Query Rewrites with Views

for XML in DB2

Parke Godfrey[1,3] **Jarek Gryz[1,3]** **Andrzej Hoppe[2]**

Wenbin Ma[4] **Calisto Zuzarte[3,4]**

[1] York University
Toronto, Canada

[3] Centre for Advanced Studies
IBM, *Toronto, Canada*

[2] Microsoft
Vancouver, Canada

[4] IBM Toronto Laboratory
Toronto, Canada

30 March 2009

ICDE

Shanghai, China

Query Rewrites with Views

for XML in DB2

Parke Godfrey[1,3] **Jarek Gryz**[1,3] **Andrzej Hoppe**[2]

Wenbin Ma[4] **Calisto Zuzarte**[3,4]

[1]York University
Toronto, Canada

[3]Centre for Advanced Studies
IBM, *Toronto, Canada*

[2]Microsoft
Vancouver, Canada

[4]IBM Toronto Laboratory
Toronto, Canada

30 March 2009

ICDE

Shanghai, China

Query Rewrites with Views

for XML in DB2

Parke Godfrey[1,3] **Jarek Gryz**[1,3] **Andrzej Hoppe**[2]

Wenbin Ma[4] **Calisto Zuzarte**[3,4]

[1]York University
Toronto, Canada

[3]Centre for Advanced Studies
IBM, *Toronto, Canada*

[2]Microsoft
Vancouver, Canada

[4]IBM Toronto Laboratory
Toronto, Canada

30 March 2009

ICDE

Shanghai, China

Query Rewrites with Views

for XML in DB2

Parke Godfrey[1,3] **Jarek Gryz**[1,3] **Andrzej Hoppe**[2]

Wenbin Ma[4] **Calisto Zuzarte**[3,4]

[1]York University
Toronto, Canada

[3]Centre for Advanced Studies
IBM, *Toronto, Canada*

[2]Microsoft
Vancouver, Canada

[4]IBM Toronto Laboratory
Toronto, Canada

30 March 2009

ICDE

Shanghai, China

Query Rewrites with Views

for XML in DB2

Parke Godfrey[1,3] **Jarek Gryz**[1,3] **Andrzej Hoppe**[2]

Wenbin Ma[4] **Calisto Zuzarte**[3,4]

[1]York University
Toronto, Canada

[3]Centre for Advanced Studies
IBM, *Toronto, Canada*

[2]Microsoft
Vancouver, Canada

[4]IBM Toronto Laboratory
Toronto, Canada

30 March 2009

ICDE

Shanghai, China

Query Rewrites with Views

for XML in DB2

Parke Godfrey[1,3] **Jarek Gryz**[1,3] **Andrzej Hoppe**[2]

Wenbin Ma[4] **Calisto Zuzarte**[3,4]

[1]York University
Toronto, Canada

[3]Centre for Advanced Studies
IBM, *Toronto, Canada*

[2]Microsoft
Vancouver, Canada

[4]IBM Toronto Laboratory
Toronto, Canada

30 March 2009

ICDE

Shanghai, China

Query Rewrites with Views

for XML in DB2

Parke Godfrey[1,3] **Jarek Gryz**[1,3] **Andrzej Hoppe**[2]

Wenbin Ma[4] **Calisto Zuzarte**[3,4]

[1]York University
Toronto, Canada

[3]Centre for Advanced Studies
IBM, *Toronto, Canada*

[2]Microsoft
Vancouver, Canada

[4]IBM Toronto Laboratory
Toronto, Canada

30 March 2009

ICDE

Shanghai, China

I. Motivation & Background

The Two Towers

In today's big, commercial database systems, relational and XML stand side by side.

