Generating **Efficient** Execution Plans for Vertically Partitioned XML Databases

Research paper review by

QING Pei, Edward	11500811g
LO Wing Yi, Wing	11523479g
SHAO Shuai, Philip	11552402g

April 10, 2012

What ?

Why ?

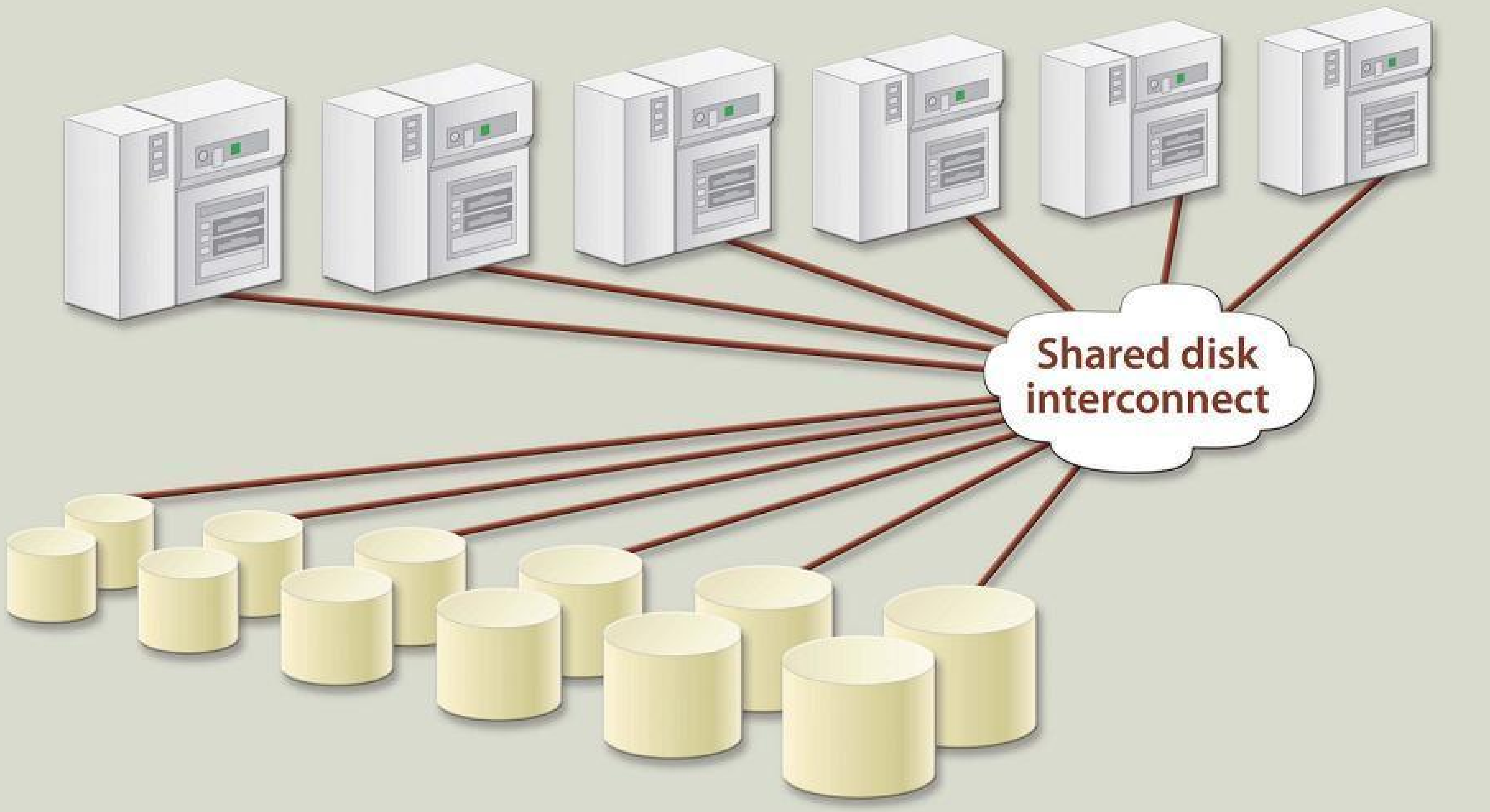
How ?

