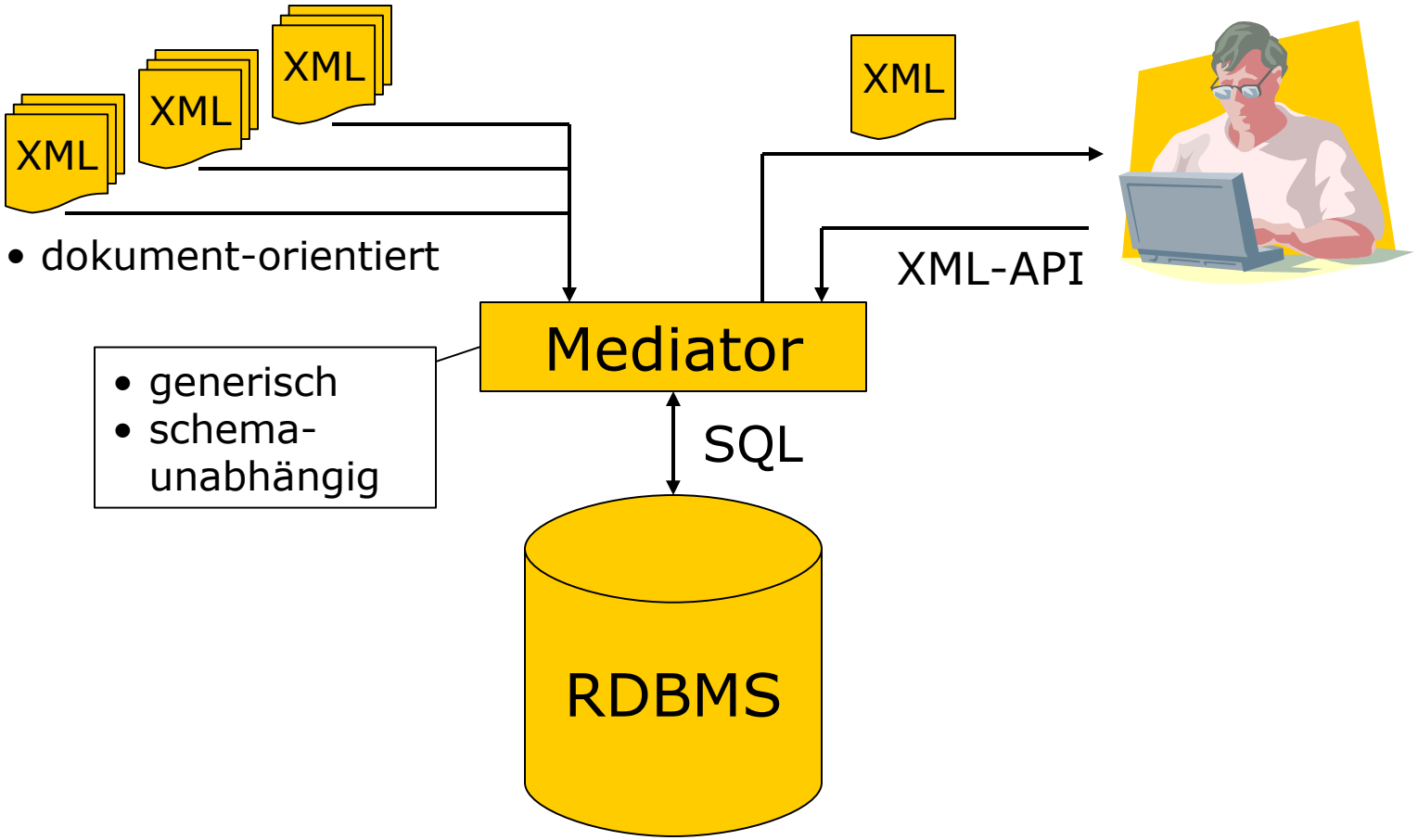
Optimierung von XPath-Anfragen in XMLRDB

Timo Böhme

Inhalt

- Motivation / Ziele
- XMLRDB
- Anfrageverarbeitung in XMLRDB
  - XPath-Implementierung
  - Optimierungen
  - Automatische Empfehlungen zur Optimierung (‚Wizard', ‚Advisor', ...)
- Evaluation

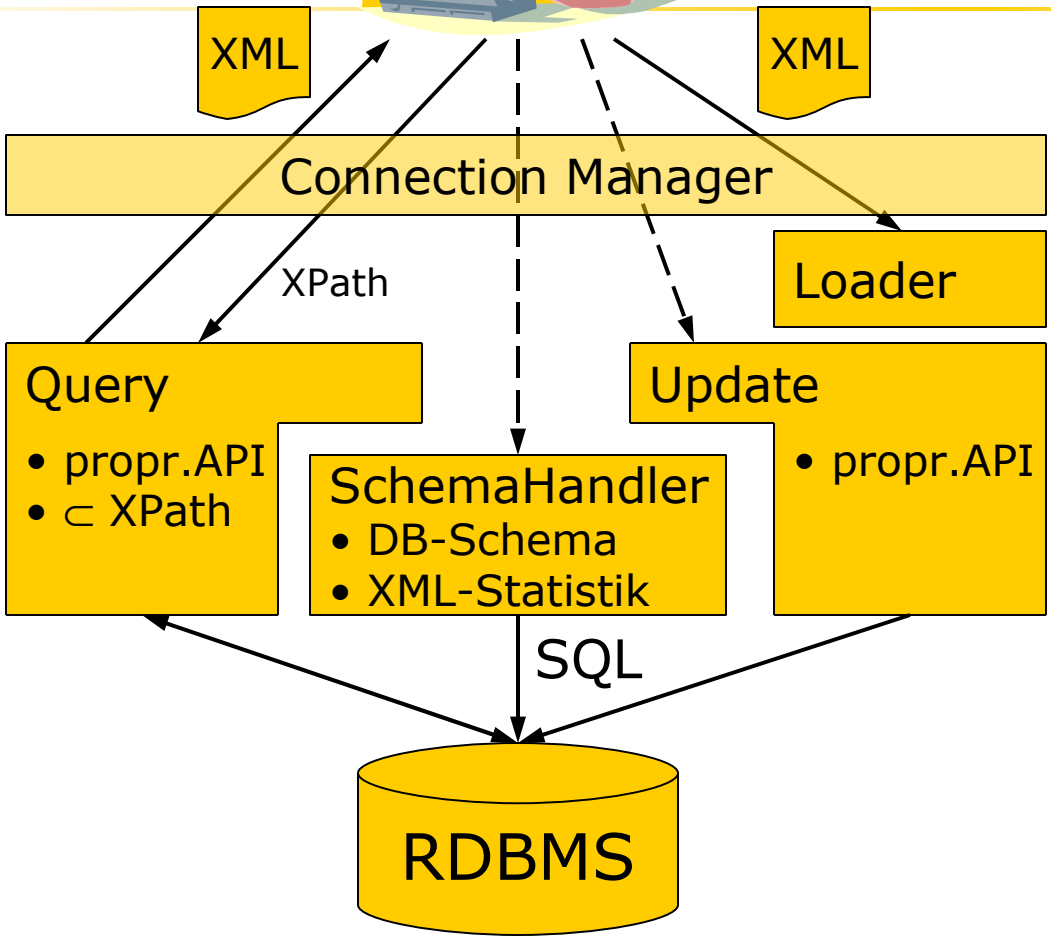
Motivation


XML
XML
XML
XML
• dokument-orientiert
XML-API
Mediator
• generisch
• schema-unabhängig
SQL
RDBMS

