

**ADT 2009**

**XML/XQuery Data Management  
MonetDB/XQuery (1/2)**

**Beyond Chapter 10 of  
Silberschatz, Korth, Sudarshan  
“Database System Concepts”**

Stefan Manegold  
Stefan.Manegold@cwi.nl  
<http://www.cwi.nl/~manegold/>

# XML Databases

## *why*

- *Motivation & The Big Picture*

## *what*

- *Crash Course XQuery*

## **WHO** ← ←

- *XML files* → *Saxon, Galax, GNU Qexo*
- *XML DBMS* → *eXist, BerkeleyDB, MonetDB, X-Hive, Tamino, Xyleme*
- *XML EAI* → *BEA Liquid Data, Data Direct, Mark Logic, Mono, Ipedo*
- *XML RDBMS* → *Oracle10g, SQLserver 2005, DB2*

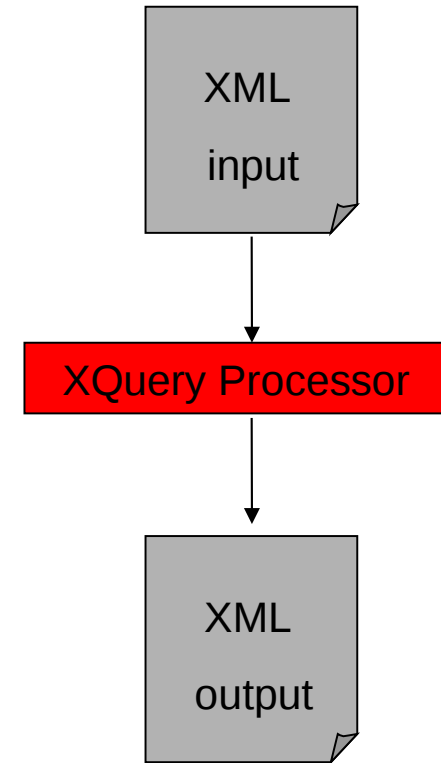
## *how*

- *Under The Hood of MonetDB/XQuery*
- *Some Benchmarks*

# XML Data Management Systems

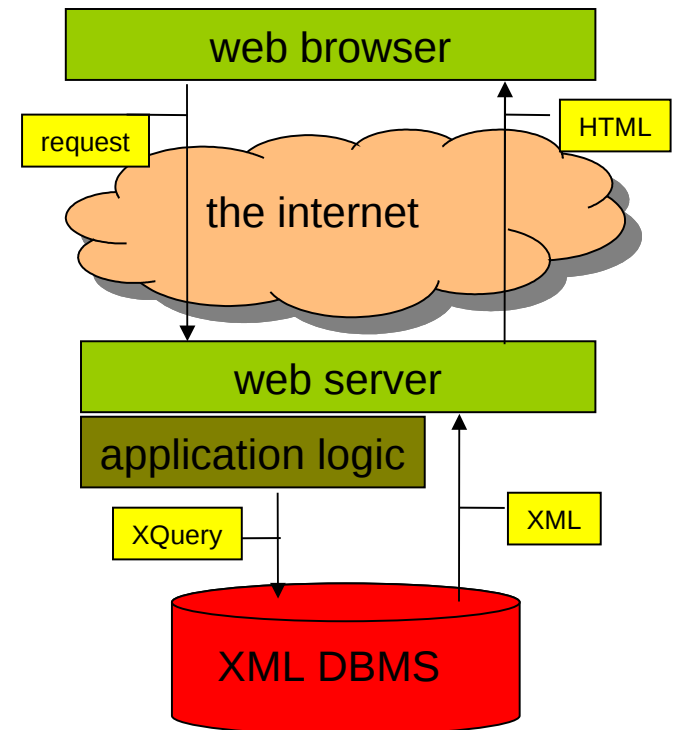
- **XML File Processors**

- *Used as part of a document processing pipeline*
- *Small documents (messages)*



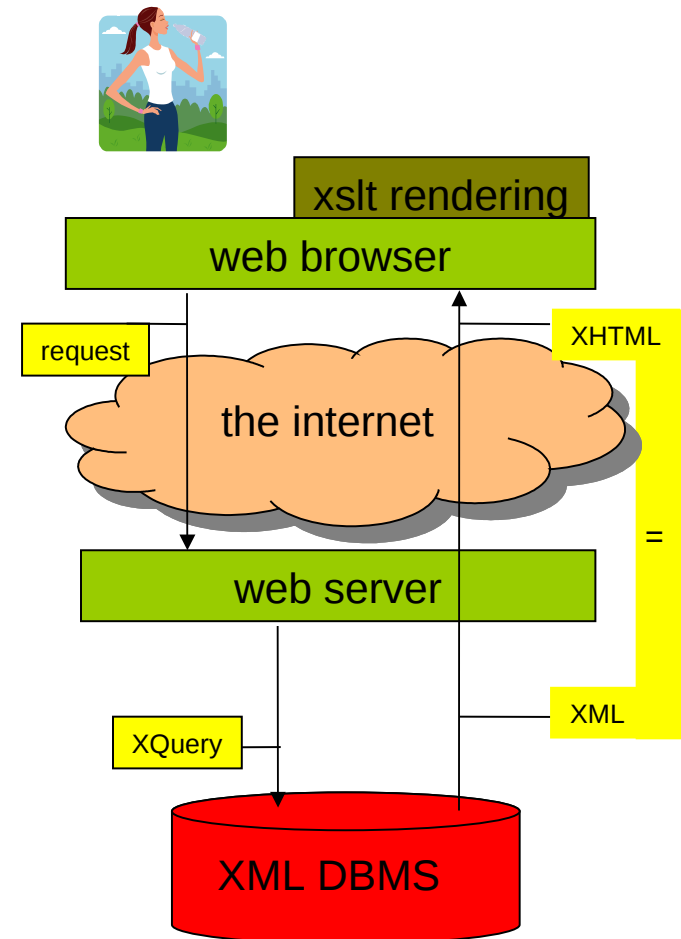
# XML Data Management Systems

- XML File Processors
  - *Used as part of a document processing pipeline*
  - *Small documents (messages)*
- **XML Databases**
  - *Manage large collections of XML documents*
  - *Text Keyword Search Support (XML contains text..)*
  - *Integration with Web Servers/Platforms*



# XML Data Management Systems

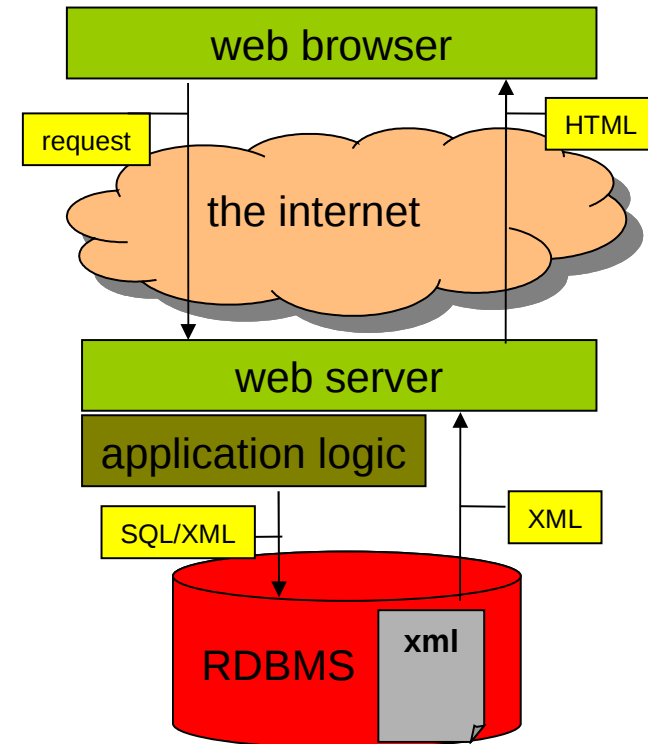
- XML File Processors
  - *Used as part of a document processing pipeline*
  - *Small documents (messages)*
- **XML Databases**
  - *Manage large collections of XML documents*
  - *Text Keyword Search Support (XML contains text..)*
  - *Integration with Web Servers/Platforms*