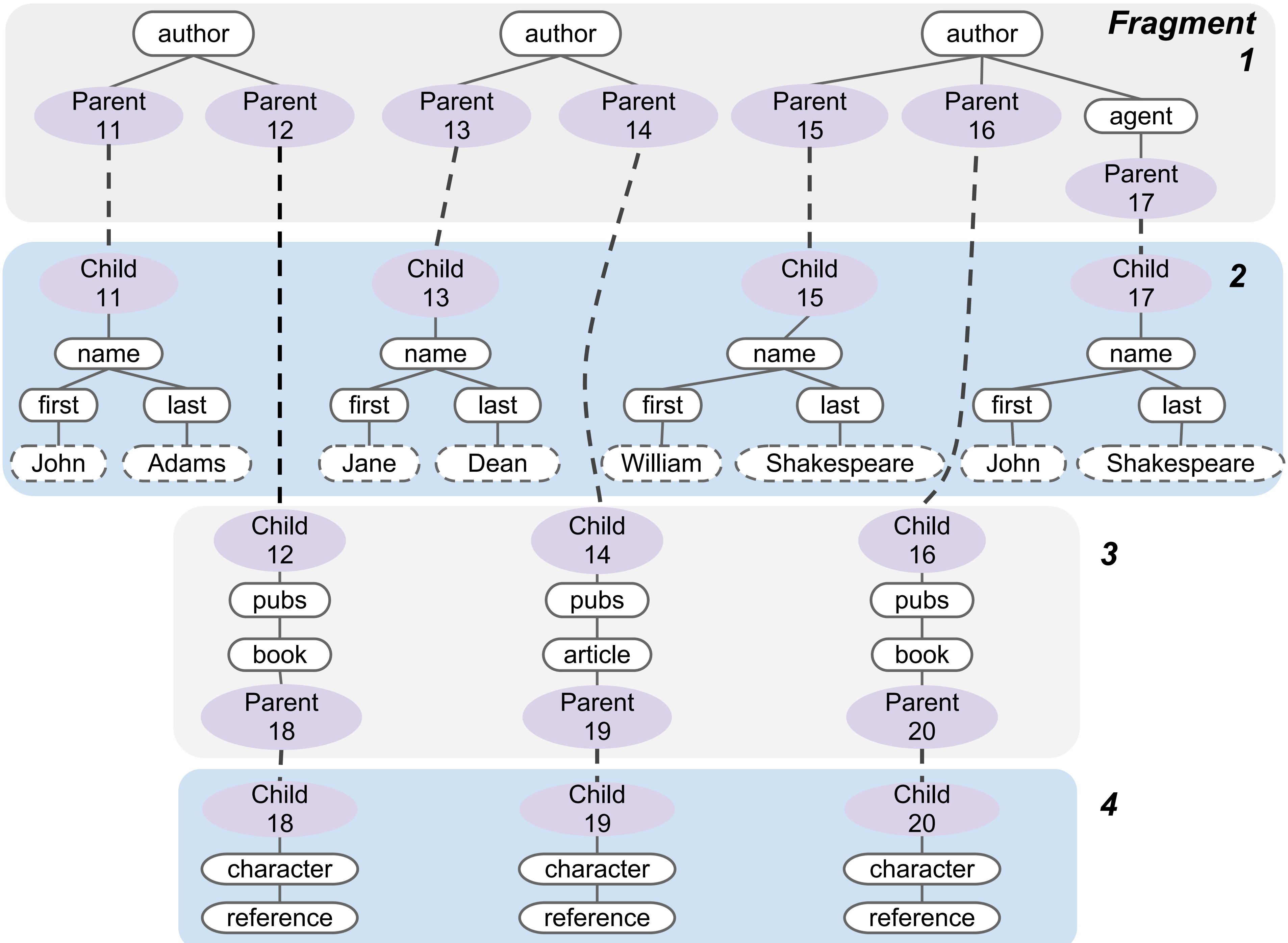
What ?

Query Processing

	Centralized	**Distributed**
RDBMS	✔	✔
XML	✔	*This paper*

XML in the Cloud





Fragment 1
author
author
author
Parent 11
Parent 12
Parent 13
Parent 14
Parent 15
Parent 16
agent
Parent 17
2
Child 11
name
first
last
John
Adams
Child 13
name
first
last
Jane
Dean
Child 15
name
first
last
William
Shakespeare
Child 17
name
first
last
John
Shakespeare
3
Child 12
pubs
book
Parent 18
Child 14
pubs
article
Parent 19
Child 16
pubs
book
Parent 20
4
Child 18
character
reference
Child 19
character
reference
Child 20
character
reference

Why ?

Distributed **architecture**
leads to
Different **execution plans**

For a single query, the **order** in which *joins* are performed results in various time consumed.

Response time

=

local execution time

+

joining time

local execution time

snip(i): the number of document subtrees accessed by the local plan at *fragment i*

smaller snip(i) preferred

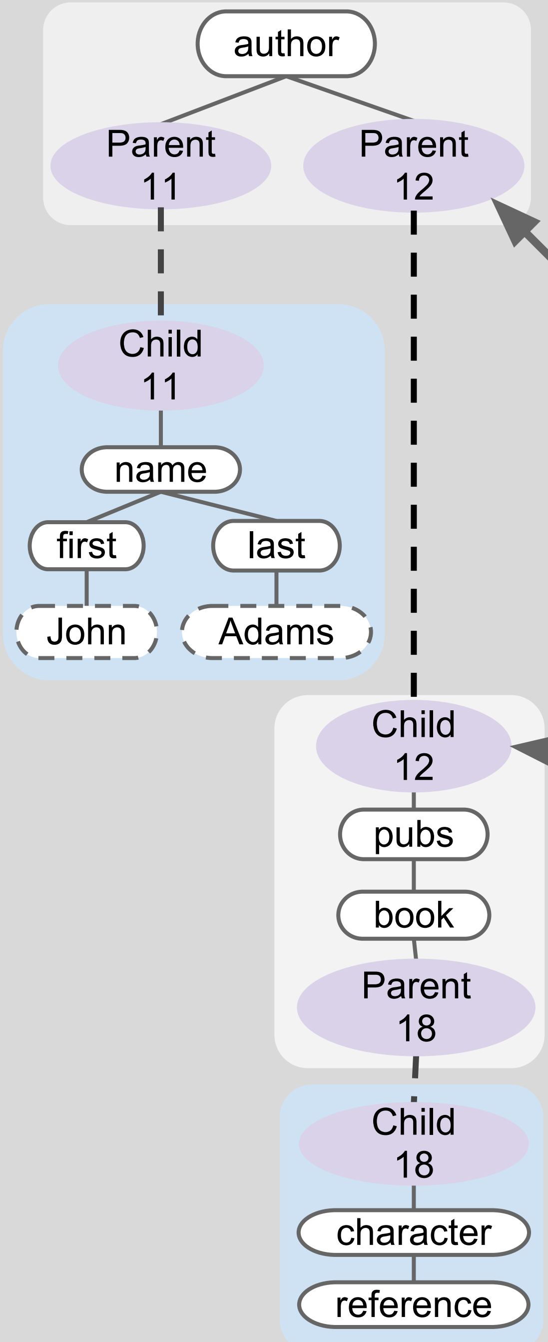
joining time

card(i): the number of tuples that are returned by the local plan when evaluated at *fragment i*

smaller card(i) preferred

Which **plan has the *minimum* response time?**

How ?