Ziele

- Verwaltung von XML-Daten in RDBMS
  - Unterstützung des gesamten Spektrums von XML-Dokumenten
  - Erhalt der Strukturinformation
  - Effiziente Operationen (XML-APIs: DOM, XPath)
- (automatische) Optimierung von Anfragen

XMLRDB


XML
XML
Connection Manager
XPath
Loader
Query
• propr.API
• ⊂ XPath
SchemaHandler
• DB-Schema
• XML-Statistik
Update
• propr.API
SQL
RDBMS

XMLRDB
DB-Schema

NODE

document	: int
dlnId	: long
cId	: int
parent	: int
rightSibling	: int
name	: int
value	: string
dblValue	: double
valType	: int
nodeType	: char

ATTR

document	: int
dlnId	: long
cId	: int
name	: int
value	: string
dblValue	: double
valType	: int

TEXT

document	: int
dlnId	: long
cId	: int
value	: string

NAMEMAP

nameId	: int
name	: string

NAMESPACE

prefix	: string
nsURL	: string

DOCMAP

docId	: int
docURL	: string

XPath-Umsetzung

- /Achse::Knotentest[Prädikat]
- Implementiert
  - Achsen: alle (außer namespace)
  - Knotentest: alle außer DocumentTest(), Schema*Test()
  - Prädikat: XPathExpr [op Value]
position() op Value
NumValue **to** NumValue
  - Funktionen: contains, count, empty, last, position
- nicht umgesetzt
  - logische Verknüpfungen, numerische Operatoren
  - Typhierarchie (nur String, Numeric, Boolean, Sequence)
  - for, if, Quantifizierung

XMLRDB - XPath
Anfrageverarbeitung

- parsen

```
//person/name
```

- normalisieren

```
/descendant-or-self::node()/
 child::element(person)/
 child::element(name)
```

- XPath-Optimierung
  - äquiv. Transformation

```
//name[parent::person]
```

  - Ersetzen durch Pfadindex

```
name##person#%
```

  - Ersetzung durch PathViews

```
/PATHVIEW::ID1/name
```

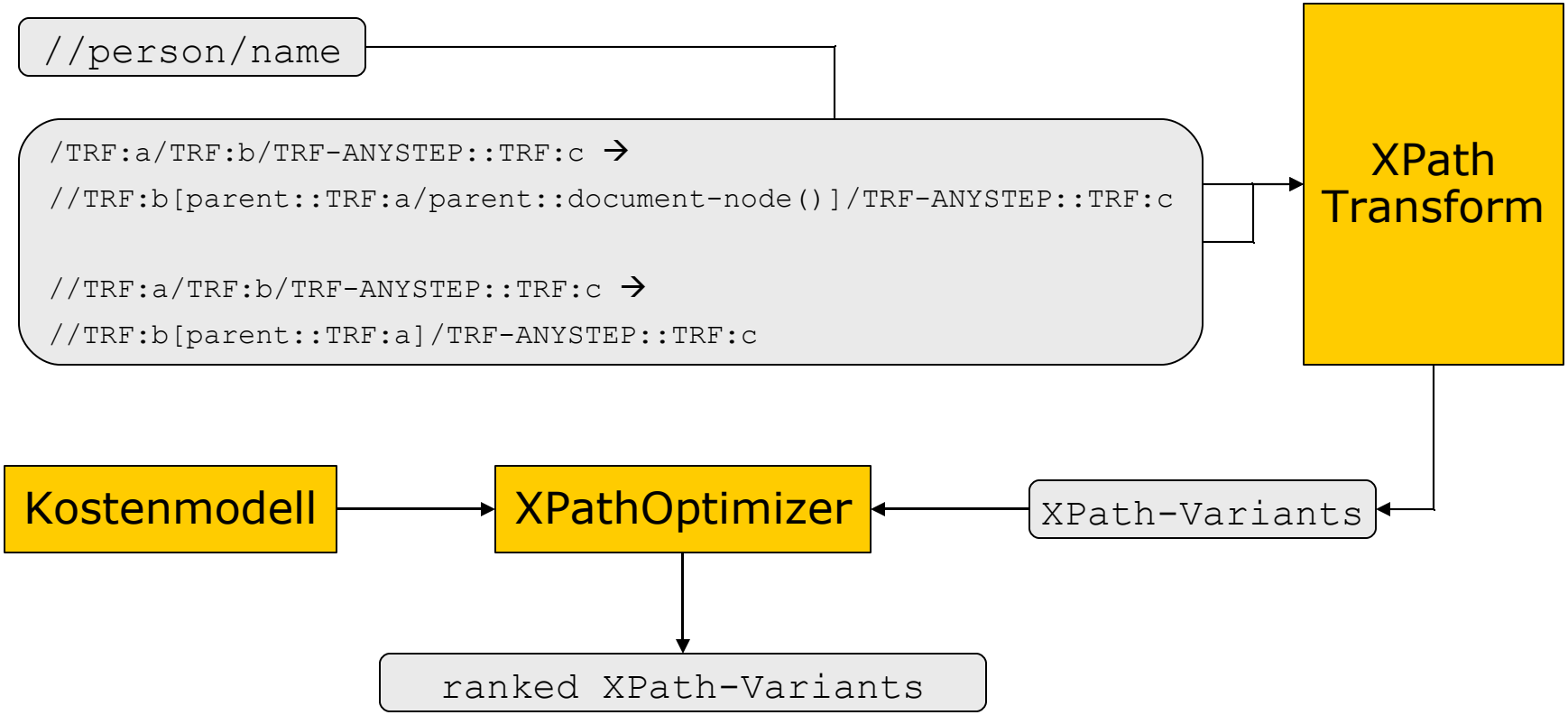
- Überführung nach SQL

```
SELECT x2.doc, x2.id0, x2.id1
FROM   xmlnode x1, xmlnode x2
WHERE  x1.name=1 AND x2.name=2 AND
       x2.parent=x1.cid
```

  - Optimierungen: ElementViews, Attribut-Inlining, ...
- SQL-Ausdruck ausführen
- Ergebnis materialisieren (3 Bereichsanfragen pro Ergebnisknoten)