# XML Data Management Systems



- XML File Processors
  - Used as part of a document processing pipeline
  - Small documents (messages)
- XML Databases
  - Manage large collections of XML documents
  - Text Keyword Search Support (XML contains text..)
  - Integration with Web Servers/Platforms



- **RDBMS with XML Functionality**

- Easily mix relational and XML, can be very useful for that .ini/properties file
- Query the XML VARCHAR with SQL/XML → ugly but works

# XML Data Management Systems

(Google hits)

(ranking)

<b>Saxon</b>	xquery	<b>743K</b>	<b>#1</b>
<b>Qexo</b>	xquery	<b>41K</b>	<b>#2</b>
<b>Galax</b>	xquery	<b>17K</b>	<b>#3</b>

## XML File Processors

<b>eXist</b>	xquery	<b>1440K</b>	<b>#1</b>
<b>Mark Logic</b>	xquery	<b>438K</b>	<b>#2</b>
<b>Xindice</b>	xquery	<b>225K</b>	<b>#2</b>
<b>BerkeleyDB XML</b>	xquery	<b>142K</b>	<b>#3</b>
<b>MonetDB</b>	xquery	<b>87K</b>	<b>#4</b>
<b>X-Hive</b>	xquery	<b>65K</b>	<b>#5</b>
<b>Tamino</b>	xquery	<b>56K</b>	<b>#6</b>

## XML Databases

<b>Oracle</b>	xquery	<b>1470K</b>	<b>#1</b>
<b>SQLServer</b>	xquery	<b>1220K</b>	<b>#2</b>

## RDBMS with SQL/XML Functionality

# XML Databases

- **Querying**
- How well is XPath/XQuery implemented?
  - All Axes? Collations? XMLSchema? Dynamic Typing? Modules? Recursive UDF?
- **Updates**
- What update dialect is used? (XUpdate / updateX / WebDAV / other)
  - W3C Update Facility Proposal (since jan 2006!)
- **DB properties**
- Query performance/throughput (benchmarks published?)
- Update consistency model (fully serializable / snapshot consistency / s.th. less)
- Replication, Failover, Backup facilities
- **APIs**
- SOAP / WSDL
  - Call a query from outside  $\Leftrightarrow$  Calling out from a query
- Web Support
  - Cocoon/apache modules (web sessions, low overhead web queries)
  - XML Beans (J2EE served from XMLDB)
  - WebDAV
- Language bindings
  - XQJ  $\Rightarrow$  Java Binding / Xlinq  $\Rightarrow$  C# Binding

# XML DBMS comparison

	xquery	updates	scalability	replication	fulltext	APIs
<b>eXist</b>	++	+	-	+	+	++
<b>Mark Logic</b>	++	+	+	+	++	+
<b>Apache Xindice</b>	-	+	-	-	-	+
<b>BerkeleyDB XML</b>	++	++	++	+	+	+
<b>MonetDB/XQuery</b>	+	++	+++	-	-	+
<b>X-Hive</b>	++	++	++	++	+	+++
<b>Tamino</b>	+	+	+	++	+	++

# XML in a Relational DBMS

- **Store XML as a BLOB in relational database**
  - index by materializing indexed expressions in separate columns
  - Plus: store XML in parsed and validated form
  - Minus: proprietary solution (blob is a black box)
  - Minus: replicate data for indexing
- **Schema-based Shredding**
  - Map XML Schema / DTD to SQL DDL
  - Plus: integrates well with relational data
  - Minus: missing tools, complicated
- **SQL / XML**
  - Extend SQL with XML data type
  - Plus: integrates well with relational data
  - Minus: not clear how it integrates with application, odd „marriage“

# History of SQL / XML

- **First edition part of SQL:2003**
  - Part 14 of the SQL standard
  - Pre-dates XQuery standard!!!
  - Limited functionality - storage and publishing
- **Second edition**
  - More complete integration of XQuery + XQuery Data Model
  - Advanced Query capabilities
  - Published in 2006

# Publishing Rel. Data as XML (1/2)

## Phantasy-People

Id	Name
4711	Wutz
911	Potter

```
<Phantasy-People>
```

```
<row>
```

```
<Id>4711</Id>
```

```
<Name>Wutz</Name>
```

```
</row>
```

```
<row>
```

```
<Id>911</Id>
```

```
<Name>Potter</Name>
```

```
</row>
```

```
</Phantasy-People>
```

# Publishing Rel. Data as XML (2/2)

## SQL/XML Publishing Example

```

SELECT
  XMLELEMENT(NAME "Department",
    XMLATTRIBUTES (e.department AS "name" ),
    XMLAGG( XMLELEMENT(NAME "emp", e.firstname) )
      ) AS "department_list"
FROM employee e
WHERE .....
GROUP BY e.department;
  
```

firstname	lastname	department
SEAN	LEE	A00
MICHAEL	JOHNSON	B01
VINCENZO	BARELLI	A00
CHRISTINE	SMITH	A00

department\_list

```

<Department name="A00">
  <emp>CHRISTINE</emp>
  <emp>VINCENZO </emp>
  <emp>SEAN</emp>
</Department>

<Department name="B01">
  <emp>MICHAEL</emp>
</Department>
  
```

# Publishing XML as Rel. Table

## XMLTable: make table from XML

- Each item in the sequence returned by the XQuery will result in a row
- The row data is created with that item as context

```

SELECT X.* from
XMLTABLE ('$MyDB//customer'
  COLUMNS
  CID          INTEGER          PATH '@id',
  Name        VARCHAR(30)     PATH 'name',
  Type        CHAR(2)         PATH 'name/@type',
  Zip         XML              PATH 'zip'
) AS X

```

CID	NAME	TYPE	ZIP
1325	Bobby	US	<zip>33129</zip>
4711	Jane	US	<zip>95023</zip>

# XML Type in SQL

- A new type (like varchar, date, numeric)
- SQL:2003 - XML type restricted to
  - XML document or
  - XML element or
  - Sequence of XML elements
- SQL / XML, 2nd edition
  - Full support of XQuery Data Model
  - XML(SEQUENCE), XML(ANY CONTENT), ...

## Example (SQL:2003)

```
create table books(
  title    varchar(20),
  authors  XML);
```

**No schema validation, no typing!**

<b>Title</b>	<b>Authors</b>
XQuery 1.0	<author>D. Chamberlin </author> <author>D. Florescu</author> <author>et al.</author>
MonetDB	„P. Boncz“

- SQL:2006 => explicit validation with **XMLVALIDATE ( )**

# XMLQuery

## XMLQuery (

XQuery-expression

PASSING { BY REF | BY VALUE }

(value-expression AS identifier [BY REF | BY VALUE] )\*

RETURNING { CONTENT | SEQUENCE } { BY REF | BY VALUE }

)

- If PASSING value has no identifier, then that is context node
- BY REF - preserves Id (of an XML type)
- BY VALUE - creates a copy of the data

# XML Databases

## *why*

- *Motivation & The Big Picture*

## *what*

- *Crash Course XQuery*

## *who*

- *XML files* → *Saxon, Galax, GNU Qexo*
- *XML DBMS* → *eXist, BerkeleyDB, MonetDB, X-Hive, Tamino, Xyleme*
- *XML EAI* → *BEA Liquid Data, Data Direct, Mark Logic, Mono, Ipedo*
- *XML RDBMS* → *Oracle10g, SQLserver 2005, DB2*

## **HOW** ← ←

- *Under The Hood of MonetDB/XQuery*
- *Some Benchmarks*

# XQuery Systems: 2 Approaches

## Native

Tree is basic data structure

- tree-storage manager
- tree-query processing (algebra)
- tree-query optimization

Re-inventing the *wheel*?

