



i MonetDB/XQuery

Georgina Ramírez i Camps
CWI, Amsterdam

(Peter Boncz and Stefan Manegold)



Outline

- MonetDB
 - Introduction: What, Why, Where, Who?
 - How? Main ideas
 - Some notes on performance, portability and usage
- MonetDB/XQuery
 - Introduction: W3C, XML, XPath, XQuery
 - XQuery in a RDBMS: main ideas and some optimizations
 - Performance comparison
- Conclusions



MonetDB

**A next-generation database kernel
for query-intensive applications**



<http://monetdb.cwi.nl>

<http://sourceforge.net/projects/monetdb>



What?

- Main-memory based DBMS backend/kernel
 - Developed at CWI since 1992
- Specially designed for “query-intensive” applications;
 - Data mining
 - Data warehousing / decision support
 - Multi-media information retrieval (text, images, video, audio, ...)
 - XML databases
 - GIS



Why?

- Relational DBMS dominates the scene
 - Oracle, SQLserver, DB2
- Databases a solved problem?



Why?

- Are relational DBMS fit for the job?
 - Developed in end 1970's begin 1980's



Why?

- Are relational DBMS fit for the job?
 - Developed in end 1970's begin 1980's
- Hardware has changed
 - CPUs get faster but more vulnerable
 - Capacity and bandwidth increase rapidly
 - Bottleneck are I/O **and** RAM



Why?

- Are relational DBMS fit for the job?
 - Developed in end 1970's begin 1980's
- Hardware has changed
 - CPUs get faster but more vulnerable
 - Capacity and bandwidth increase rapidly
 - Bottleneck are I/O **and** RAM
- Applications have changed
 - RDBMS tuned for transaction processing
 - Not query-intensive
 - Only business domain



Transactions (OLTP)

[illegible]

```
find client
10032112
```

OLTP queries:
access all
columns of
just one row.



Where?

- CWI (Centre for Mathematics and Computer Science).
- Dutch national research institute
- Established in 1946
- Organization:
 - 4 “clusters”
 - 3-6 “themes” per cluster
- ca. 220 employees:
 - ~20 profs., ~40 postdocs, ~50 support



Who?