XPath-Optimierungen

- äquivalente Transformation



XPath-Optimierungen

- Pfadindex

```
<site>
  <people>
    <person id='p1'>
      <name>Schmidt</name>
      <address>...
```

path	pid
#site#	1
#people##site#	2
#person##people##site#	3
#name##person##people##site#	4
#address##person##people##site#	5

```
//person/name
```

```
#name##person#%
```

pid	cid
1	1
2	2
3	3
...	...
3	21

```
SELECT  x1.doc, x1.id0, x1.id1
FROM    xmlnode x1, pathIdx p
WHERE   p.pid=4 AND x1.cid=p.cid
```

XMLRDB - XPathOpt
Parametrisierte PathViews

- Idee: materialisiere XPath (teilweise)

```
/site/regions/region[@name='Asia']/item[category='Audio']/name
//region[@name='Asia']/item[category='Video'][1 to 10]/location
```

```
//region[@name='Asia']/item[category='Audio']
```

PXPVMgmt

path	pvId
//region[@name='Asia']/item[category='Audio']	1

pvId	cidIn	cidOut
1	21	57
1	78	113

```
/site/regions/region/PATHVIEW::ID1/name
/PATHVIEW::ID1[1 to 10]/location
```

XMLRDB - XPathOpt
PXPV (2)

- Idee: Literale können parametrisiert werden

```
/site/regions/region[@name='Asia']/item[category='Audio']/name
//region[@name='Europe']/item[category='Car'][1 to 10]/location
```

```
//region[@name=$VAR1]/item[category=$VAR2]
```

PXPVMgmt

path	tbl
//region[@name=$VAR1]/item[category=$VAR2]	PV1

cidIn	V1	V2	cidOut
21	Asia	Audio	57
78	Asia	Car	113

```
/PATHVIEW::PV1_V1Europe_V2Car[1 to 10]/location
```

XMLRDB - XPath
SQL-Optimierungen

- ElementViews
  - Instanzen eines Elementtyps mit (ausgewählten) XML-Attributen in SQL-View

```
<list>
  <item id='123' count='2'>Ordner</item>
  <item id='124' count='5'>Mappe</item>
</list>
```

```
//item[@count > 2]
```

itemView				
cID	parent	value	Aid	Acount
815	814	Ordner	123	2
816	814	Mappe	124	5

XMLRDB – XPath - SQL
ElementViews (2)

- Varianten
  - einfacher SQL-View
  - materialisiert (Tabelle)
- Vorteile
  - Verringerung Anfragekomplexität
  - Datenlokalität
- Probleme
  - Homonymität von Elementnamen
  - XML-Attribute optional → Viewdefinition mittels LEFT-OUTER-JOIN

XMLRDB – XPath - SQL
ElementViews (3)

/**directory**/**host**[@*name*='ahost1']/**host**[@*name*='bhost2']/
host[@*name*='chost3']/**path**[@*name*='apath2']//*[**doc_info**]

```
SELECT  x6.doc,x6.id0,x6.id1
FROM    xmlnode x0, xmlnode x1, xmlattr l1a0, xmlnode x2, xmlattr l2a0, xmlnode x3,
        xmlattr l3a0, xmlnode x4, xmlattr l4a0, xmlnode x6, xmlnode l6x0
WHERE   x0.name=9 AND x0.parent IS NULL AND x1.name=8 AND x1.parent=x0.cid AND l1a0.name=1 AND
l1a0.cid=x1.cid AND l1a0.value='ahost1' AND x2.name=8 AND x2.parent=x1.cid AND l2a0.name=1 AND
l2a0.cid=x2.cid AND l2a0.value='bhost2' AND x3.name=8 AND x3.parent=x2.cid AND l3a0.name=1 AND
l3a0.cid=x3.cid AND l3a0.value='chost3' AND x4.name=6 AND x4.parent=x3.cid AND l4a0.name=1 AND
l4a0.cid=x4.cid AND l4a0.value='apath2' AND x6.doc=x4.doc AND ((x6.id0>x4.id0 AND
x6.id0<=xrdb_getMaxChild0(x4.id0,x4.id1)) OR (x6.id0=x4.id0 AND x6.id1>x4.id1 AND
        x6.id1<=xrdb_getMaxChild1(x4.id0,x4.id1 ))) AND l6x0.name=5 AND l6x0.parent=x6.cid
```

```
SELECT  x6.doc,x6.id0,x6.id1
FROM    xmlnode x0, vhost v1, vhost v2, vhost v3, vpath v4, xmlnode x6, xmlnode l6x0
WHERE   x0.name=9 AND x0.parent IS NULL AND v1.parent=x0.cid AND v1.attrname='ahost1' AND
        v2.parent=v1.cid AND v2.attrname='bhost2' AND v3.parent=v2.cid AND v3.attrname='chost3' AND
        v4.parent=v3.cid AND v4.attrname='apath2' AND x6.doc=v4.doc AND ((x6.id0>v4.id0 AND
        x6.id0<=xrdb_getMaxChild0(v4.id0,v4.id1)) OR (x6.id0=v4.id0 AND x6.id1>v4.id1 AND
x6.id1<=xrdb_getMaxChild1(v4.id0,v4.id1 ))) AND l6x0.name=5 AND l6x0.parent=x6.cid
```

XMLRDB – XPath - SQL
weitere Optimierungen

- 'Inlining' von Attributen

```
a[@b] → FROM  xmlnode x
        WHERE x.name='a' AND
              EXISTS(SELECT a.doc FROM xmlattr a
                     WHERE  a.cid=x.cid AND a.name='b')
        'inlining':
        FROM  xmlnode x, xmlattr a
        WHERE x.name='a' AND a.cid=x.cid AND
a.name='b'
```

- Unterscheidung bei fn:contains() nach Wort oder Phrase
- DBMS-spezifische Konstrukte (z.B. Umsetzung Ancestor-Achse bei PostgreSQL)

XMLRDB – XPath
PXPV-Advisor

- automatische Empfehlungen für sinnvolle PXPVs
- basiert auf
  - XPath Workload mit Gewichten
  - XML-Instanzstatistik
  - Kardinalitätsabschätzung

XMLRDB – XPath – PXPV-Advisor
Funktionsweise (1)

- Workload

```
10  /site/regions/region[@name='Asia']/item[category='Audio']/name
 5  //region[@name='Asia']/item[category='Video'][1 to 10]/location
```

- pro XPath bestimme alle Fragmente als PXPV-Kandidaten (auf 'sinnvolle' eingeschränkt)

```
/site/regions/region[@name='Asia']/item[category='Audio']/name
/site, /site/regions, ..., //regions, //regions/region[@name='Asia']
//@name='Asia', ...
```

- Einsparpotential für PXPV-Kandidaten bzgl. Basis-XPath bestimmen
- Fragment-Abstraktion (Literalersetzung)

```
//regions/region[@name=$VAR1]/item[category=$VAR2]/name
```

- Kandidat in Kandidatenliste einfügen
  - wenn enthalten, Einsparung addieren, Basis-XPath-ID hinzufügen