Much work has been done

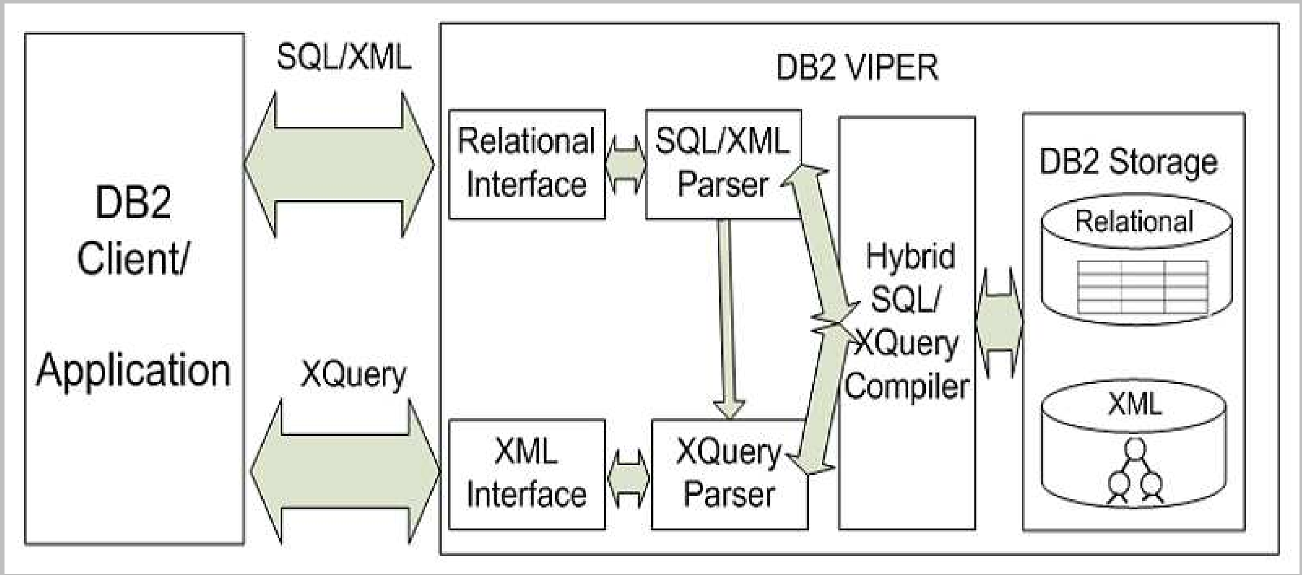
- to *handle* XML *efficiently*,
  - native XML storage
  - XML indexes
- and to *integrate* relational and XML data
  - SQL can query XML data
  - XQuery can query relational data

I. Motivation & Background

The Two Towers

In today's big, commercial database systems, relational and XML stand side by side.

Architecture of DB2 9 Viper.



I. Motivation & Background

The Two Towers

In today's big, commercial database systems, relational and XML stand side by side.

Logical integration of XML in relational

- `XML` is a data type for columns
  - each value is an XML document
  - an `XML` column can be contrained by XML Schemas.
- table function `XMLTable` extends SQL
  - part of the SQL standards (SQL/XML)

I. Motivation & Background

The Two Towers

In today's big, commercial database systems, relational and XML stand side by side.

Challenge

- SQL and relational database systems have nearly 40 years of optimization technology.
- Queries over XML (and hybrid) can be quite inefficient.
  - many fewer robust optimization techniques
  - queries are inherently "more complex"

Can we adapt successful techniques for relational queries for XML (and hybrid)?

Optimization

Materialized Views

A *materialized view* is simply a view that is evaluated and stored as a table on disk.
It is maintained and updated by the database system.

- This can speed up the evaluation of other queries.
- But updates to the tables in the view's definition will (usually) require updates to the view's table.

Optimization

Materialized Views

A *materialized view* is simply a view that is evaluated and stored as a table on disk.
It is maintained and updated by the database system.

- This can speed up the evaluation of other queries.
- But updates to the tables in the view's definition will (usually) require updates to the view's table.

The technology of materialized views

- is quite well studied, and
- is effectively implemented and used in relational database systems.

Optimization

Materialized Views

A *materialized view* is simply a view that is evaluated and stored as a table on disk.
It is maintained and updated by the database system.

- This can speed up the evaluation of other queries.
- But updates to the tables in the view's definition will (usually) require updates to the view's table.