X-Hive

Timber

DB2

BDB-XML

Galax

Microsoft

Oracle

MonetDB/  
XQuery

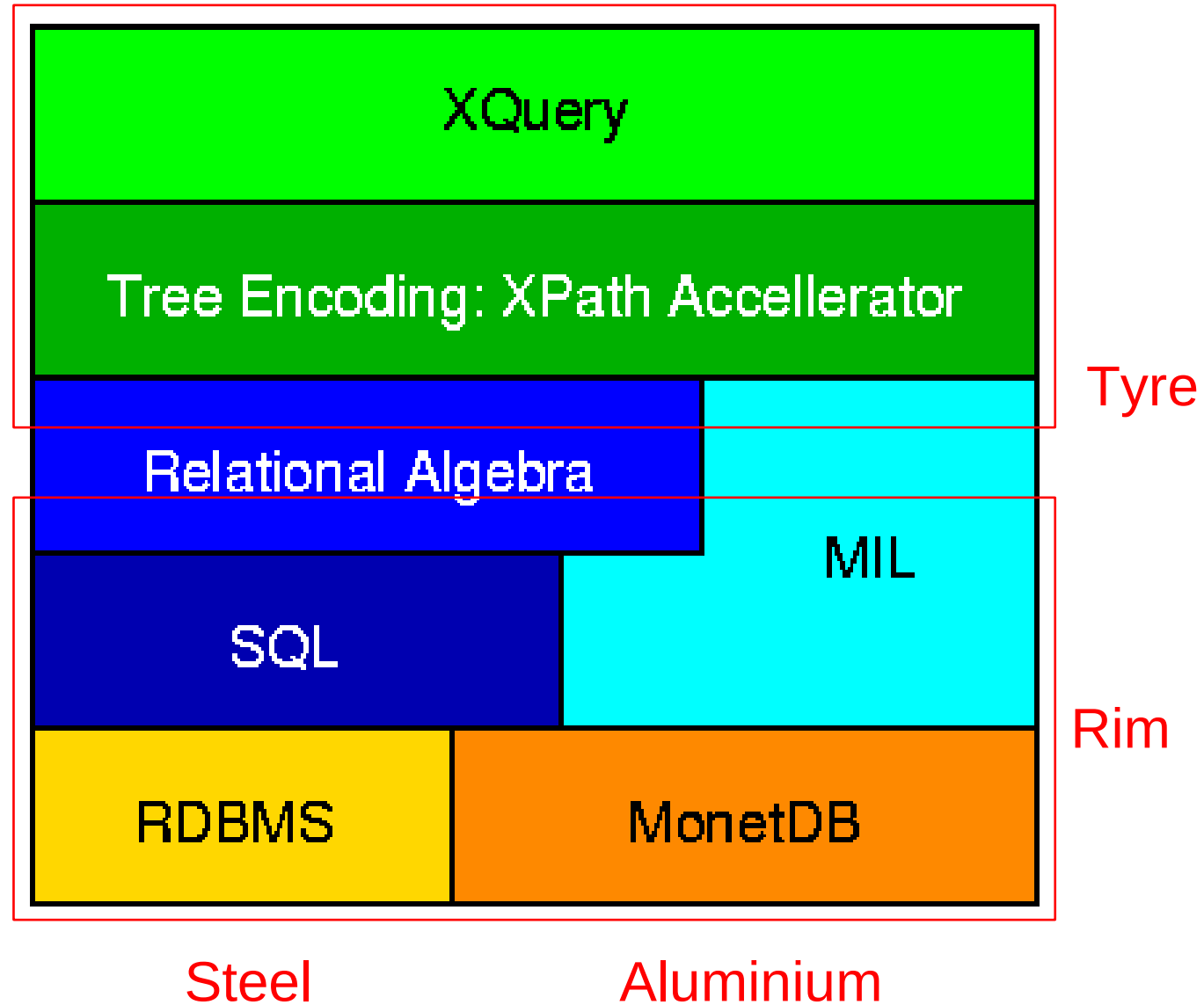
## Relational

Leverage RDBMS storage, query processing & optimization

- XML shredded into tables
- XQuery translated into SQL

*Let's use the old rim,  
but make a new tyre!*

# The Idea



# Storing XML in Relations: Schema-Based Approach

Idea: Use schema information (from a DTD) to find a suitable relational schema.

- ▶ Proposed, e. g., by Shanmugasundaram et al., 1999.
- ▶ Example:

```

<persons>
  <person>
    <name>John Doe</name>
    <address>
      <street>13 Main Street</street>
      <city>Miami, FL 12345</city>
    </address>
  </person>
  <person>
    <name>Martha Johnson</name>
    <address>
      <street>42 Kin
      <city>New York
    </address>
  </person>
</persons>

```



<i>id</i>	<i>name</i>	<i>street</i>	<i>city</i>
1	John Doe	13 Main Street	Miami, FL 12345
2	Martha Johnson	42 Kings Road	New York, NY 54321

# Schema-Based Approach: Benefits & Drawbacks

## Some benefits and drawbacks of schema-based storage:

- ⊕ Takes full advantage of specialized data types, indexes,...
- ⊖ Requires DTD/Schema information.
- ⊖ Many issues with XML/XQuery semantics:
  - ▶ XML reconstruction,
  - ▶ document order,
  - ▶ recursive XML documents,...

Schema-based approaches are thus mainly used

- ▶ for **data-centric** XML, or
- ▶ if **interoperation** with “relational” applications is required.

# MonetDB/XQuery

A Fast XQuery Processor Powered by a Relational Engine

Peter Boncz, Stefan Manegold  
(CWI Amsterdam)



Torsten Grust, Jens Teubner, Jan Rittinger  
(Technische Universität München)



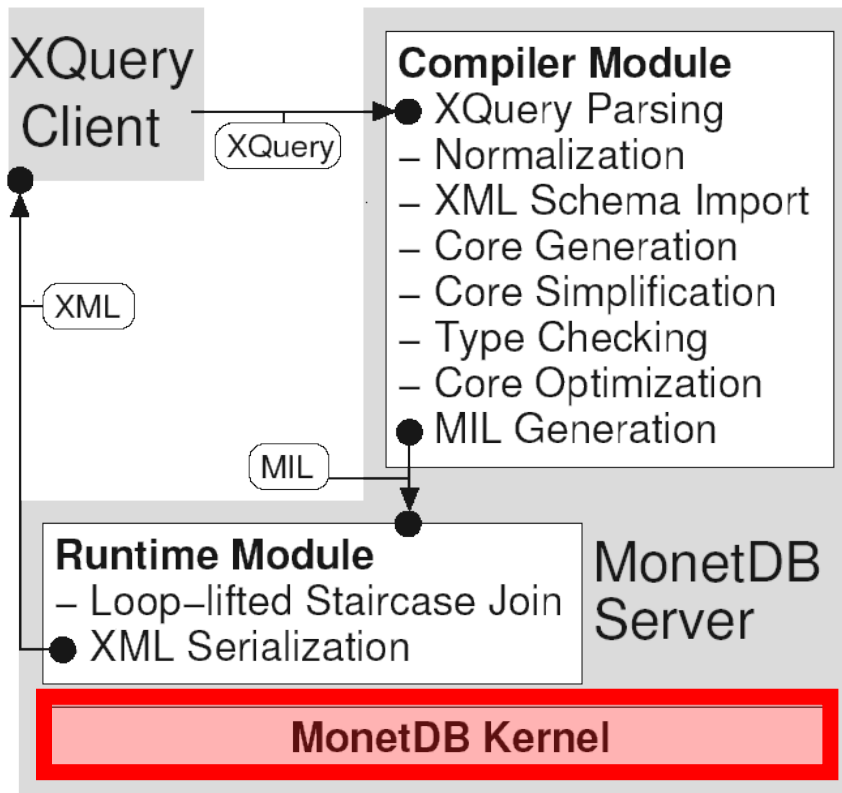
Maurice van Keulen  
(Technische Universiteit Twente)