XMLRDB – XPath – PXPV-Advisor
Funktionsweise (2)

- Ersetze Variable durch Literale (bei identischem Wert)

```
//regions/region[@name='Asia']/item[category=$VAR2]
```

- sortiere Liste nach Einsparung (+Heuristiken)

```
5677  //regions/region[@name='Asia']/item[category=$VAR2]
 534  //regions/region[@name='Asia']/item[category=$VAR2]/name
```

- Kandidatenliste filtern
  - Annahme:
    - nur ein PXPV kann für Anfrage verwendet werden
    - 'teure' Anfrage produziert mehrere 'gewichtige' Kandidaten, nur einer ist einsetzbar
  - nimm ersten Kandidaten
    - lösche dort behandelte Basis-XPaths in allen anderen Fragmenten mit Anteil an Einsparung
  - sortiere Liste neu, wiederhole ersten Schritt

```
5677  //regions/region[@name='Asia']/item[category=$VAR2]
```

XMLRDB – XPath – PXPV-Advisor
Funktionsweise (3)

- Bestimmung Einsparpotential eines PXPV-Kandidaten K für XPath X
  `/X_Teil1/K/X_Teil2`
  - Ansatz 1
    - nutze externes Kostenmodell, Einsparpotential sind Kosten von K
    - Problem: unterschiedlicher Queryplan der DBMS
  - Ansatz 2
    - nutze Kostenmodell des DBMS via EXPLAIN
    - bestimme Kosten für //K (Kontext künstlich auf 1 beschränkt)
    - normalisiere Kosten über Verhältnis der Ausgangskardinalitäten von /X_Teil1/K und //K
    - Problem: Ausführungsplan von //K anders als in /X_T1/K/X_T2

XMLRDB – XPath – PXPV-Advisor
Funktionsweise (3)

- Bestimmung Einsparpotential eines PXPV-Kandidaten K für XPath X
  `/X_Teil1/K/X_Teil2`
  - Ansatz 3
    - nutze Kostenmodell des DBMS via EXPLAIN
    - verwende Dummy-PXPVs
    - bestimme C(X) = Kosten für X
    - bestimme C(X_T1/D/X_T2)
    - Einsparpotential = weight(X) * (C(X) – C(X_T1/D/X_T2))
    - Auswahl Dummy-PXPV mittels In-/Out-Kardinalität aus externem Kostenmodell