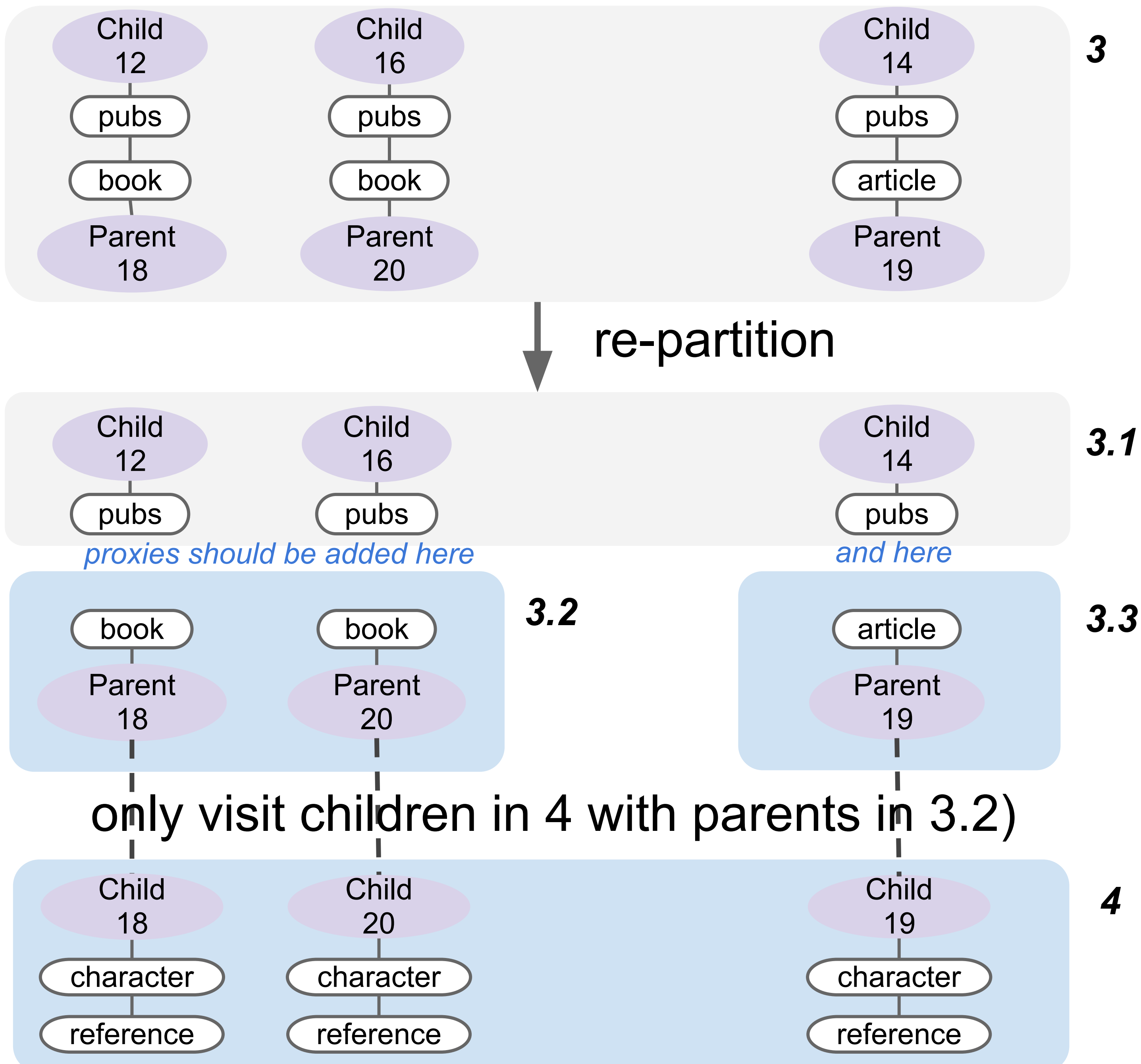
keep the relation between fragments

proxy nodes

Optimizing distributed plans

Pushing Cross-Fragment Joins

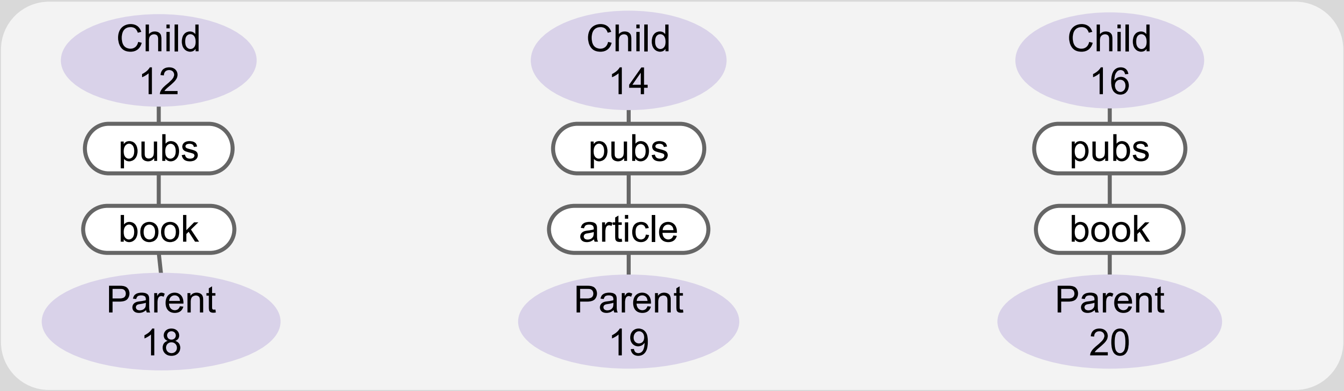
fully works on left-deep plans



Child 12
pubs
book
Parent 18
Child 16
pubs
book
Parent 20
Child 14
pubs
article
Parent 19
3
re-partition
Child 12
pubs
Child 16
pubs
Child 14
pubs
3.1
proxies should be added here
and here
book
Parent 18
book
Parent 20
3.2
article
Parent 19
3.3
only visit children in 4 with parents in 3.2)
Child 18
character
reference
Child 20
character
reference
Child 19
character
reference
4