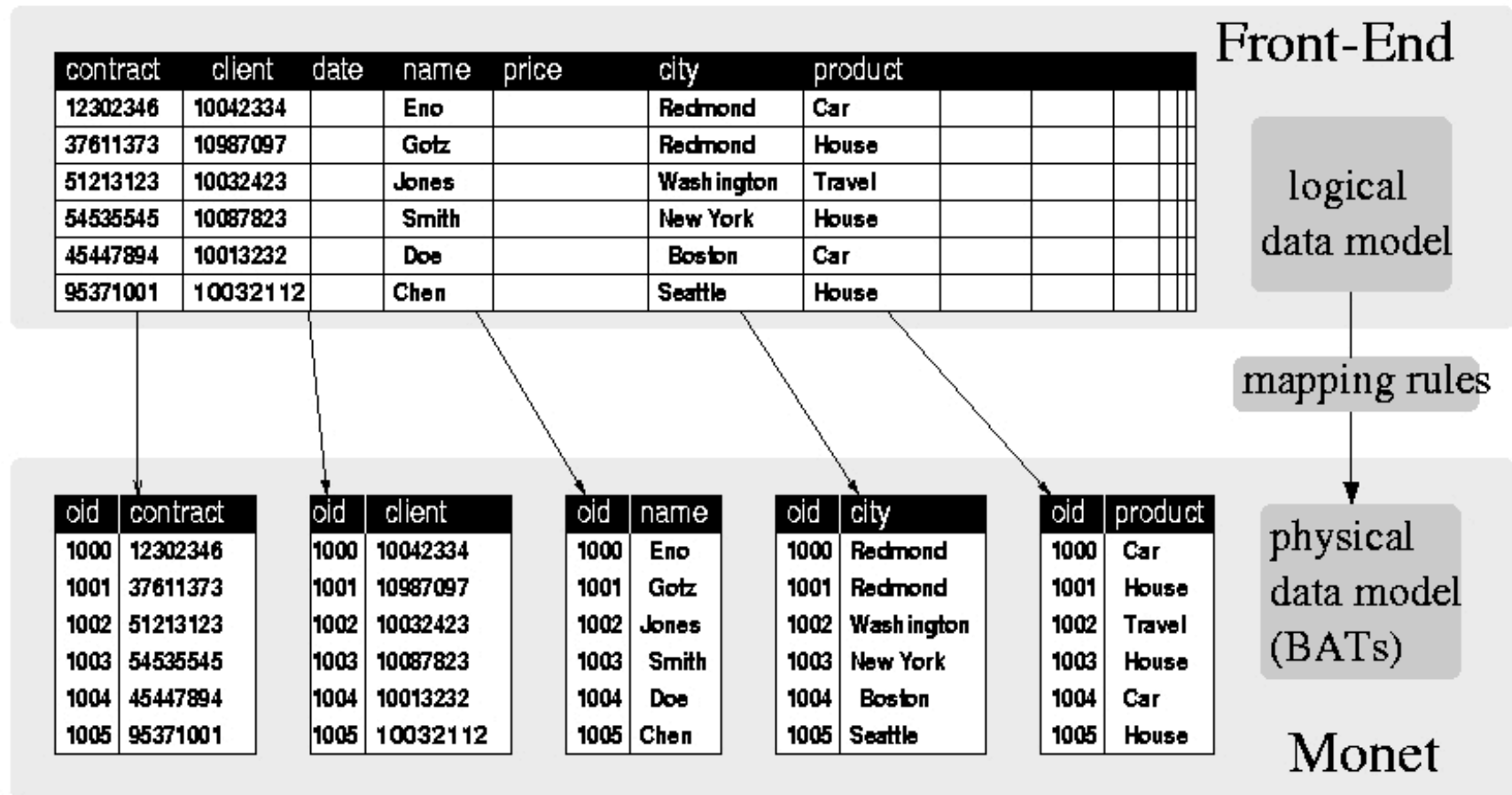
Contributors

Martin Kersten	1993-
Peter Boncz	1994-
Niels Nes	1996-
Stefan Manegold	1998-
Sjoerd Mullender	2002-
Arjen de Vries	1998-
Tim Ruhl	1999-
Menzo Windhouwer	1998-
Alex van Ballegooij	2000-
Johan List	2001-
Georgina Ramirez	2003-
Marcin Zukowski	2004-
Fabian Groffen	2004-
Steffen Goeldner	2004-
Torsten Grust	2003-
Jens Teubner	2003-
Maurice van Keulen	2003-
Jan Flokstra	2003-
Jan Rittinger	2004-
Wouter Alink	2005-
Jennie Zhang	2005-
Florian Waas	1997-2001
Albrecht Schmidt	1999-2003
Jonas Karlsson	1997-2001
Martin van Dinther	2002
Peter Bosch	2000-2001
Carel van den Berg	1994-1998
Wilco Quak	1994-2000

Monet Hacker, architect Monet3+5, Mknife
Monet hacker, architect Monet4
Monet Hacker, SQL front end
Monet Hacker, Test Web, Calibrator
64-bits porting, ODBC development
 Various Multimedia Retrieval Modules
 bug fixing at Data Distilleries
 URL, crawlers, Saxophone, Zeep(Sop)
 RAM front end (array query language)
 XML storage and retrieval
 multimedia retrieval (BOND module)
 SMP parallelism
 JDBC interface
 The Perl API
 XQuery compiler, semantics
 XQuery compiler, semantics
 XQuery compiler implementation
 XQuery MIL code generation
 XQuery MIL code generation
 XQuery web site
 SOAP
 dynamic loadable modules
 XML storage
 SDDs, dynamic loadable modules
 ODBC driver
 quad-trees
 ODMG front end
 GIS modules



Storing Relations in MonetDB



BAT Storage Optimizations

Dense ascending
sequence

oid	str
100	married
101	single
102	married
103	divorced
104	married
105	widow

logical BAT

type=VOID
seqbase=100

column
descriptor

type=enum1[x]