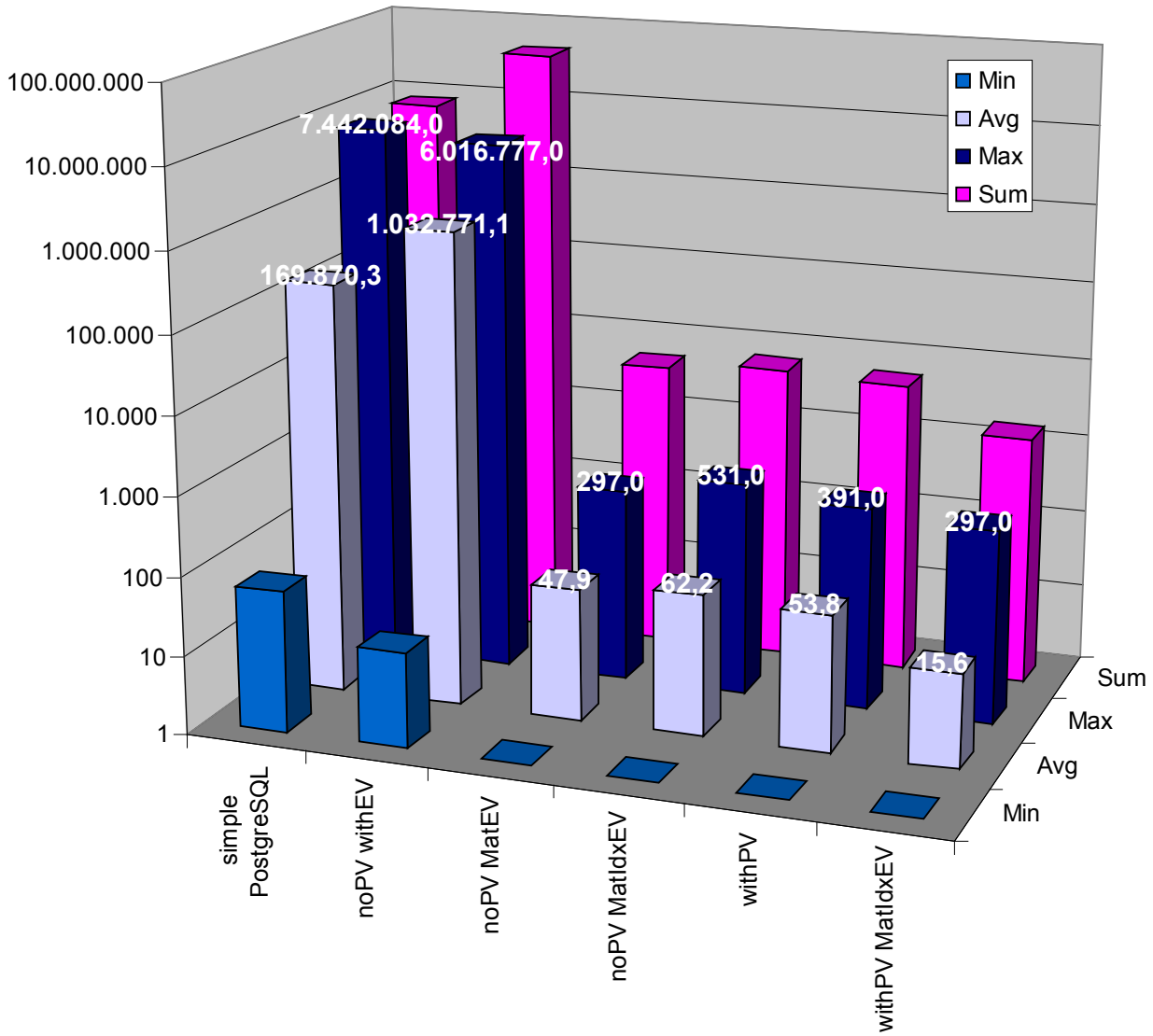
Evaluation

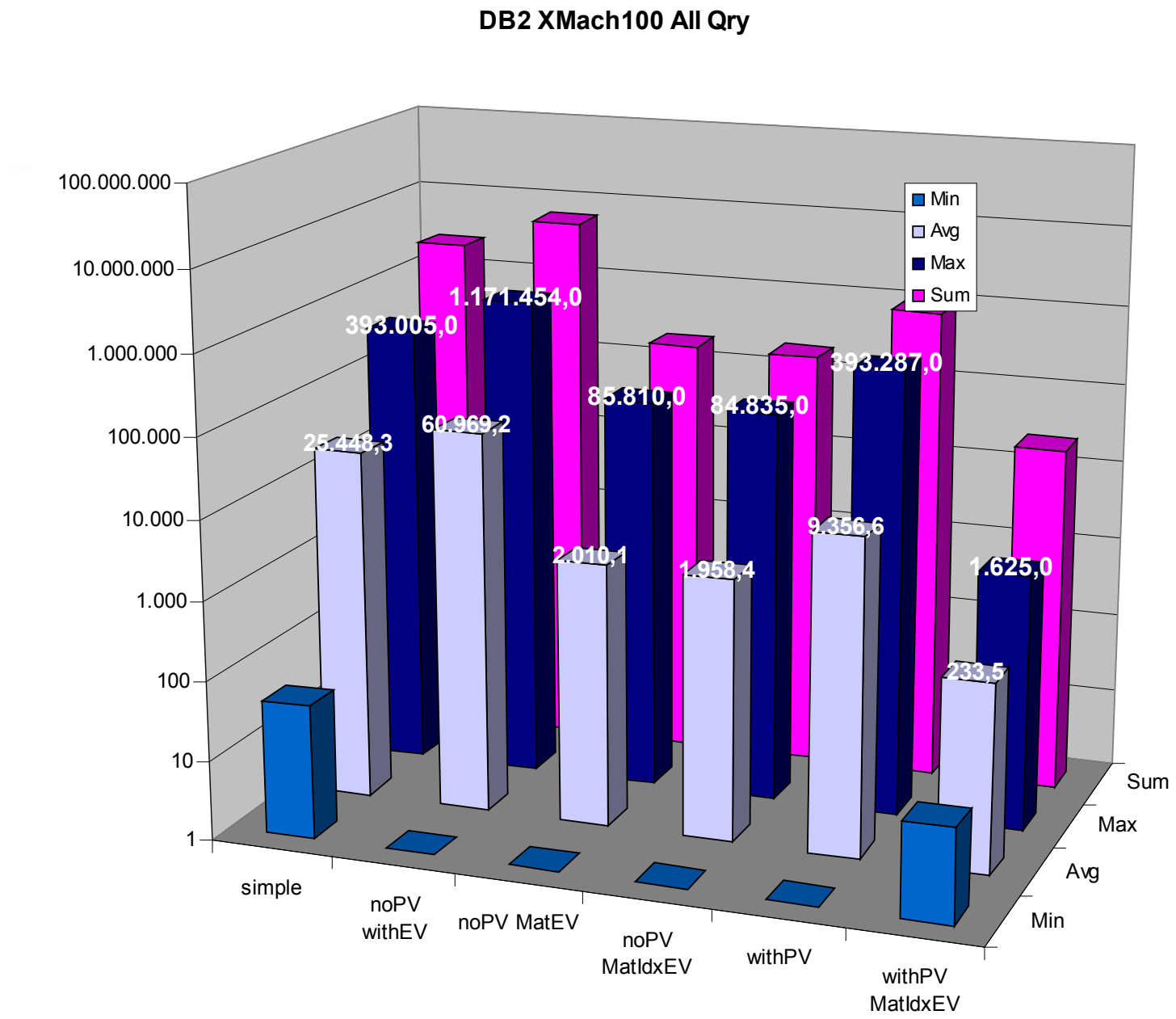
- XMach-1
  - 10.000 Dokumente, 160MB Rohdaten, ~2,2 * 10^6 Knoten, ~3,9 * 10^5 Attribute
  - 100 zufällig ausgewählte XPath-Anfragen
- Rechner: P IV 2,4 GHz, 1GB
- DBMS: DB2, PostgreSQL

Evaluation

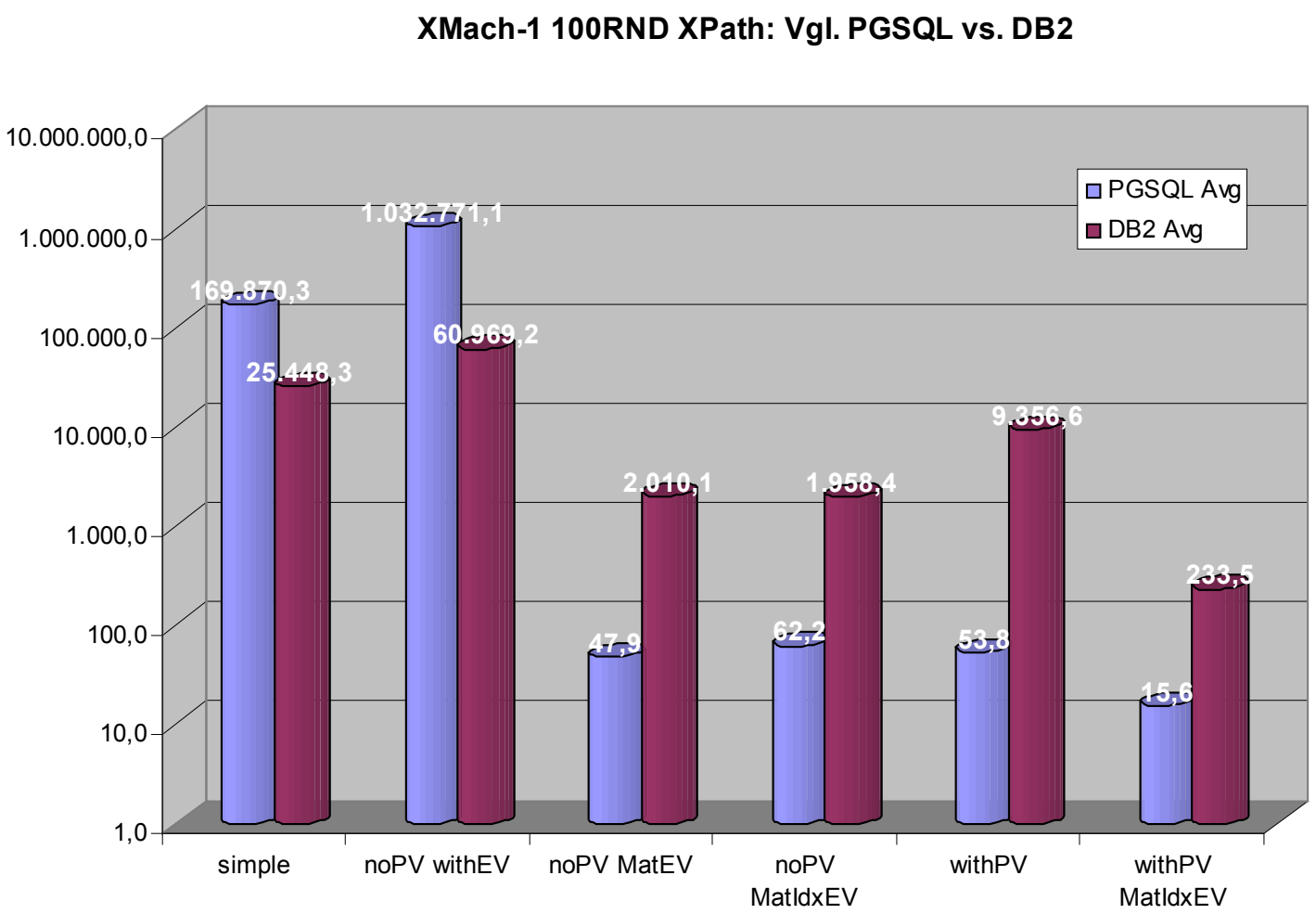
- PathViews:
  - PGSQL:
    - /descendant::host[@name=$xrdb:PW0]/host[@name=$xrdb:PW1]/host[@name = $xrdb:PW2]/path[@name = $xrdb:PW3]
    - /descendant::host[@name=$xrdb:PW0]/host[@name=$xrdb:PW1]//*[doc_info]/@name
  - DB2:
    - /directory/host[@name=$xrdb:PW0]/host[@name=$xrdb:PW1]
    - /*[@doc_id=$xrdb:PW0]

