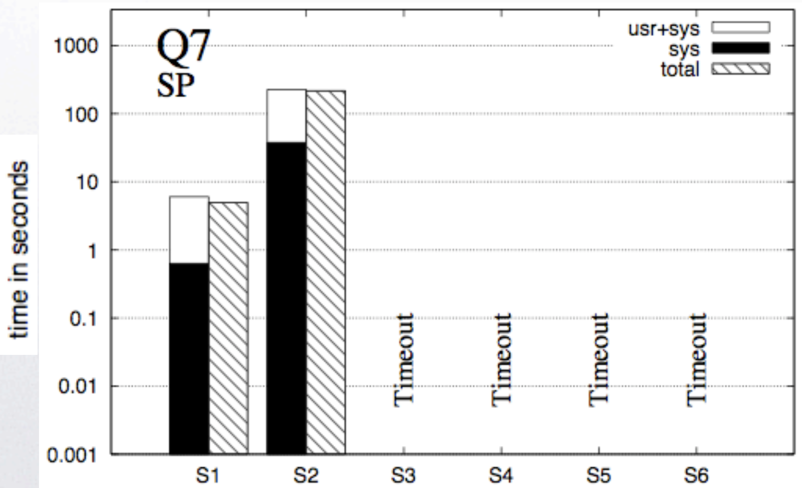
Problem when translating into VP:
Unbound predicates require large unions over all predicate tables; in contrast, query can be easily translated into TR scheme.

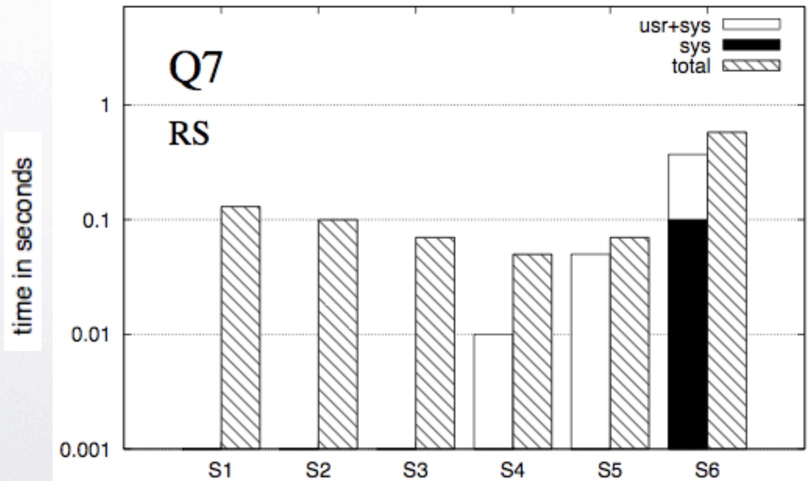
SPARQL (original benchmark query)

#Triples: S1=10k / S2=50k / S3=250k / S4=1M / S5=5M / S6=25M









Conclusion

- Optimizers of RDBMs often not laid out for the specific challenges that arise in the context of processing SW data
- Vertical Partitioning not a general solution: Limitations for queries with unbound predicates, non subject-subject joins, and in general more complex queries
- Triple Store with *(predicate,subject,object)* physical sort order often competitive to VP, since data is arranged in the same way on disk
- Typically gap of one order of magnitude compared to relational data processing yet on small documents, increasing with document size

New storage schemes and query evaluation approaches necessary, to bring forward the evaluation of SPARQL queries!

A promising approach: Cathrin Weiss, Panagiotis Karras, Abraham Bernstein: Hexastore: Sextuple Indexing for Semantic Web Data Management. In VLDB'08.

Thank you for your attention!