[ACM/SIGMOD 2006]

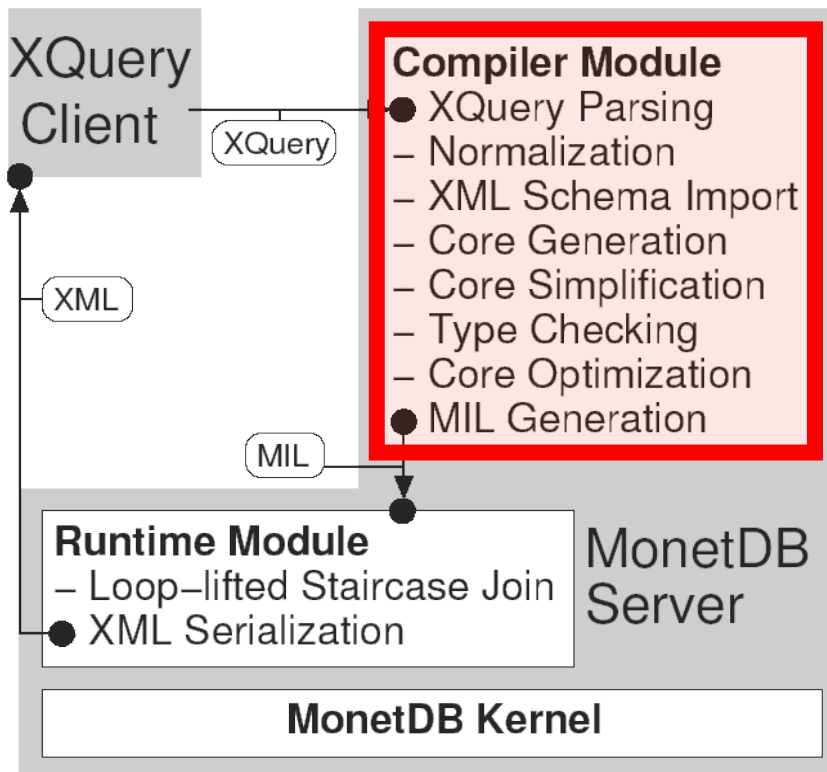
# MonetDB

open-source Mozilla license => download at [monetdb-xquery.org](http://monetdb-xquery.org)





# MonetDB/XQuery

open-source Mozilla license => download at [monetdb-xquery.org](http://monetdb-xquery.org)



## Pathfinder Project

 Torsten Grust, Jens Teubner, Jan Rittinger

 Maurice van Keulen

# Schema-Oblivious Storage: XPath Accelerator

There are schema-independent ways to store XML.

- ▶ **XPath accelerator** [Grust 2002]
- ▶ Advantages:
  - ▶ support for **all XPath axes**, from **any context node**,
  - ▶ low space overhead,
  - ▶ efficiently implementable.
- ▶ Idea:
  - ▶ The **sequence of start/end tags** in the serialized XML document implies the full tree information.
  - ▶ Encode this sequence in a **numbering scheme**.

# XPath Accelerator: *pre/post* plane

Node-based relational encoding of XQuery's data model

```
<a>  
  <b>  
    <c/>  
  </b>  
  <d/>  
  <e>  
    <f>  
      <g/>  
      <h/>  
    </f>  
    <i>  
      <j/>  
    </i>  
  </e>  
</a>
```

# XPath Accelerator: *pre/post* plane

Node-based relational encoding of XQuery's data model

```
0<a>  
  1<b>  
    2<c/>  
  </b>  
  3<d/>  
  4<e>  
    5<f>  
      6<g/>  
      7<h/>  
    </f>  
    8<i>  
      9<j/>  
    </i>  
  </e>  
</a>
```

# XPath Accelerator: *pre/post* plane

Node-based relational encoding of XQuery's data model

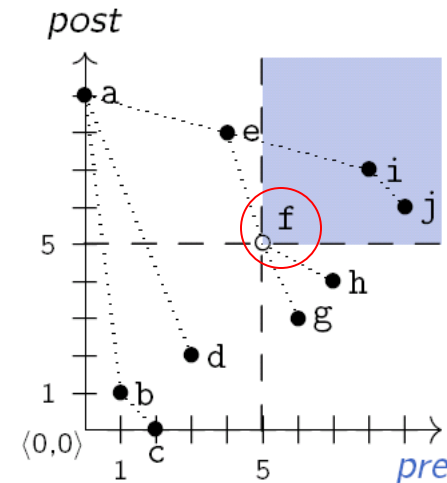
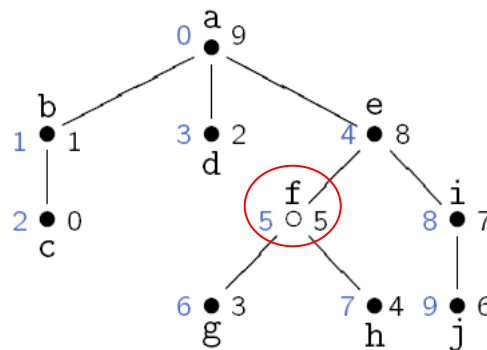
```
0<a>  
  1<b>  
    2<c/>0  
  </b>1  
  3<d/>2  
  4<e>  
    5<f>  
      6<g/>3  
      7<h/>4  
    </f>5  
    8<i>  
      9<j/>6  
    </i>7  
  </e>8  
</a>9
```

# XPath Accelerator: *pre/post* plane

Node-based relational encoding of XQuery's data model

```

0<a>
  1<b>
    2<c/>0
  </b>1
  3<d/>2
  4<e>
    5<f>
      6<g/>3
      7<h/>4
    </f>5
    8<i>
      9<j/>6
    </i>7
  </e>8
</a>9
  
```



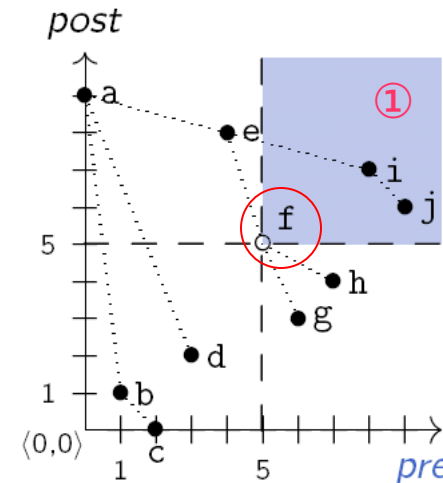
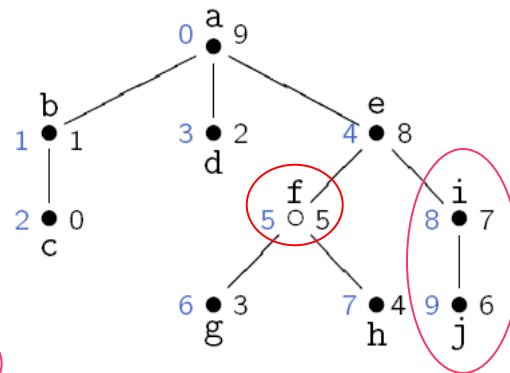
pre	post
0	9
1	1
2	0
3	2
4	8
5	5
6	3
7	4
8	7
9	6