PGSQL XMach100 All Qry

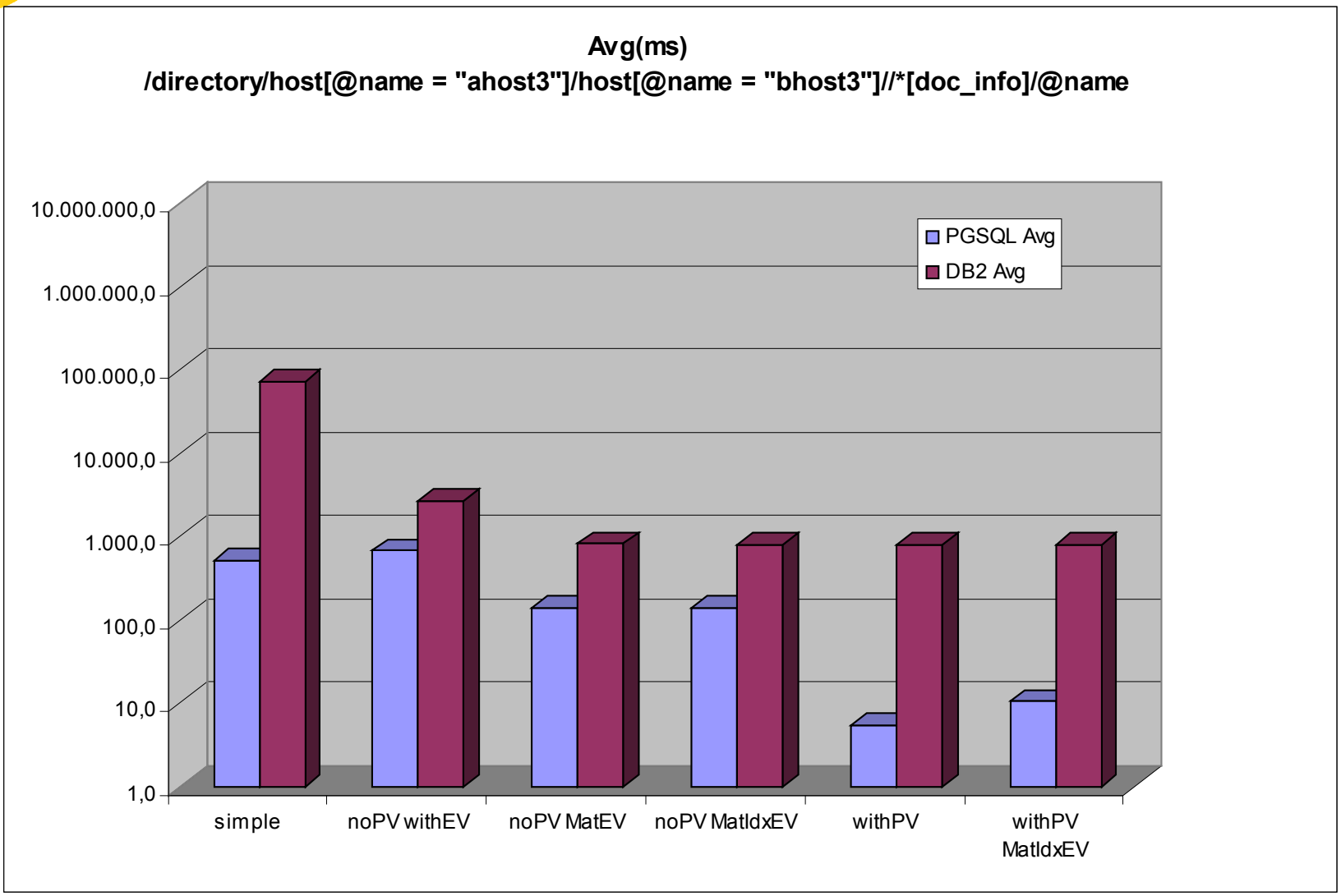



DB2 XMach100 All Qry
100.000.000
10.000.000
1.000.000
100.000
10.000
1.000
100
10
1
Min
Avg
Max
Sum
393.005,0
1.171.454,0
85.810,0
84.835,0
393.287,0
25.448,3
60.969,2
2.010,1
1.958,4
9.356,6
1.625,0
233,5
simple
noPV withEV
noPV MatEV
noPV MatIdxEV
withPV
withPV MatIdxEV
Sum
Max
Avg
Min