W3C: **Resource Description Framework (RDF)**. http://www.w3.org/RDF/.
W3C: **SPARQL Query Language**. http://www.w3.org/TR/rdf- sparql- query/.
Bizer, C., Cyganiak, R.: **D2R Server – Publishing the DBLP Bibliography Database**. http://www4.wiwiss.fu- berlin.de/dblp/.
Alexaki, S., Christophides, V., Karvounarakis, G., Plexousakis, D.: **On Storing Voluminous RDF Descriptions: The case of Web Portal Catalogs**. In WebDB. (2001)
Broekstra, J., Kampman, A., van Harmelen, F.: **Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema**. In ISWC. (2002)
Bonstrom, V., Hinze, A., Schweppe, H.: **Storing RDF as a Graph**. In Web Congress. (2003)
Theoharis, Y., Christophides, V., Karvounarakis, G.: **Benchmarking RDF Representations of RDF/S Stores**. In ISWC. (2005)
Chong, E.I., Das, S., Eadon, G., Srinivasan, J.: **An Efficient SQL-based RDF Querying Scheme**. In VLDB. (2005)
Wilkinson, K.: **Jena Property Table Implementation**. In International Workshop on SSWKB. (2006)
Abadi, D.J., Marcus, A., Madden, S., Hollenbach, K.J.**: Scalable Semantic Web Data Management Using Vertical Partitioning**. In VLDB. (2007)
Abadi, D.J., Marcus, A., Madden, S., Hollenbach, K.J.: **Using the Barton libraries dataset as an RDF benchmark**. TR, MIT.
Schmidt, M., Hornung, T., Lausen, G., Pinkel, C.: **SP^{2}Bench: A SPARQL Performance Benchmark**. In ICDE'09, to appear.
Ley, M.: **DBLP Database**. http://www.informatik.uni- trier.de/~ ley/db/.
openRDF.org: **Home of Sesame**. http://www.openrdf.org/documentation.jsp.
Sidirourgos, L., Goncalves, R., Kersten, M., Nes, N., Manegold, S.: **Column-store Support for RDF Data Management: not all swans are white**. In VLDB. (2008)
Bizer, C., Schultz, A.**: The Berlin SPARQL Benchmark**. http://www4.wiwiss. fu- berlin.de/bizer/BerlinSPARQLBenchmark/.
Stonebraker, M, et al.: **C-store: a Column-oriented DBMS**. In VLDB. (2005) 553–564
CWI Amsterdam: **MonetDB. http://monetdb.cwi.nl/**.
Chebotko, A., Lu, S., Yamil, H.M., Fotouhi, F.: **Semantics Preserving SPARQL- to-SQL Query Translation for Optional Graph Patterns**. Technical report, TR- DB-052006-CLJF. (2006).
Cyganiac, R.: **A Relational Algebra for SPARQL**. TR, HP Bristol.
Harris, S.: **SPARQL Query Processing with Conventional Relational Database Systems**. In SSWS. (2005)

Additional Resources

- Benchmark Requirements
- Data generator implementation
- Query characteristics summary
- Distribution of outgoing citations
- Triple table approach with physical *(subject, predicate, object)* sort order
- Purely relational scheme

Benchmark Requirements

- *Relevance:* test typical operations within the benchmark domain
- *Scalability*: support tests on different data sizes
- *Portability*: possible execution on different platforms, applicability to different systems
- *Understandability*: since otherwise, it will not be accepted in practice

J. Gray: The Benchmark Handbook for Database and Transaction Systems. Morgan Kaufmann, 1993.

Data Generator Implementation

- Technical challenges to data generator
  - Efficient generation of large data sets (scales linearly to document size, constant memory)
  - Deterministic (random functions with fixed seed)
  - Incremental data generation
  - Platform independent
  - Physical Database Size