Q. Can we do the same for views that use `XMLTable` to optimize SQL/XML queries?

SQL/XML

```
SELECT x.title, x.sections
FROM books,
  XMLTable ('$book/book/chapter'
    PASSING bookdoc AS "book"
    COLUMNS
      "title" VARCHAR (60)
                    PATH 'title',
      "sections" XML PATH 'section'
) AS x;
```

SQL/XML

```
SELECT x.title, x.sections
FROM books,
  XMLTable ('$book/book/chapter'
    PASSING bookdoc AS "book"
    COLUMNS
      "title" VARCHAR (60)
                  PATH 'title',
      "sections" XML PATH 'section'
) AS x;
```

```
<?xml version="1.0" ...?>
<book>
  <chapter>
    <title>Introduction</title>
    <section>Goals</section>
  </chapter>
  <chapter>
    <title>Related</title>
    <section>Views</section>
    <section>XML</section>
  </chapter>
  <chapter>
    <title>Solutions</title>
  </chapter>
</book>
```

SQL/XML

```sql
SELECT x.title, x.sections
FROM books,
  XMLTable ('$book/book/chapter'
    PASSING bookdoc AS "book"
    COLUMNS
      "title" VARCHAR (60)
                   PATH 'title',
      "sections" XML PATH 'section'
) AS x;
```

- **row generator:** Generates rows based on documents in the XML column.

```xml
<?xml version="1.0" ...?>
<book>
  <chapter>
    <title>Introduction</title>
    <section>Goals</section>
  </chapter>
  <chapter>
    <title>Related</title>
    <section>Views</section>
    <section>XML</section>
  </chapter>
  <chapter>
    <title>Solutions</title>
  </chapter>
</book>
```

SQL/XML

```
SELECT x.title, x.sections
FROM books,
  XMLTable ('$book/book/chapter'
    PASSING bookdoc AS "book"
    COLUMNS
      "title" VARCHAR (60)
                    PATH 'title',
      "sections" XML PATH 'section'
) AS x;
```

```
<?xml version="1.0" ...?>
<book>
  <chapter>
    <title>Introduction</title>
    <section>Goals</section>
  </chapter>
  <chapter>
    <title>Related</title>
    <section>Views</section>
    <section>XML</section>
  </chapter>
  <chapter>
    <title>Solutions</title>
  </chapter>
</book>
```

- **row generator:** Generates rows based on documents in the XML column.
- **navigtors:** Populate the columns' values for those rows.

SQL/XML

```
SELECT x.title, x.sections
FROM books,
  XMLTable ('$book/book/chapter'
    PASSING bookdoc AS "book"
    COLUMNS
      "title" VARCHAR (60)
                  PATH 'title',
      "sections" XML PATH 'section'
) AS x;
```

```
<?xml version="1.0" ...?>
<book>
  <chapter>
    <title>Introduction</title>
    <section>Goals</section>
  </chapter>
  <chapter>
    <title>Related</title>
    <section>Views</section>
    <section>XML</section>
    /chapter>
    hapter>
    <title>Solutions</title>
    /chapter>
   ok>
```

title	**sections**
Introduction	<section>Goals</section>
Related	<section>Views</section> <section>XML</section>
Solutions	null

II. Materialized Views for SQL/XML

The Challenges

A *matching* and *compensation* framework is needed.

- *matching:* discovering whether the view is equivalent to, or contains, the query.
- *compensation:* determine further restrictions—navigation steps and predicates—which, when applied to the view, is equivalent to the query.

II. Materialized Views for SQL/XML

• The Challenges

A *matching* and *compensation* framework is needed.

- *matching:* discovering whether the view is equivalent to, or contains, the query.
- *compensation:* determine further restrictions—navigation steps and predicates—which, when applied to the view, is equivalent to the query.

related work

- `XPath` rewrites were investigated in

  [Balmin, Özcan, Beyer, Cochrane, & Pirahesh, VLDB 2004]

  [Xu & Özsoyoglu, VLDB 2005]

- and for SQL/XML in

  [Krishnaprasad, Liu, Manikutty, Warner, Arora, & Kotsovolos, VLDB 2004]

II. Materialized Views for SQL/XML

• The Challenges

A *matching* and *compensation* framework is needed.