# XPath Accelerator: *pre/post* plane

Node-based relational encoding of XQuery's data model

```

0<a>
  1<b>
    2<c/>0
  </b>1
  3<d/>2
  4<e>
    5<f>
      6<g/>3
      7<h/>4
    </f>5
    8<i>
      9<j/>6
    </i>7
  </e>8
</a>9
  
```



pre	post
0	9
1	1
2	0
3	2
4	8
5	5
6	3
7	4
8	7
9	6

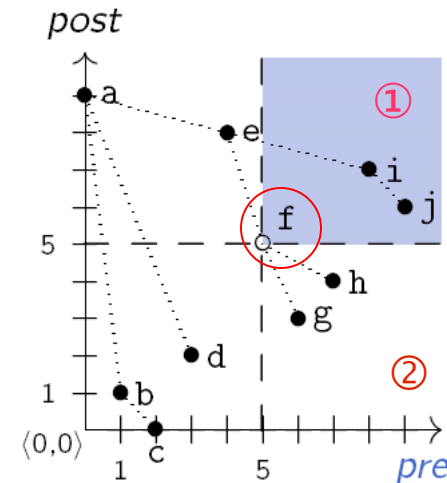
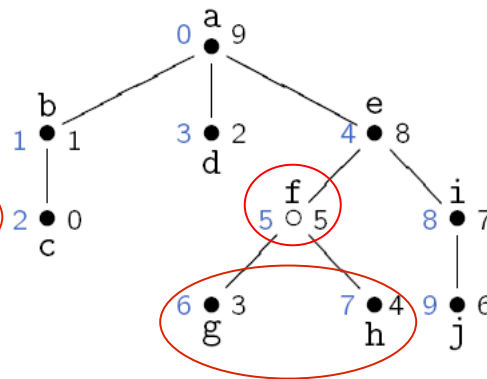
① **f/following:** `SELECT * FROM pre_post WHERE pre > f.pre AND post > f.post`

# XPath Accelerator: *pre/post* plane

Node-based relational encoding of XQuery's data model

```

0<a>
  1<b>
    2<c/>0
  </b>1
  3<d/>2
  4<e>
    5<f>
      6<g/>3
      7<h/>4
    </f>5
  8<i>
    9<j/>6
  </i>7
</e>8
</a>9
  
```



pre	post
0	9
1	1
2	0
3	2
4	8
5	5
6	3
7	4
8	7
9	6

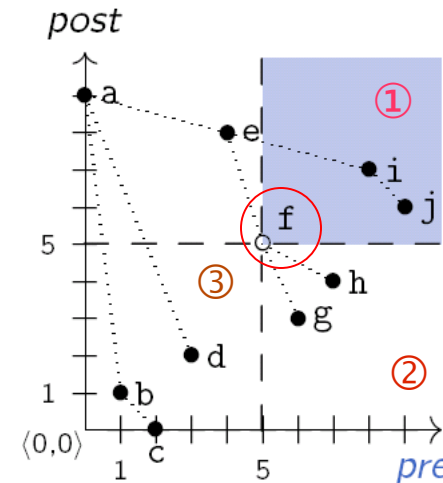
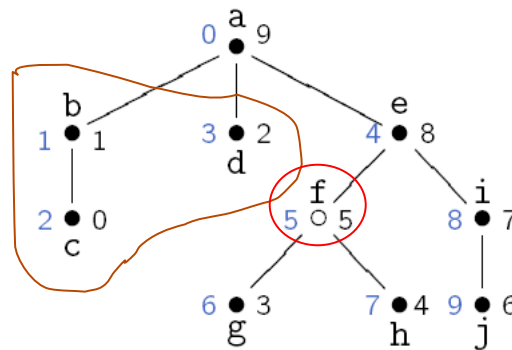
- ① **f/following:** SELECT \* FROM pre\_post WHERE pre > f.pre AND post > f.post
- ② **f/descendant:** SELECT \* FROM pre\_post WHERE pre > f.pre AND post < f.post

# XPath Accelerator: *pre/post* plane

Node-based relational encoding of XQuery's data model

```

0<a>
  1<b>
    2<c/>0
  </b>1
  3<d/>2
  4<e>
    5<f>
      6<g/>3
      7<h/>4
    </f>5
    8<i>
      9<j/>6
    </i>7
  </e>8
</a>9
  
```

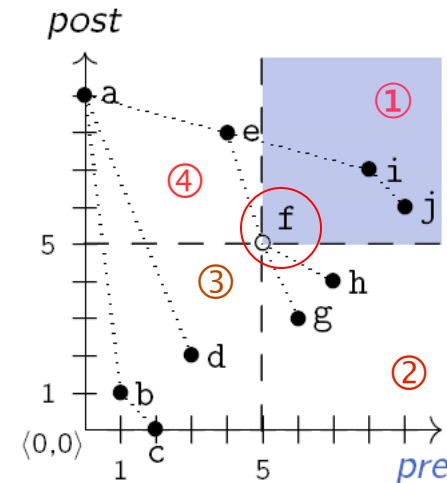
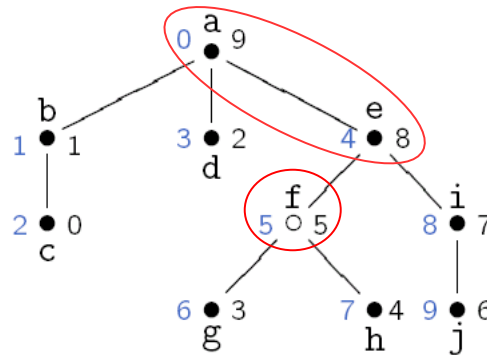
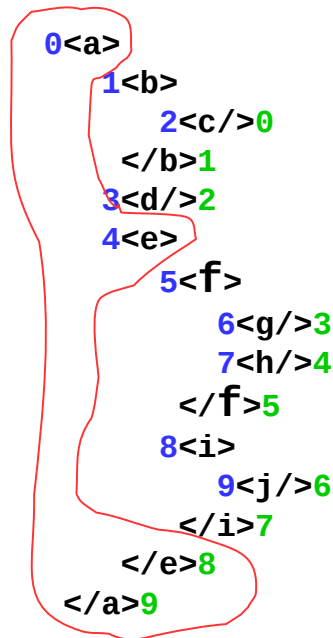


pre	post
0	9
1	1
2	0
3	2
4	8
5	5
6	3
7	4
8	7
9	6

- ① **f/following:** `SELECT * FROM pre_post WHERE pre > f.pre AND post > f.post`
- ② **f/descendant:** `SELECT * FROM pre_post WHERE pre > f.pre AND post < f.post`
- ③ **f/preceding:** `SELECT * FROM pre_post WHERE pre < f.pre AND post < f.post`

# XPath Accelerator: *pre/post* plane

## Node-based relational encoding of XQuery's data model



pre	post
0	9
1	1
2	0
3	2
4	8
5	5
6	3
7	4
8	7
9	6

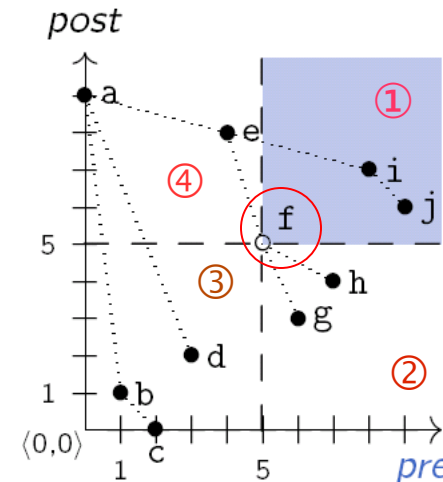
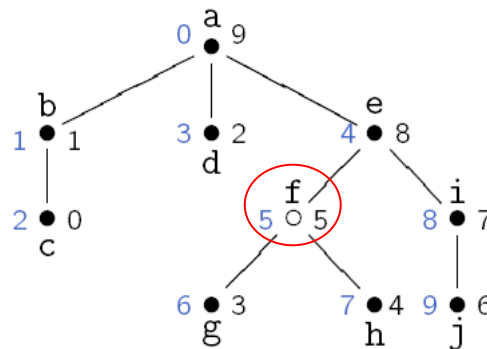
- ① **f/following:** `SELECT * FROM pre_post WHERE pre > f.pre AND post > f.post`
- ② **f/descendant:** `SELECT * FROM pre_post WHERE pre > f.pre AND post < f.post`
- ③ **f/preceding:** `SELECT * FROM pre_post WHERE pre < f.pre AND post < f.post`
- ④ **f/ancestor:** `SELECT * FROM pre_post WHERE pre < f.pre AND post > f.post`