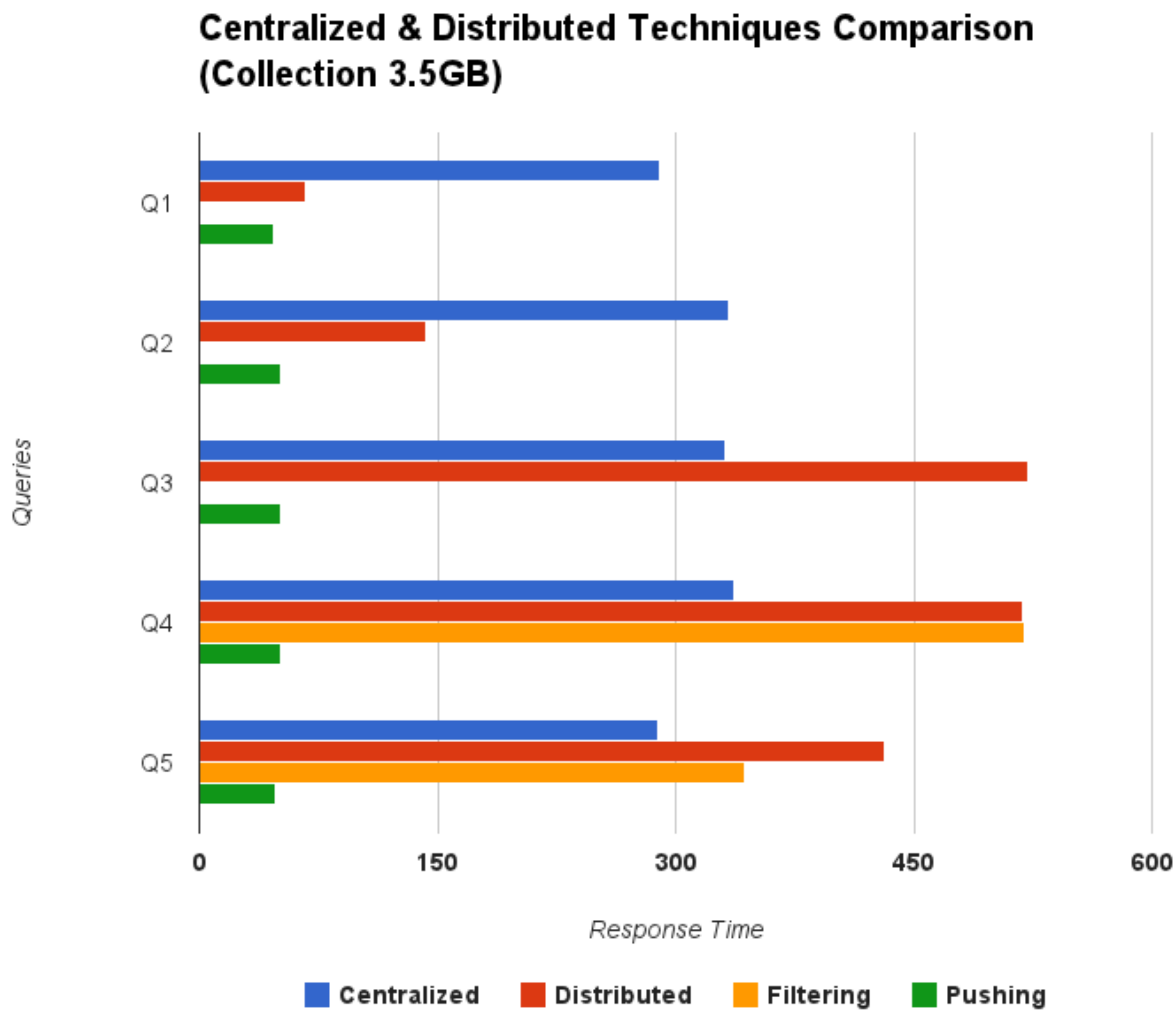
Label Path Filtering

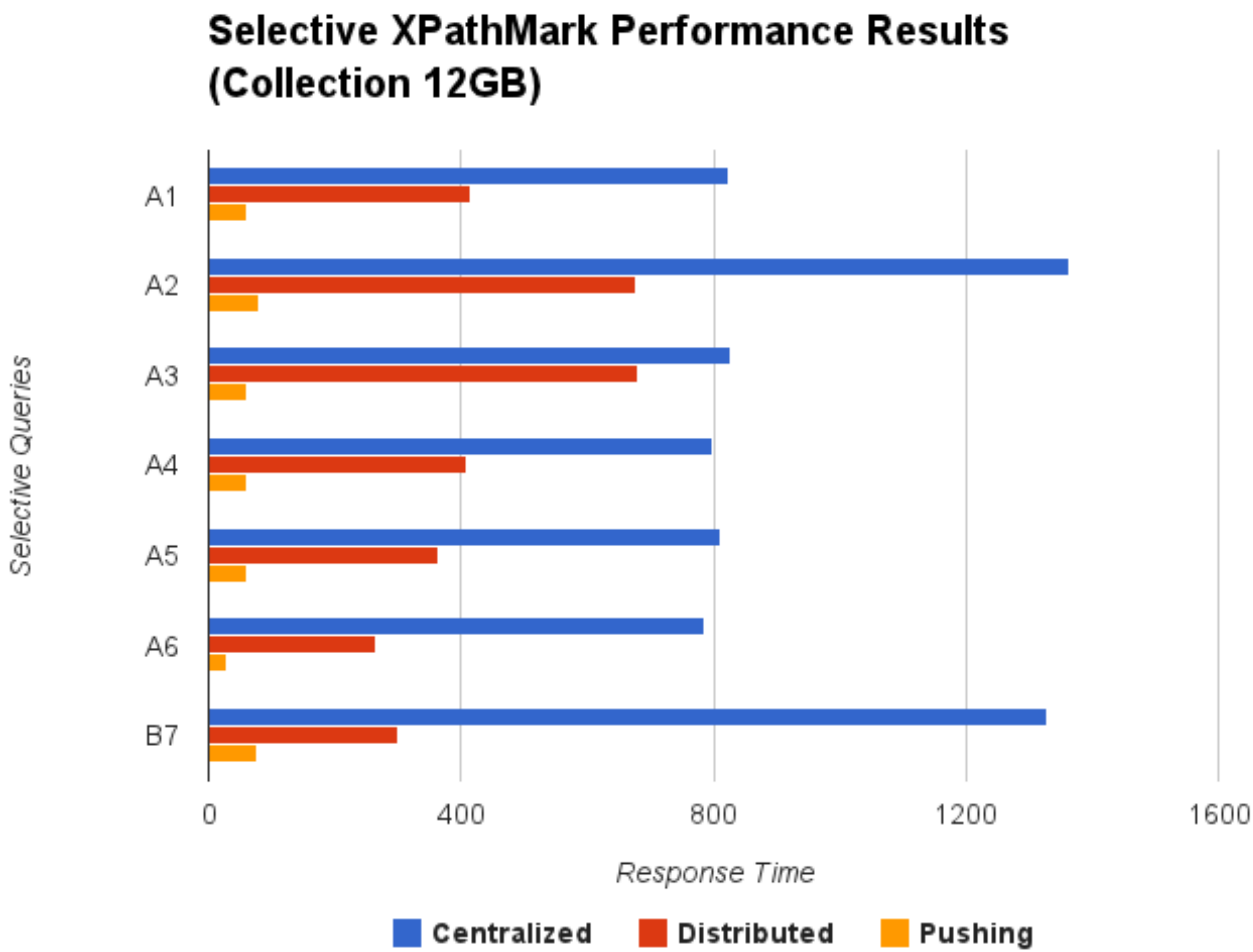
//book//reference



3

Evaluation


Centralized & Distributed Techniques Comparison
(Collection 3.5GB)
Queries
Q1
Q2
Q3
Q4
Q5
0
150
300
450
600
Response Time
Centralized
Distributed
Filtering
Pushing