BUN Heap

0
1
0
2
0
3

underlying data structure

encoding
BAT 'x'

chr	str
0	married
1	single
2	divorced
3	widow

BAT Storage Optimizations

Void (virtual-oid) columns

- Contain dense sequence 0,1,2,3,4,...
- Require no space
- Positional access (nice for XPath skipping)
 - pre = void

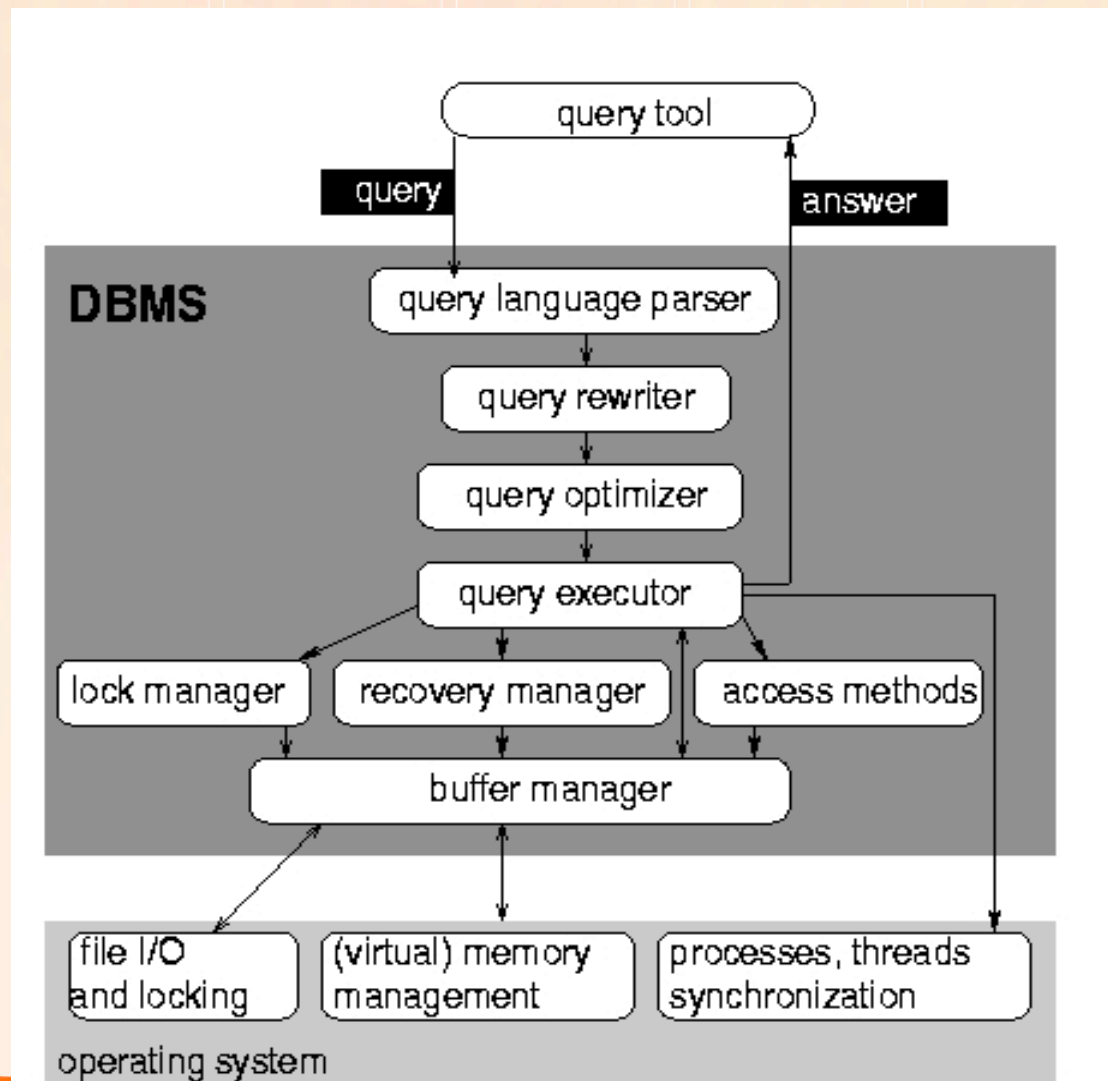


How?