- *matching:* discovering whether the view is equivalent to, or contains, the query.
- *compensation:* determine further restrictions—navigation steps and predicates—which, when applied to the view, is equivalent to the query.

Considerations

- The row generator of the query and view *must* be the same.
- The navigators must
  1. be type convertible,
  2. satisfy the prefix property (be less restrictive), and
  3. satisfy compensation locality.

II. Materialized Views for SQL/XML

• The Challenges

A *matching* and *compensation* framework is needed.

- *matching:* discovering whether the view is equivalent to, or contains, the query.
- *compensation:* determine further restrictions—navigation steps and predicates—which, when applied to the view, is equivalent to the query.

A *step* is less restrictive than another if

- the steps have the same axis navigators,
- the step's test is less restrictive than the other's, and
- the step's predicate implies the other's.

Implementation

The existing matching and compensation framework in DB2 9.5 (VIPER 2) was extended to handle new types of matching using `XMLTable`.

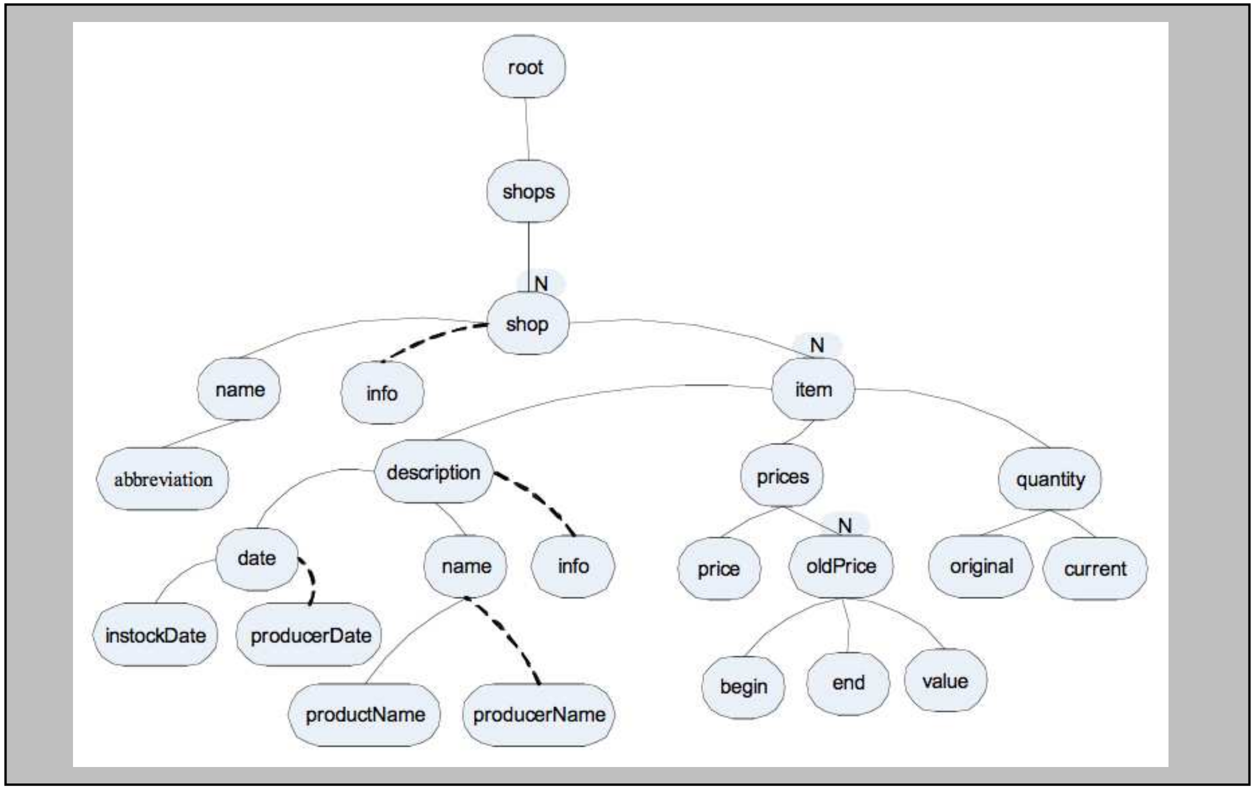
Implementation

The existing matching and compensation framework in DB2 9.5 (VIPER 2) was extended to handle new types of matching using `XMLTable`.

- Developed a subsumption algorithm.
  - Given a “subsumer” tree of size m and a “subsumee” tree of size n, takes mn steps.
- Relaxed strict *locality*.
- Integrated within DB2’s optimizer’s materialized view mechanism.
- Addressed current “impedance” issues.

Testing Environment

- A code branch of IBM DB2 9.5 was developed, implementing the above.
- 13 `XMLTable` queries were devised based on the `Shops` schema to test the different subsumption patterns.
- A 10MB and a 100MB `Shops` databses were synthetically generated.



Query Suite

```
SELECT xtable.column
FROM shops, XMLTable ('$r/root//item'
    PASSING xmldoc AS "r"
    COLUMNS
       "column"  XML PATH
           'description/date/instockDate[@year]'
) AS xtable;
```

Query 4

```
SELECT xtable.column
FROM shops, XMLTable ('$r/root//item'
    PASSING xmldoc AS "r"
    COLUMNS
       "column"  XML PATH 'description/date'
) AS xtable;
```

View 4

```
SELECT xtable.column
FROM shops, XMLTable ('$r/root//item'
    PASSING xmldoc AS "r"
    COLUMNS
       "column"  XML PATH
           'description/date/instockDate[@year]'
) AS xtable;
```