Selective XPathMark Performance Results (Collection 12GB)
Selective Queries
A1
A2
A3
A4
A5
A6
B7
0
400
800
1200
1600
Response Time
Centralized
Distributed
Pushing

Conclusion

Greatly improves response time of querying large XML collections.

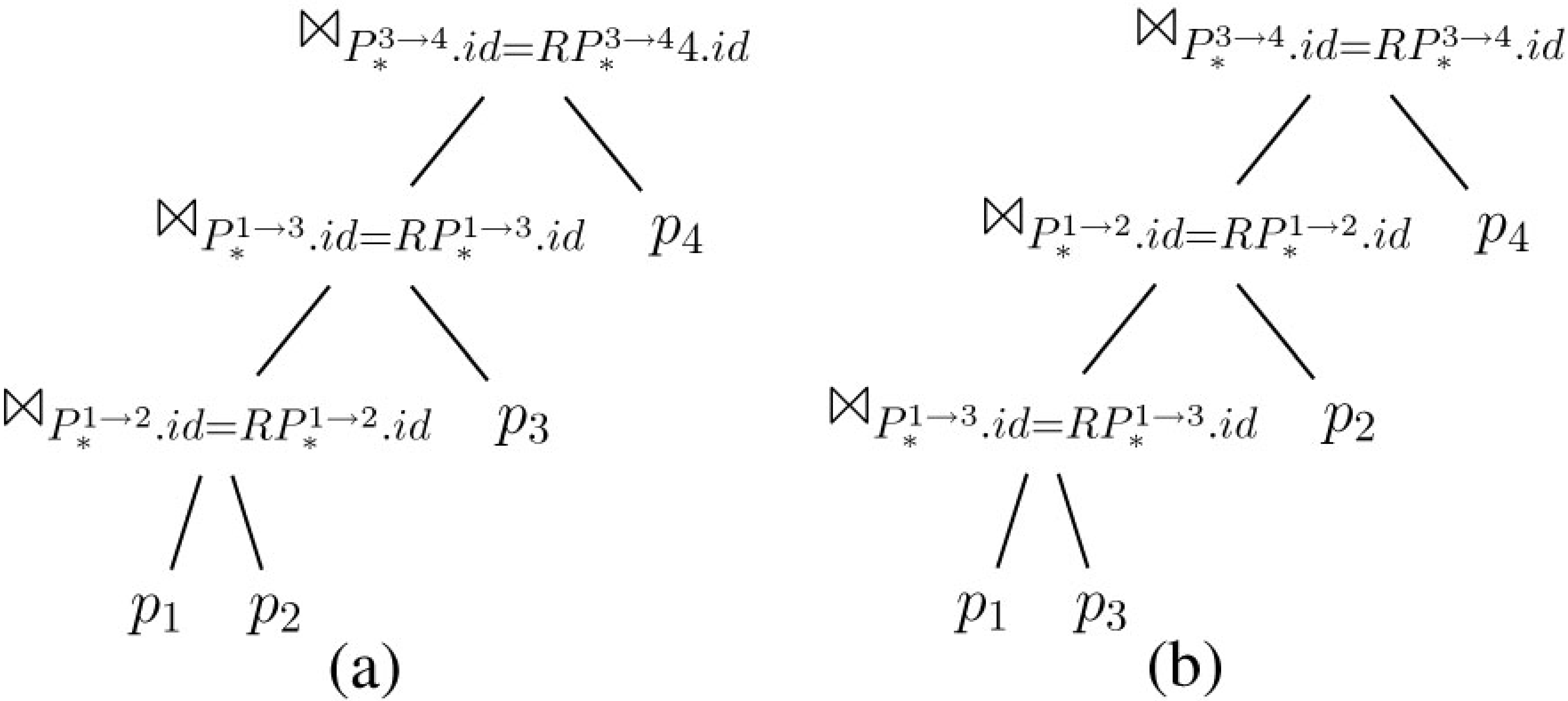
Small overhead. Choosing the fastest plan took < 0.01 seconds.