Query Characteristics

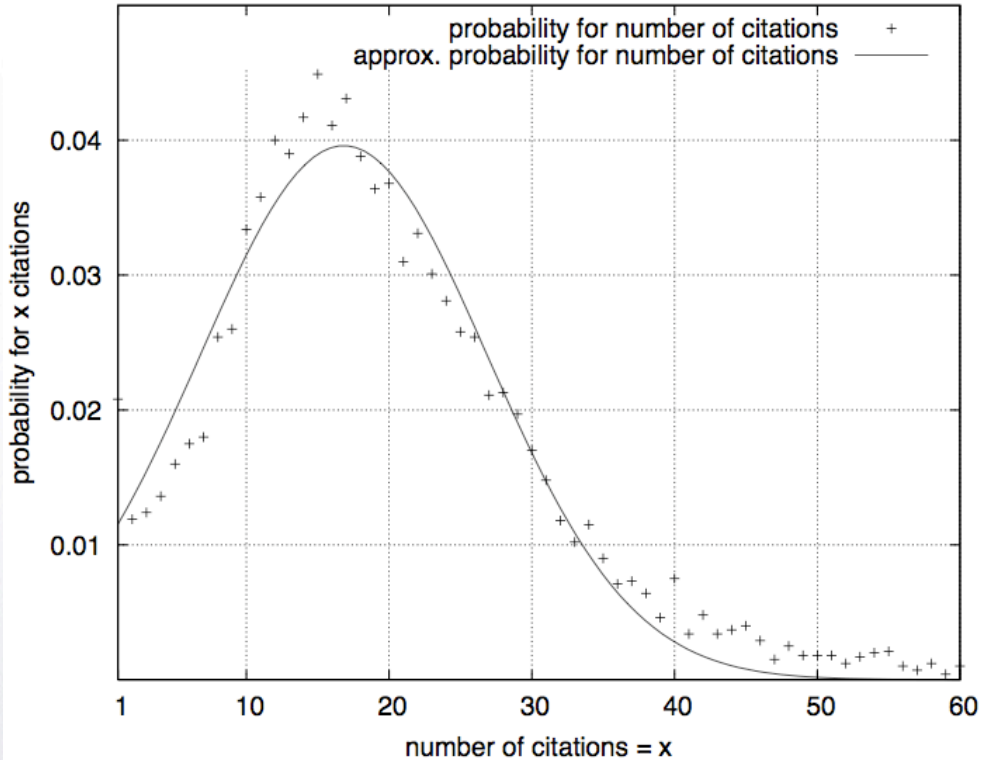
Category	*Construct*	*Q1*	*Q2*	*Q3*	*Q4*	*Q5*	*Q6*	*Q7*	*Q8*	*Q9*	*Q10*	*Q11*
Operators	And	✓	✓	✓	✓	✓	✓	✓	✓	✓		
	Union								✓	✓		
	Filter			✓	✓	✓	✓	✓	✓			
	Optional		✓				✓	✓				
Solution Modifiers	Distinct				✓	✓		✓	✓	✓		
	Limit											✓
	Offset											✓
	Order by		✓									✓
Data Access	Blank				✓	✓	✓		✓	✓		
	Literal	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓
	URI	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Relevance and Understandability

- Data with real-world characteristics

Probability for a paper having x citations

real DBLP
vs.
approx. in SP^2Bench



Merge Joins (Triple Table)

```
SELECT T2.object AS author
FROM Triples T1,
     Triples T2,
WHERE
  T1.predicate="type" AND
  T1.object="Book" AND
  T2.predicate="author" AND
  T1.subject=T2.subject
```

"Select all book authors"