# XPath Accelerator: *pre/post* plane

Node-based relational encoding of XQuery's data model

```

0<a>
  1<b>
    2<c/>0
  </b>1
  3<d/>2
  4<e>
    5<f>
      6<g/>3
      7<h/>4
    </f>5
    8<i>
      9<j/>6
    </i>7
  </e>8
</a>9
  
```



pre	post
0	9
1	1
2	0
3	2
4	8
5	5
6	3
7	4
8	7
9	6

- ① **f/following:** `SELECT * FROM pre_post WHERE pre > f.pre AND post > f.post`
- ② **f/descendant:** `SELECT * FROM pre_post WHERE pre > f.pre AND post < f.post`
- ③ **f/preceding:** `SELECT * FROM pre_post WHERE pre < f.pre AND post < f.post`
- ④ **f/ancestor:** `SELECT * FROM pre_post WHERE pre < f.pre AND post > f.post`

Similar queries for other XPath axes

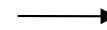
# pre/post Table & pre/size/level Table

```

<a>
  <b>
    <c/>
  </b>
  <d/>
  <e>
    <f>
      <g/>
      <h/>
    </f>
    <i>
      <j/>
    </i>
  </e>
</a>

```

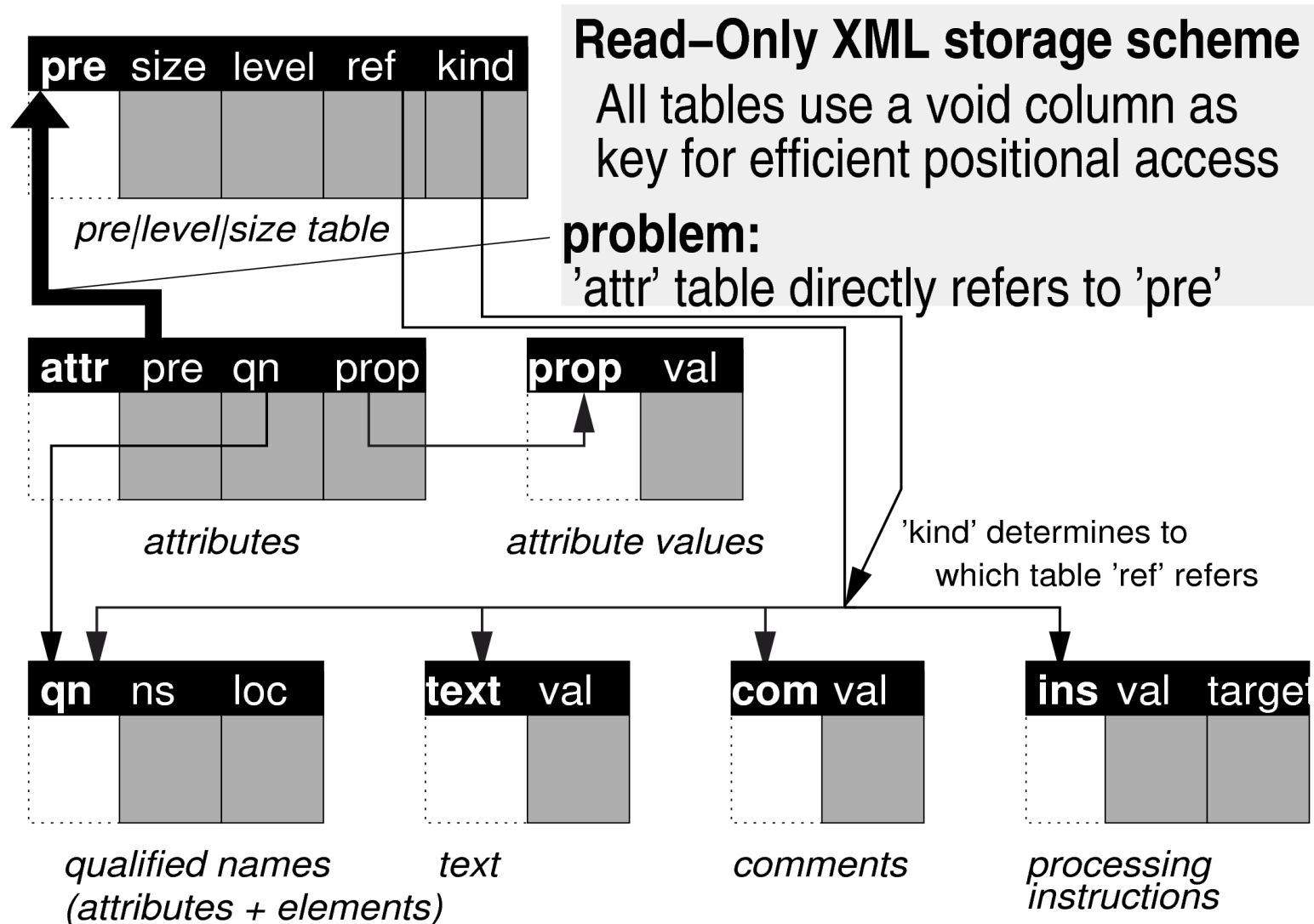
	Pre	Post
a	0	9
b	1	1
c	2	0
d	3	2
e	4	8
f	5	5
g	6	3
h	7	4
i	8	7
j	9	6



Pre	Size	Level
0	9	0
1	1	1
2	0	2
3	0	1
4	5	1
5	2	2
6	0	3
7	0	3
8	1	2
9	0	3

$$\text{Post} = \text{pre} + \text{size} - \text{level}$$

# Complete XML Storage Schema



# XPath evaluation (SQL)

Example query:

`/descendant::open_auction[./bidder]/annotation`

```

SELECT DISTINCT a.pre
  FROM doc r, doc oa, doc b, doc a
 WHERE r.pre=0
       AND oa.pre > r.pre AND oa.post < r.post      <- descendant
       AND oa.name = "open_auction" AND oa.kind = "elem"
       AND b.pre > oa.pre AND b.post < oa.post      } child
       AND b.level = oa.level + 1
       AND b.name = "bidder" AND b.kind < "elem"
       AND a.pre > oa.pre AND a.post < oa.post      } child
       AND a.level = oa.level + 1
       AND a.name = "annotation" AND a.kind = "elem"
 ORDER BY a.pre

```

# XQuery Systems: 2 Approaches

## Native

Tree is basic data structure

- tree-storage manager
- tree-query processing (algebra)
- tree-query optimization

Re-inventing the *wheel*?