Q & A

Merci beaucoup

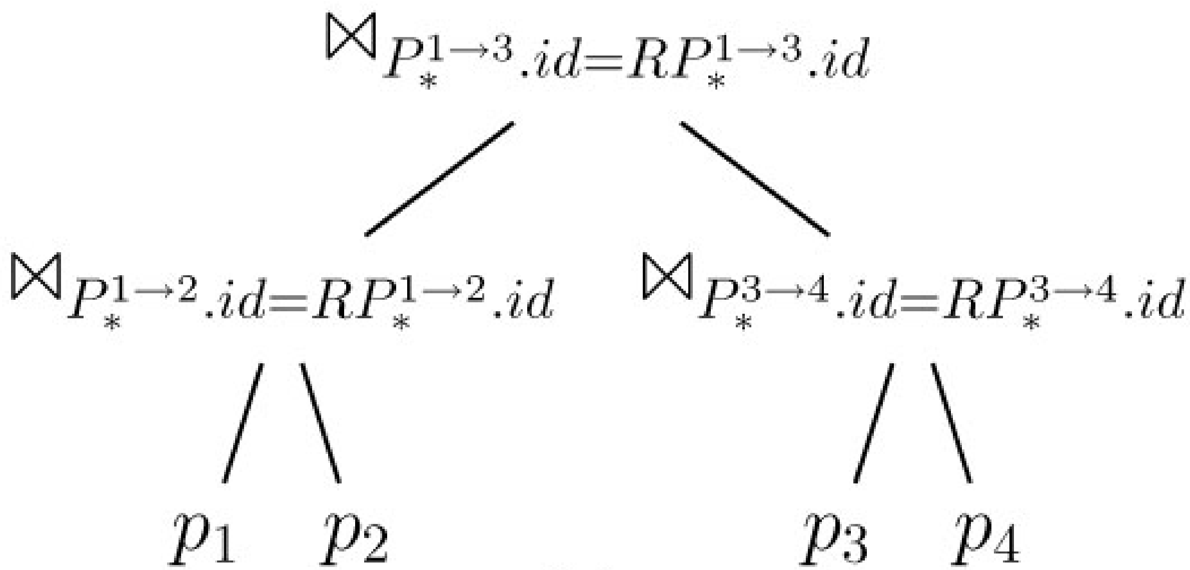
Appendix

Distributed Execution Plans



left-deep execution plans

Distributed Execution Plans



(c)

not a left-deep execution plan

Appendix

Queries used for evaluation

Q1 /open auction[initial > 200]//item//mail/from

Q2 /open auction[initial > 200][.//author/person/name[starts-with(., ’Ry’)]]//item//mail/from

Q3 /open auction[initial > 200][.//author/person/name[starts-with(., ’Ry’)]]//item//category/id

Q4 /open auction[initial > 200][.//author/person[profile/age > 30]/name[starts-with(., ’Ry’)]]//item//category/id

Q5 /open auction[initial > 200]//author/person[starts-with (name, ’Ry’)]/profile/interest/category/description

Appendix

Queries used for XPathMark

A1 /site/closed auctions/closed auction/annotation/description/text/keyword

A2 //closed auction//keyword

A3 /site/closed auctions/closed auction//keyword

A4 /site/closed auctions/closed auction [annotation/description/text/keyword]/date

A5 /site/closed auctions/closed auction[descendant:: keyword]/date

A6 /site/people/person[profile/gender and profile/age]/name

B7 //person[profile/@income]/name

Appendix

Queries used for Selective XPathMark

A1S /site/closed auctions/closed auction[price > 600]
/annotation/description/text/keyword

A2S //closed auction[price > 600]//keyword

A3S /site/closed auctions/closed auction[price > 600]
//keyword

A4S /site/closed auctions/closed auction[price > 600]
[annotation/description/text/keyword]/date

A5S /site/closed auctions/closed auction[price > 600]
[descendant::keyword]/date

A6S /site/people/person[starts-with(name, ’Ry’)]
[profile/gender and profile/age]/name