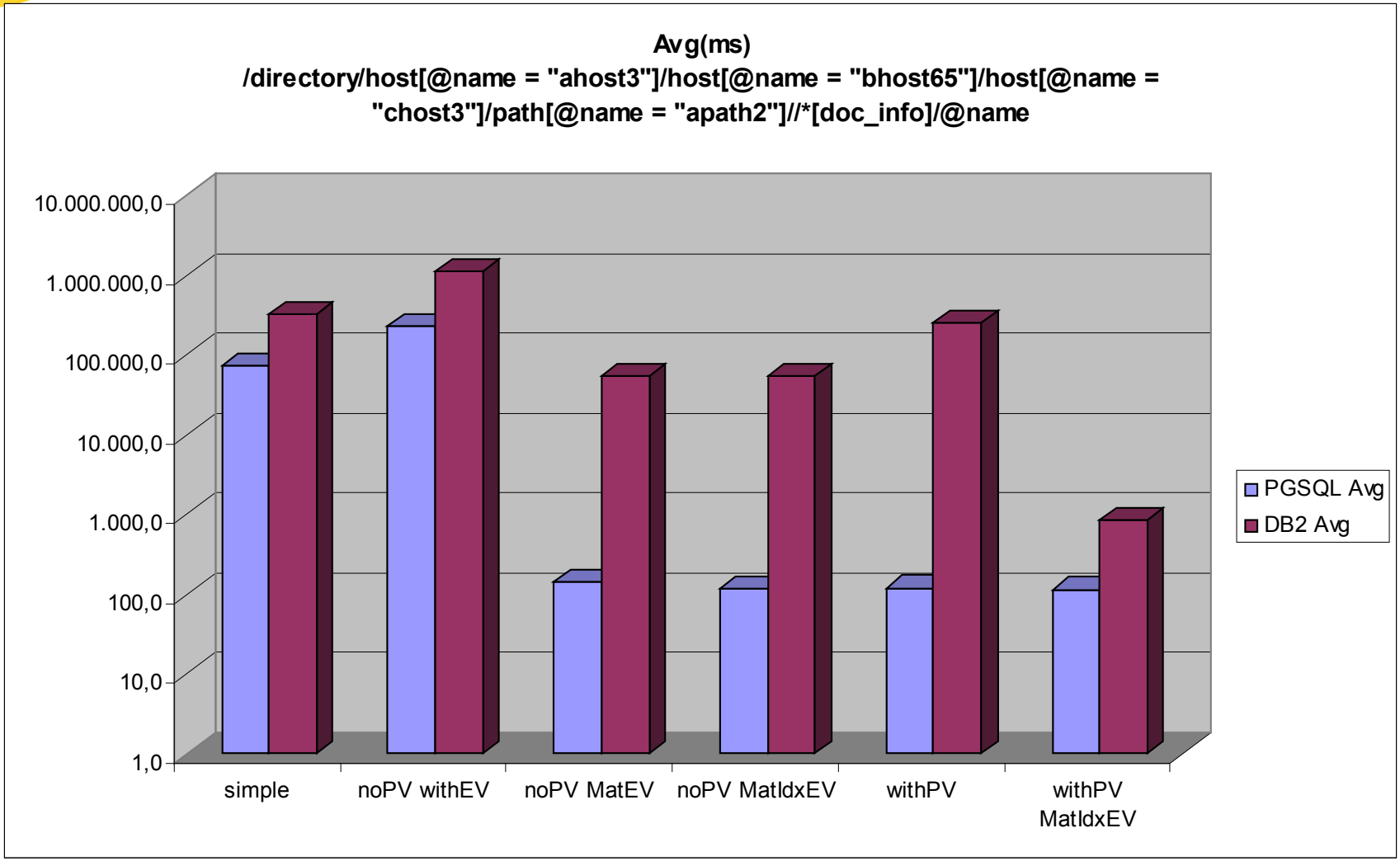
Evaluation
Gesamter Workload



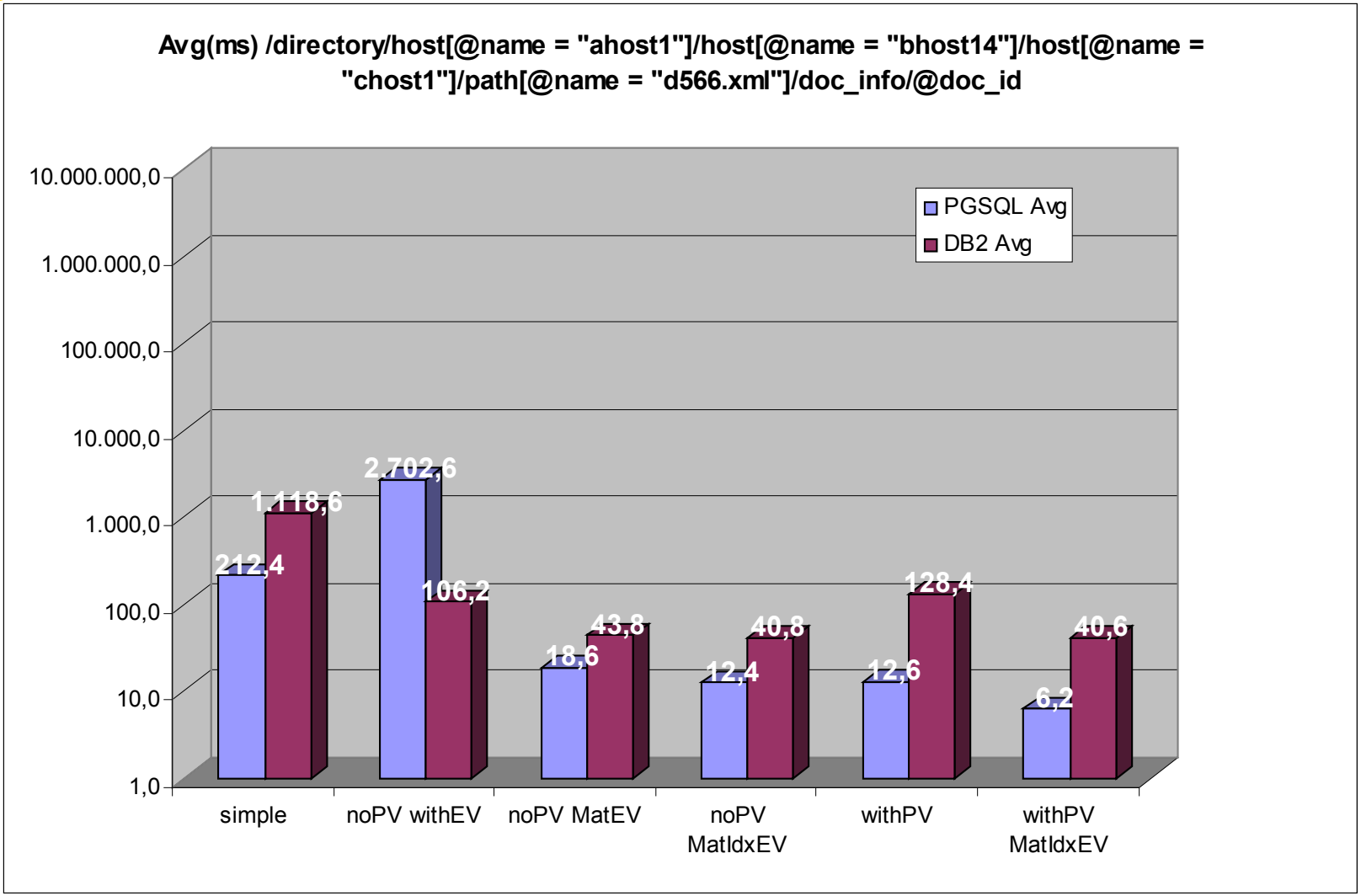
Evaluation


Avg(ms)
/directory/host[@name = "ahost3"]/host[@name = "bhost3"]//*[doc_info]/@name
10.000.000,0
1.000.000,0
100.000,0
10.000,0
1.000,0
100,0
10,0
1,0
PGSQL Avg
DB2 Avg
simple
noPV withEV
noPV MatEV
noPV MatIdxEV
withPV
withPV MatIdxEV