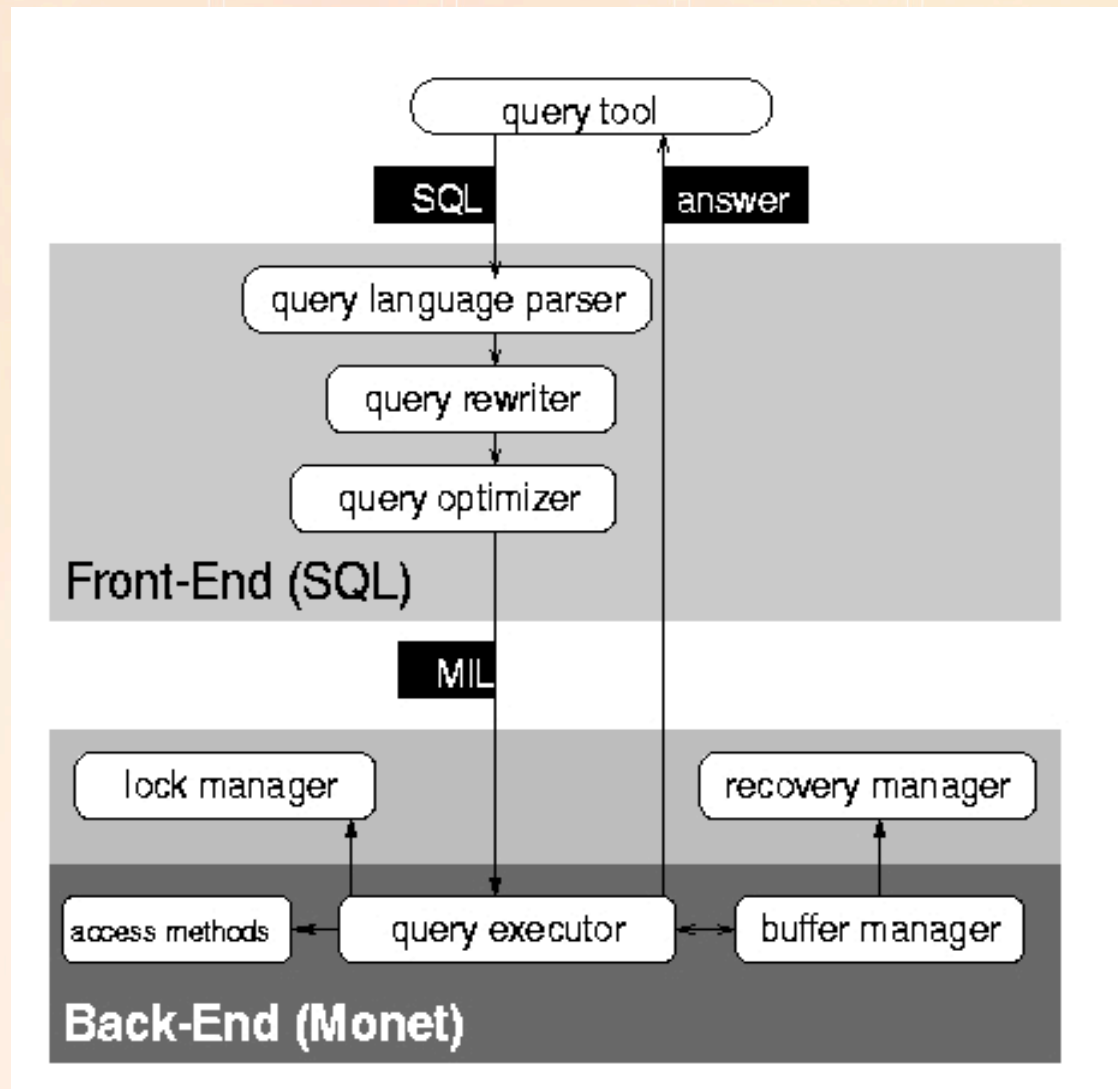
- full vertical fragmentation: always!
 - everything in binary (2-column) tables: BAT [oid, X]
 - don't touch what you don't need!
- the RISC approach to databases
 - simple data model, simple query language (MIL)
 - don't need (to pay for) a buffer manager 🏠🧠 manage virtual memory
 - explicit transaction management 🏠🧠 DIY approach to ACID
- CPU and memory cache optimized
 - programming team experienced in main memory DBMS techniques
 - use of scientific programming optimizations