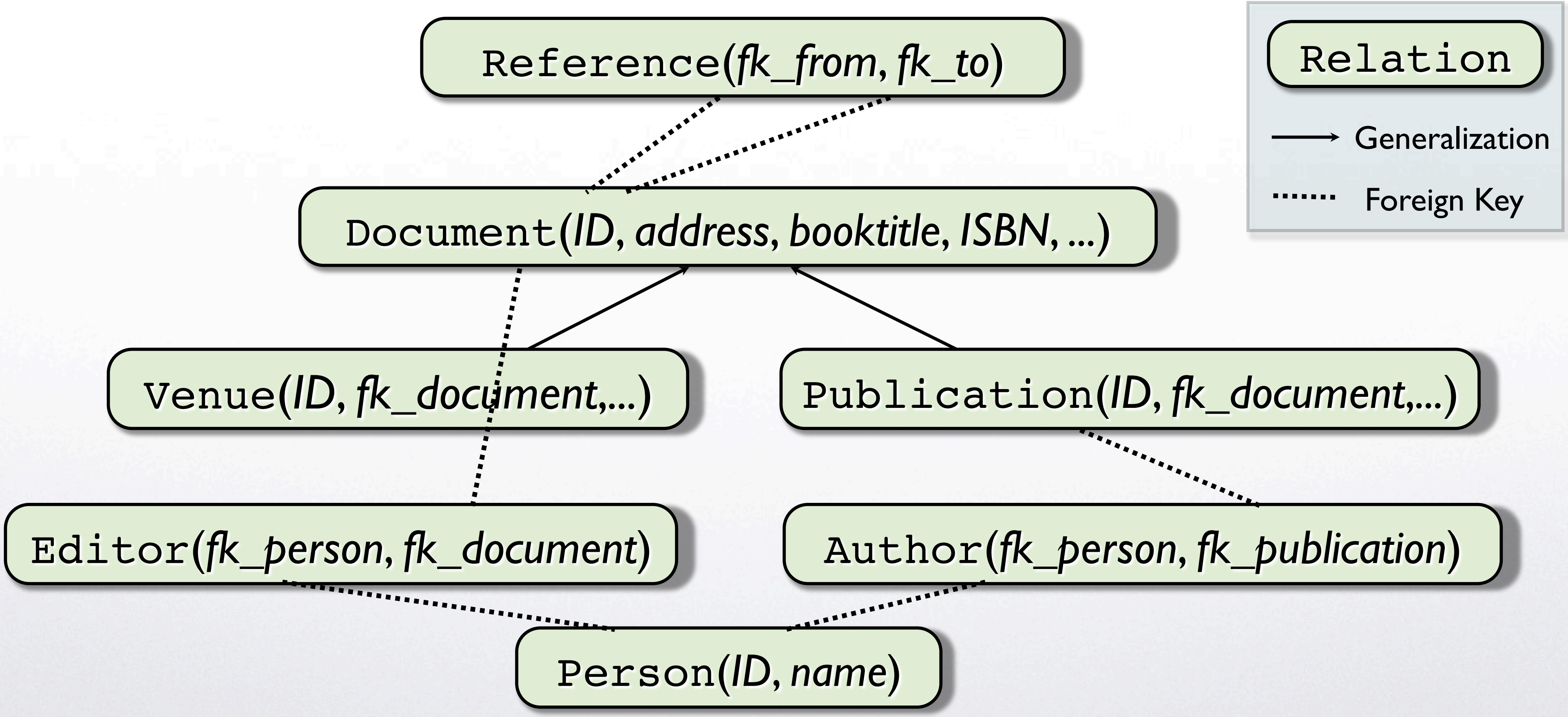
No efficient join evaluation possible when data is physically sorted by (subject,predicate,object)!

Triples

subject	predicate	object
Book1	author	Person1
Book1	author	Person2
...	...	...
Book1	type	Book
...	...	...
Book2	author	Person2
...	...	...
Book2	type	Book
...	...	...
Book3	author	Person4
Book3	author	Person5
Book3	author	Person6
...	...	...
Book3	type	Book

authors physically distributed

The Relational Scheme RS



Physical Database Size (incl. Indizes)

#triples in document	***SP***	***TR***	***VP***	***RS***
10k	3 MB	3 MB	6 MB	4 MB
50k	14 MB	5 MB	8 MB	5 MB
250k	69 MB	18 MB	20 MB	13 MB
1M	277 MB	63 MB	58 MB	42 MB
5M	1376 MB	404 MB	271 MB	195 MB
25M	6928 MB	2395 MB	1168 MB	913 MB