Rewrite 4

Query Suite

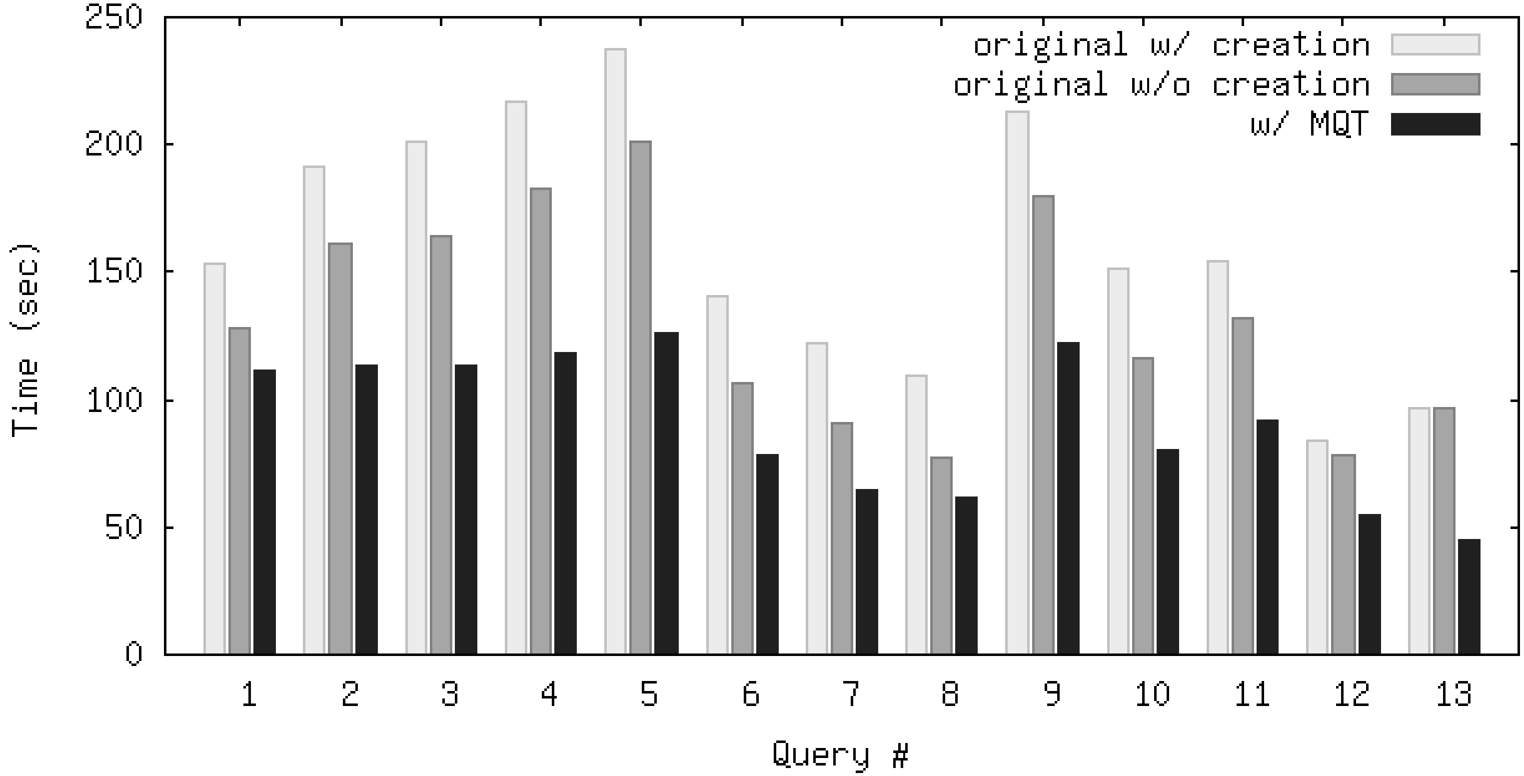
#	Row Generator / MQT Navigator	Query Navigator / Compensation
1	`/root/shops/shop/item`	`prices/price`
	`prices`	`prices/price`
2	`/root//item`	`prices/price`
	`prices`	`prices/price`
3	`/root//item`	`description/date/instockDate`
	`description`	`description/date/instockDate`
4	`/root//item`	`description/date/instockDate[@year]`
	`description/date`	`date/instockDate[@year]`
5	`/root//item`	`description/date/instockDate[@year and @month and @day]`
	`description/date`	`date/instockDate[@year and @month and @day]`
6	`/root//item`	`description[info]/date/instockDate[@year>2006]`
	`description`	`description[info]/date/instockDate[@year>2006]`

Query Suite

#	Row Generator	Query Navigator
	MQT Navigator	Compensation
7	/root//item	description[contains(info,"car")]/name
	description	description[contains(info,"car")]/name
8	/root//item	description/date/productionDate[@year>2005]/../instockDate
	description	description/date/productionDate[@year>2005]/../instockDate
9	/root//item	description[date]/name/productName
	description[date]	description/name/productName
10	/root//item	description[info and date]/name/productName
	description[date]	description[info and date]/name/productName
11	/root/*/*/item	description/date
	description	description/date
12	/root//*//shop	name/abbreviation
	name/*	abbreviation

Experimental Results

& Lessons Learned



Times over the 100MB dataset.

III. Conclusions & Future Work

conclusions

- This work offers a strong evidence that materialized views can be highly effective for SQL/XML queries.
- Is the first work, to the best of our knowledge, that considers use of materialized views to improve SQL/XML performance for XML data.
- While our present matching and compensation is quite simple, it seems applicable for a wide range of SQL/XML queries.

III. Conclusions & Future Work

conclusions

- This work offers a strong evidence that materialized views can be highly effective for SQL/XML queries.
- Is the first work, to the best of our knowledge, that considers use of materialized views to improve SQL/XML performance for XML data.
- While our present matching and compensation is quite simple, it seems applicable for a wide range of SQL/XML queries.

future work

1. Extend the matching and compensation framework for a wider class of XML/SQL queries.
2. Move the framework into the cost-based optimizer.
3. Make the framework applicable to XQuery.