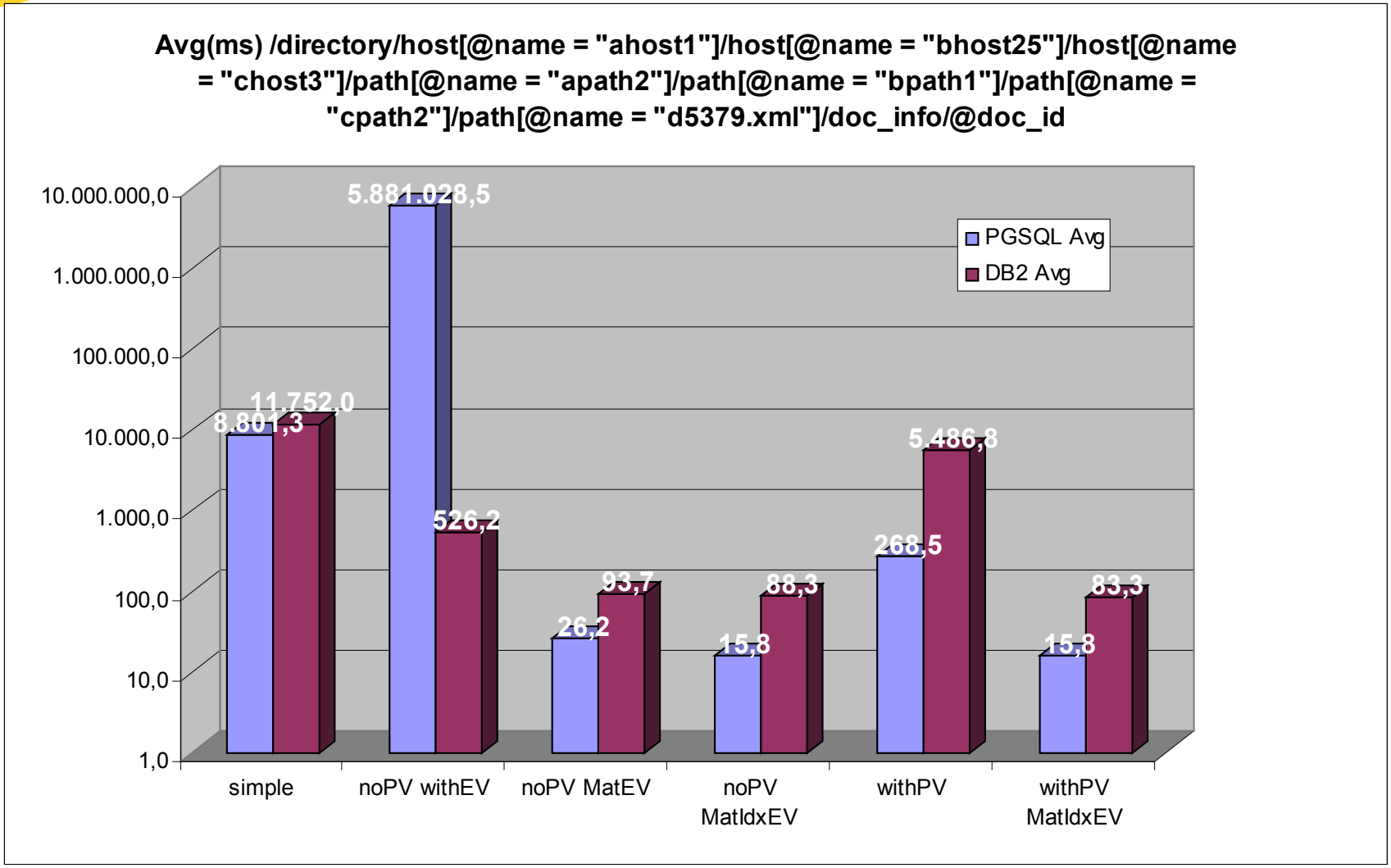
Evaluation


Avg(ms)
/directory/host[@name = "ahost3"]/host[@name = "bhost65"]/host[@name = "chost3"]/path[@name = "apath2"]//*[doc_info]/@name
10.000.000,0
1.000.000,0
100.000,0
10.000,0
1.000,0
100,0
10,0
1,0
simple
noPV withEV
noPV MatEV
noPV MatIdxEV
withPV
withPV MatIdxEV
PGSQL Avg
DB2 Avg

Evaluation


Avg(ms) /directory/host[@name = "ahost1"]/host[@name = "bhost14"]/host[@name = "chost1"]/path[@name = "d566.xml"]/doc_info/@doc_id
10.000.000,0
1.000.000,0
100.000,0
10.000,0
1.000,0
100,0
10,0
1,0
PGSQL Avg
DB2 Avg
212,4
1.118,6
2.702,6
106,2
18,6
43,8
12,4
40,8
12,6
128,4
6,2
40,6
simple
noPV withEV
noPV MatEV
noPV MatIdxEV
withPV
withPV MatIdxEV

Evaluation


Avg(ms) /directory/host[@name = "ahost1"]/host[@name = "bhost25"]/host[@name = "chost3"]/path[@name = "apath2"]/path[@name = "bpath1"]/path[@name = "cpath2"]/path[@name = "d5379.xml"]/doc_info/@doc_id
10.000.000,0
1.000.000,0
100.000,0
10.000,0
1.000,0
100,0
10,0
1,0
PGSQL Avg
DB2 Avg
8.801,3
11.752,0
5.881.028,5
526,2
26,2
93,7
15,8
88,3
268,5
5.486,8
15,8
83,3
simple
noPV withEV
noPV MatEV
noPV MatIdxEV
withPV
withPV MatIdxEV

Evaluation
PXPV-Negativbeispiel

/site/open_auctions/open_auction/bidder[1]/increase

```
SELECT DISTINCT x4.doc,x4.id0,x4.id1
FROM xmlnode x0, xmlnode x1, xmlnode x2,
        TABLE (SELECT x3.*, row_number() OVER (ORDER BY x3.doc ASC,x3.id0
                  ASC,x3.id1 ASC) AS x3RN
                    FROM xmlnode x3
                    WHERE x3.name=60 AND x3.parent=x2.cid ) AS x3T, xmlnode x4
WHERE x0.name=77 AND x0.parent IS NULL AND
          x1.name=72 AND x1.parent=x0.cid AND
          x2.name=42 AND x2.parent=x1.cid AND
          x3T.x3RN =1 AND x4.name=59 AND x4.parent=x3T.cid
ORDER BY x4.doc,x4.id0,x4.id1
```

Query execution time: 5266

/site/open_auctions/PATHVIEW::ID1

```
SELECT DISTINCT x2.doc,x2.id0,x2.id1
FROM xmlnode x0, xmlnode x1, pv_view p2, xmlnode x2
WHERE x0.name=77 AND x0.parent IS NULL AND
               x1.name=72 AND x1.parent=x0.cid AND
               p2.pid=1 AND p2.startid=x1.cid AND p2.endid=x2.cid
ORDER BY x2.doc,x2.id0,x2.id1
```

Query execution time: 8094