X-Hive

Timber

DB2

BDB-XML

Galax

Microsoft

Oracle

MonetDB/  
XQuery

## Relational

Leverage RDBMS storage, query processing & optimization

- XML shredded into tables
- XQuery translated into SQL

*Let's use the old rim,  
but make a new tyre!*

# Oracle10g/11g: XQuery Support

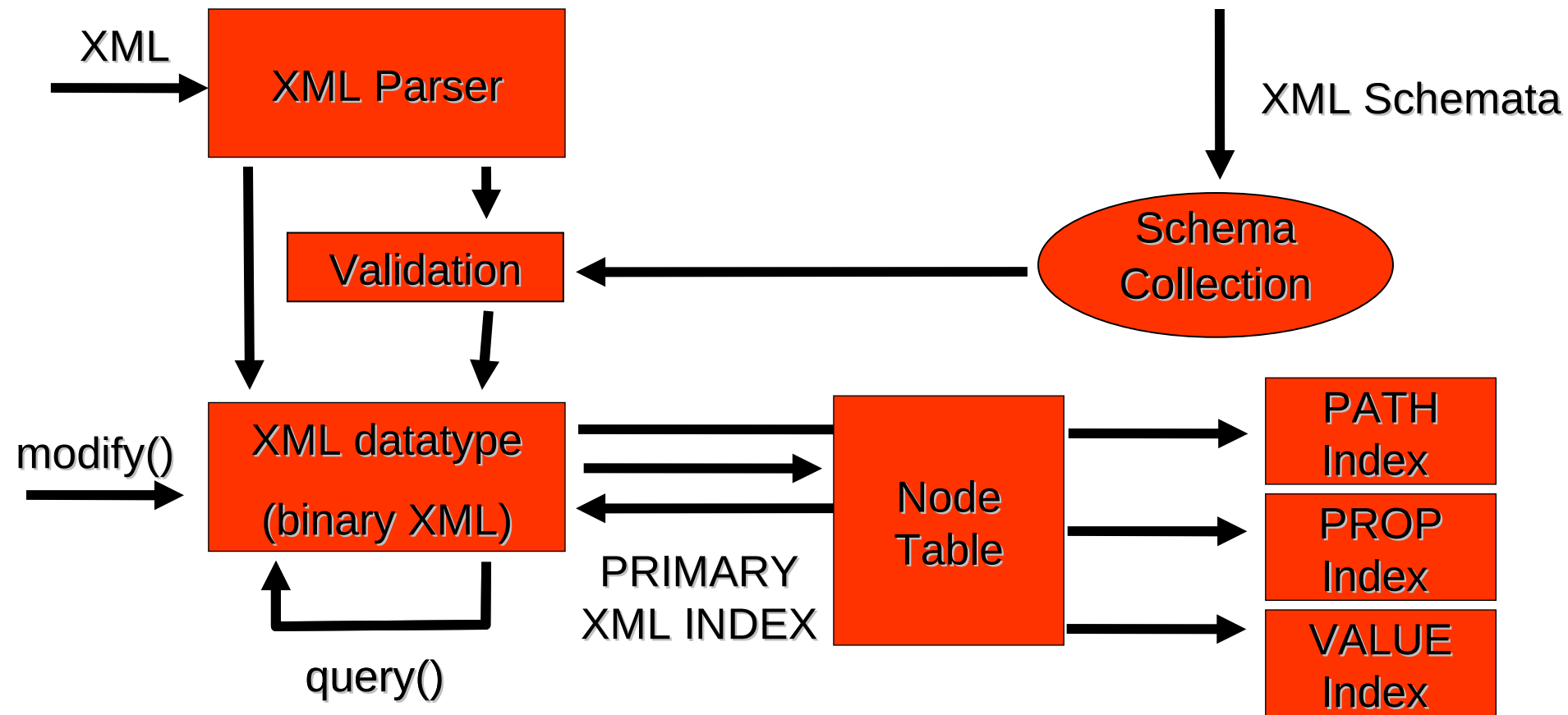
1. XMLDB integrated database engine
  - SQL / XML standard support
  - Optimized queries – rewrite to relational
2. Standalone Java query engine
  - 100% Java
  - Integrated into Oracle App Server -XDS
  - Interoperates with XSLT/XPath

First relational database to ship  
an XQuery implementation !

# Oracle 10g/11g: XQuery database support

- Production in Oracle Database 10gr2
- Supports XMLQuery and XMLTable construct
- Native compilation into SQL /XML structures
- Returns XMLType(Content)
- Can query over relational, O-R, XMLType data
- fn:doc - Maps to XDB Repository on server
- SQLPlus provides *xquery* command to execute XQuery
- XSL-T will also get compiled to XQuery

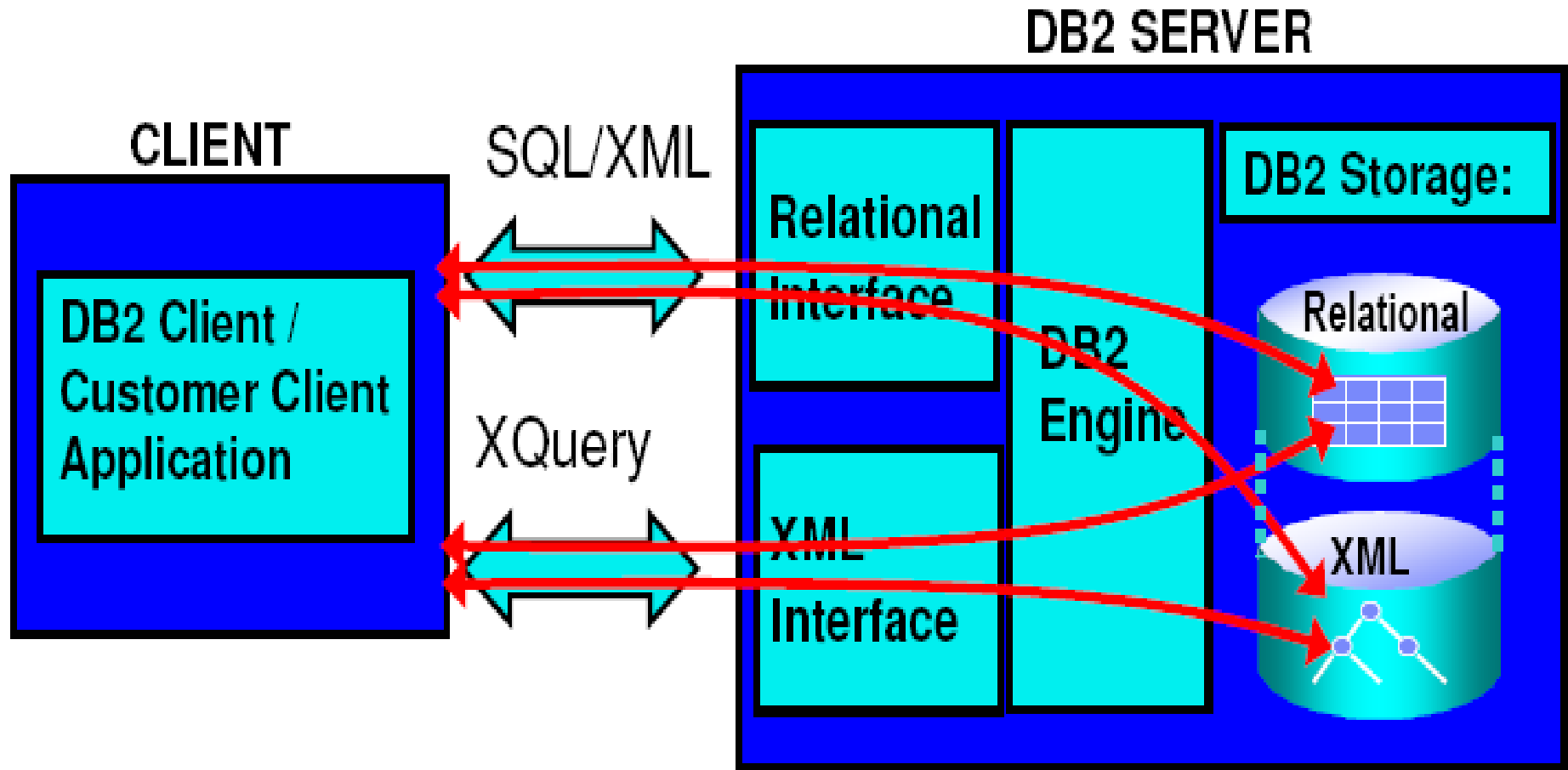
# Microsoft SQL Server 2005: Overview



# Microsoft SQL Server 2005: Indexing

- Create XML index on XML column
  - `CREATE PRIMARY XML INDEX idx_1 ON docs (xDoc)`
- Creates secondary indexes on tags, values, paths
- Speeds up queries
  - Results can be served directly from index
  - Entire query is optimized
    - ▶ Same award winning cost based optimizer
  - Indexes are used as available