B7S //person[starts-with(name, ’Ry’)][profile/@income]/name

Appendix

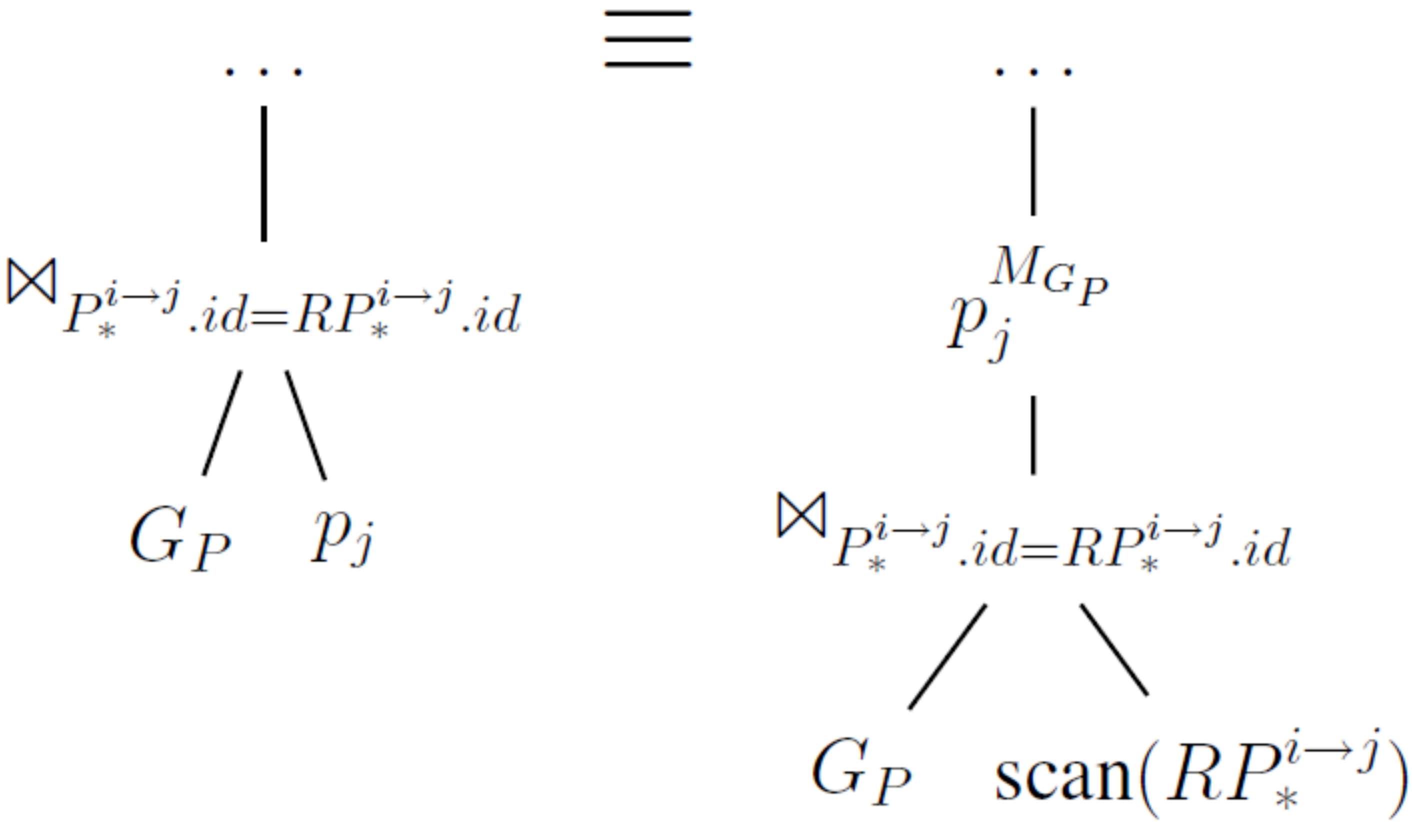


Figure 11: Cross-fragment join pushing rewrite

Appendix

$$p_j \equiv p'_j$$

$$\sigma_{RP_*^{i \rightarrow j}.label \in L_j}$$

$$\text{scan}(RP_*^{i \rightarrow j})$$