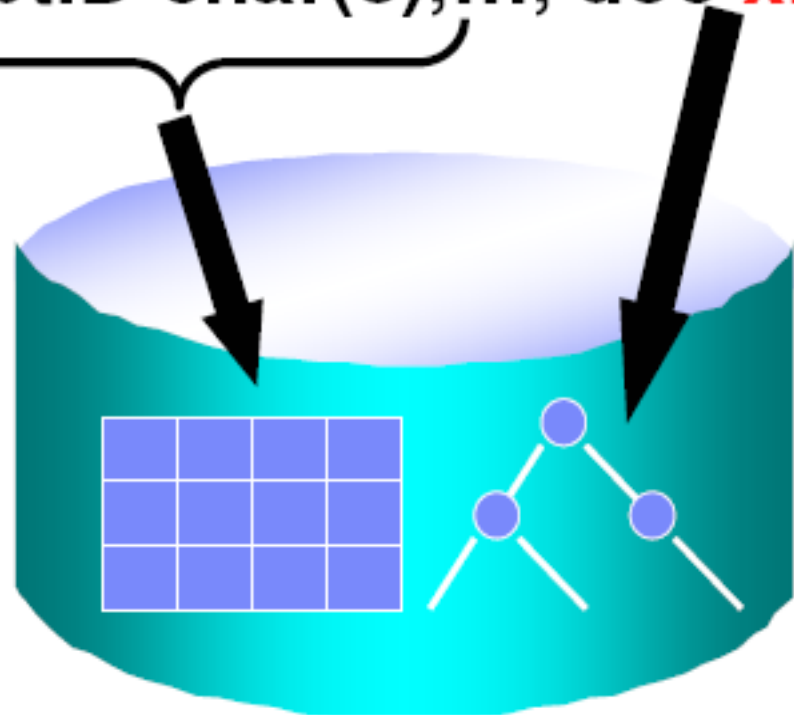
# IBM DB2 v9: Overview



# IBM DB2 v9: Storage

```
create table dept (deptID char(8),..., doc xml);
```

- Relational columns are stored in relational format
- XML columns are stored **natively**
- All XML data is stored in XML-typed columns



# IBM DB2 v9: SQL in XQuery

## XQuery access to XML column values

Create table dept(docid int, doc xml)

### ■ Identifying XML data by column

Always operates on entire column!

FOR \$d in `db2-fn:xmlcolumn("DEPT.DOC")`...

### ■ Identifying XML data via a select statement

Leverage predicates/indexes on relational columns

FOR \$d in `db2-fn:sqlquery("select doc from dept")`...

FOR \$d in `db2-fn:sqlquery("select doc from dept where deptID LIKE 'PR%' ")`...

FOR \$d in `db2-fn:sqlquery("select dept.doc from dept, unit  
where dept.deptID=unit.ID and unit.headcount > 200")`...

# IBM DB2 v9: XQuery in SQL

## XMLQUERY: embed XQuery in SQL

```
create table dept(deptID char(8) primary key, doc xml);
create table unit(unitID char(8), headcount integer, bldg integer);
```

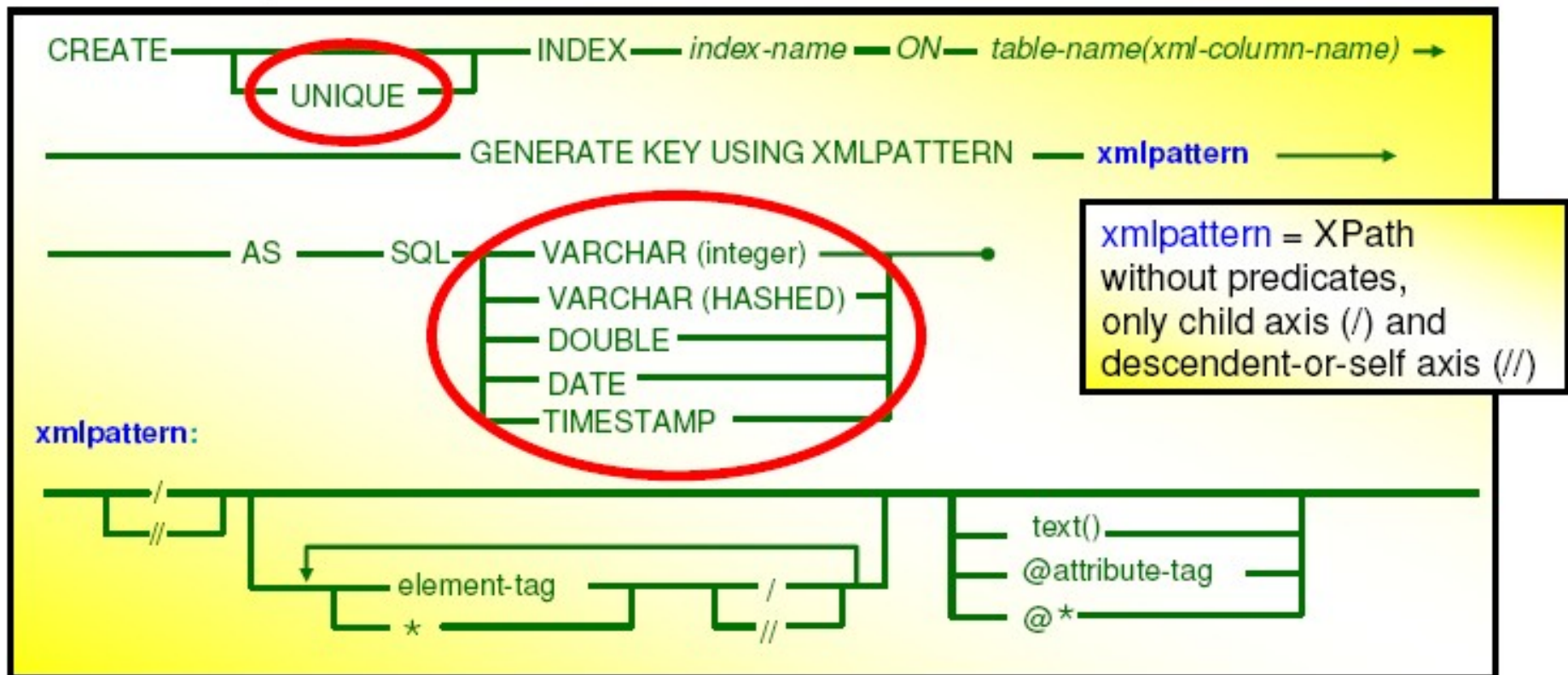
```
select deptID,xmlquery('for $d in $deptdoc/dept
                        where $d/@bldg = 101
                        return $d/name' passing doc as "deptdoc")
from dept
where deptID <> "PR27";
```

```
select d.deptID , u.headcount,
       xmlquery('$deptdoc/dept/name' passing d.doc as "deptdoc")
from dept d, unit u
where d.deptID=u.unitID
and u.headcount > 200
and xmlexists('$deptdoc/dept/employee/name' passing d.doc as "deptdoc")
```

# IBM DB2 v9: Indexing

## XML Index DDL

- create index **idx1** on **T(xmlcol)**  
generate key using **xmlpattern** **'/a/b/@c'**  
as sql date



- Declaration & use of namespace prefix supported (not shown above)