Figure 12: Label path rewrite

Appendix

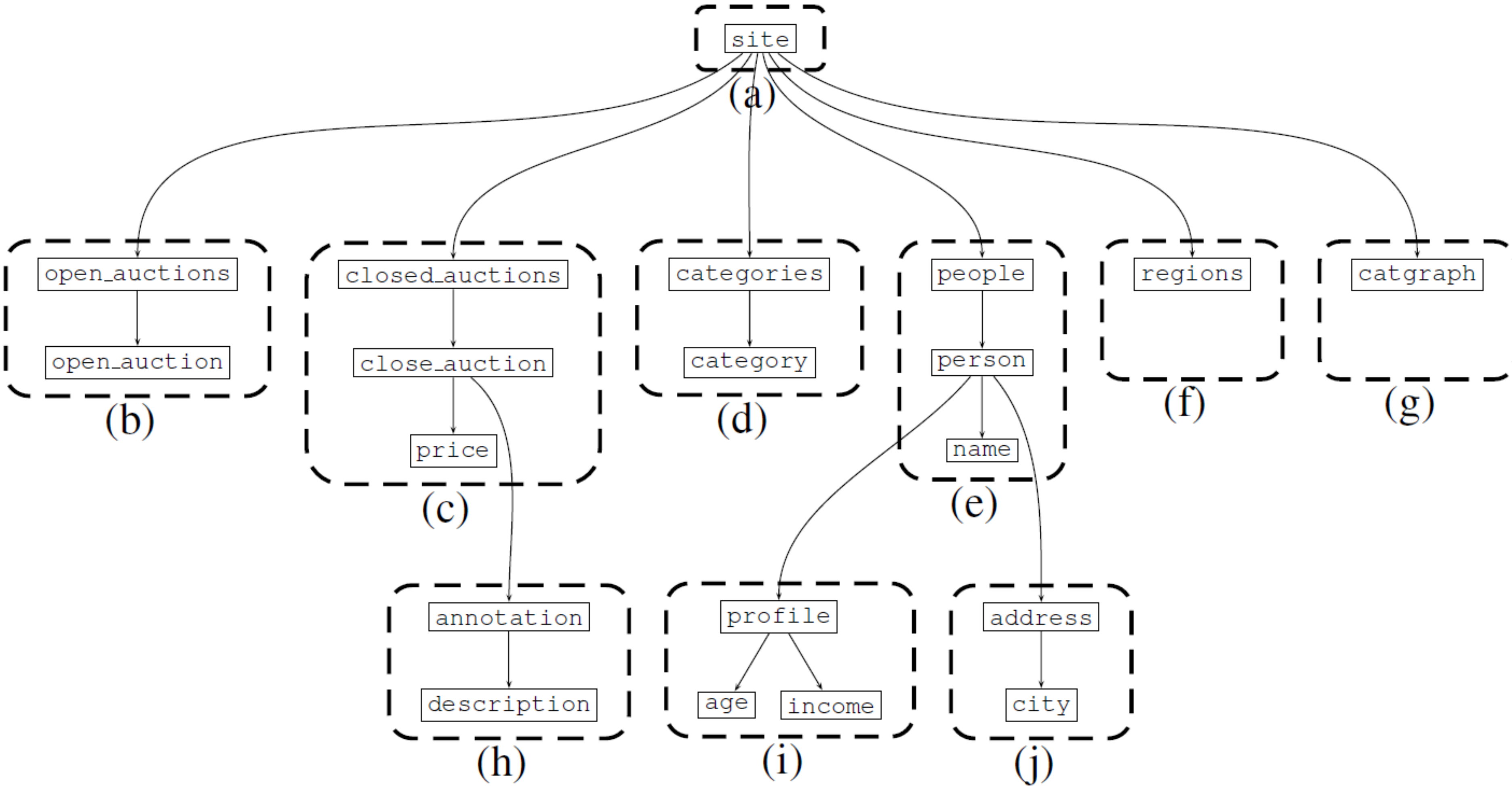


Figure 13: Fragmentation schema used in second experiment

Appendix

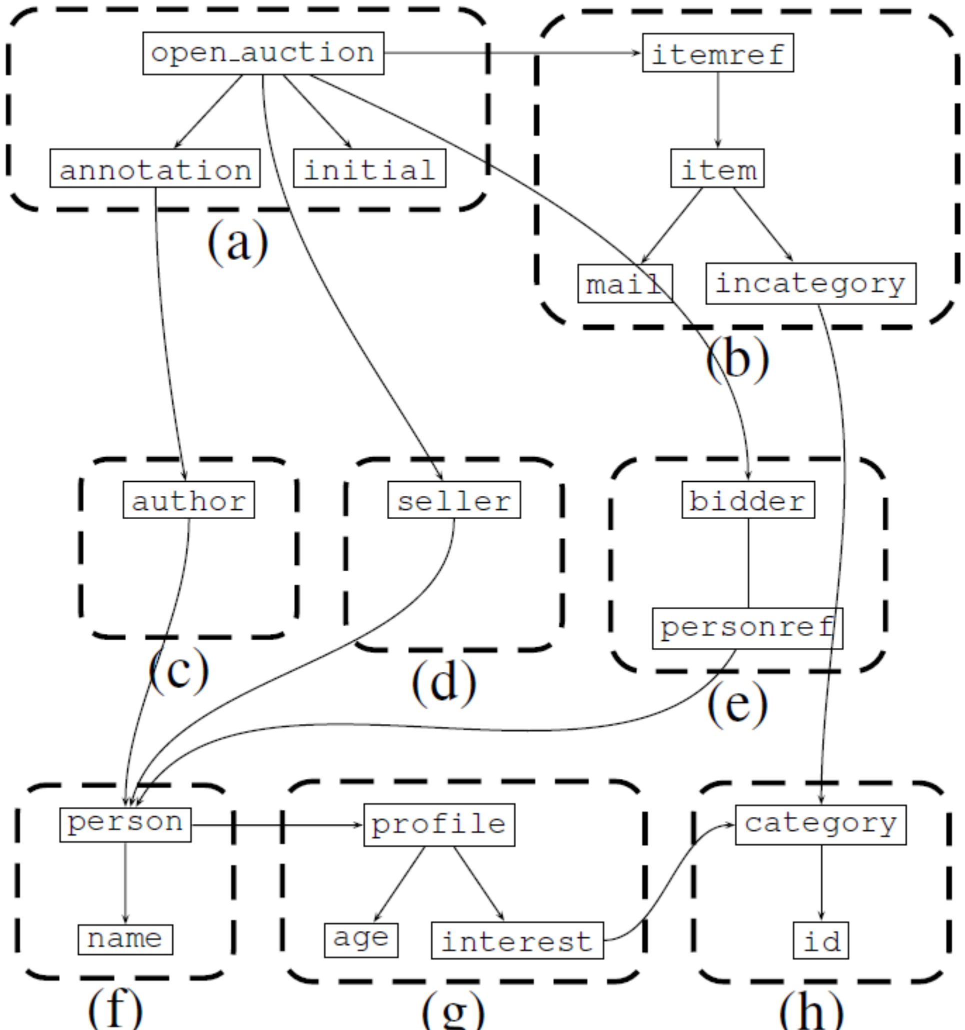


Figure 14: Fragmentation schema used in first experiment