DBMS Architecture



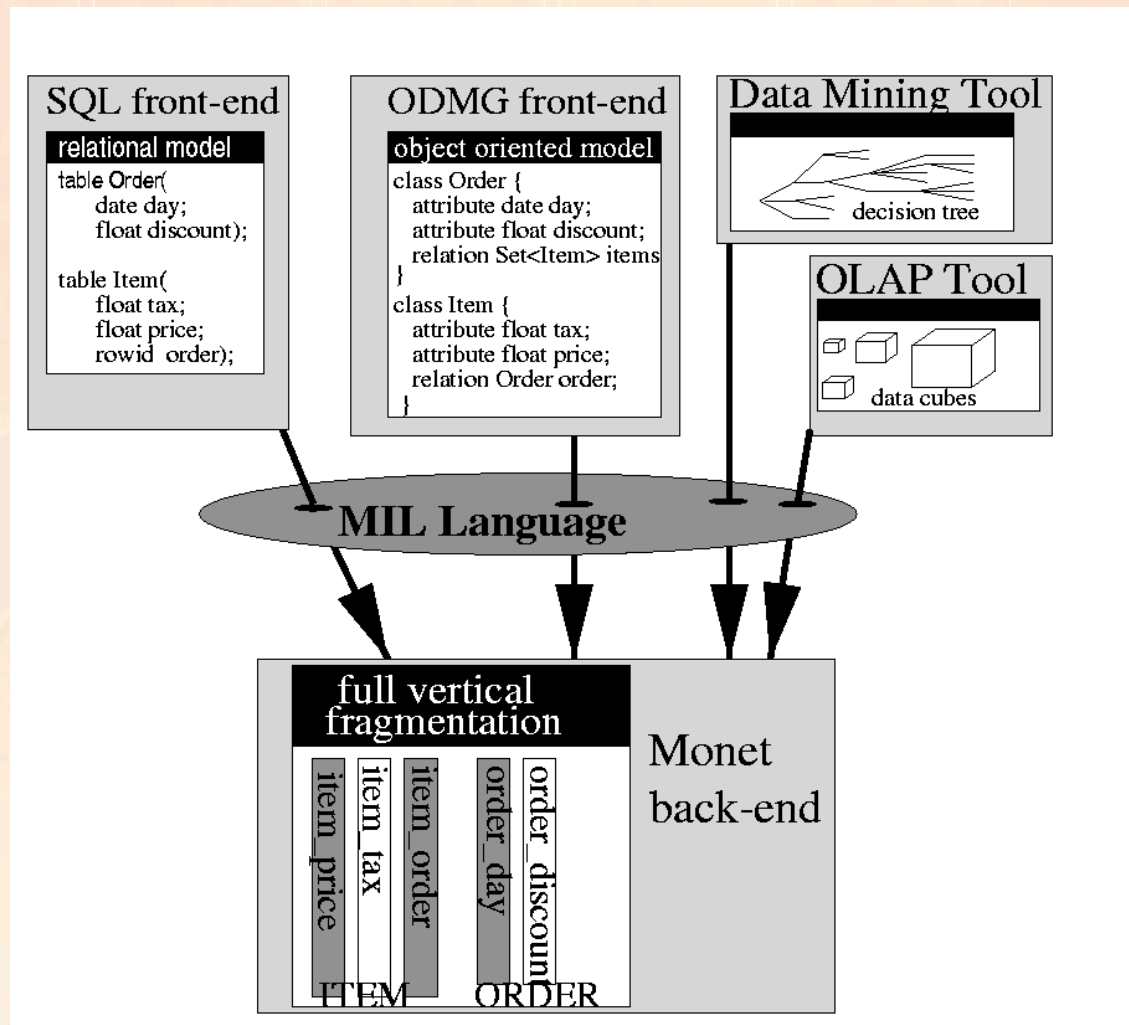
Monet: DBMS Microkernel



MonetDB: extensible architecture

Front-end/back-end:

- support multiple data models
- support multiple end-user languages
- support diverse application domains

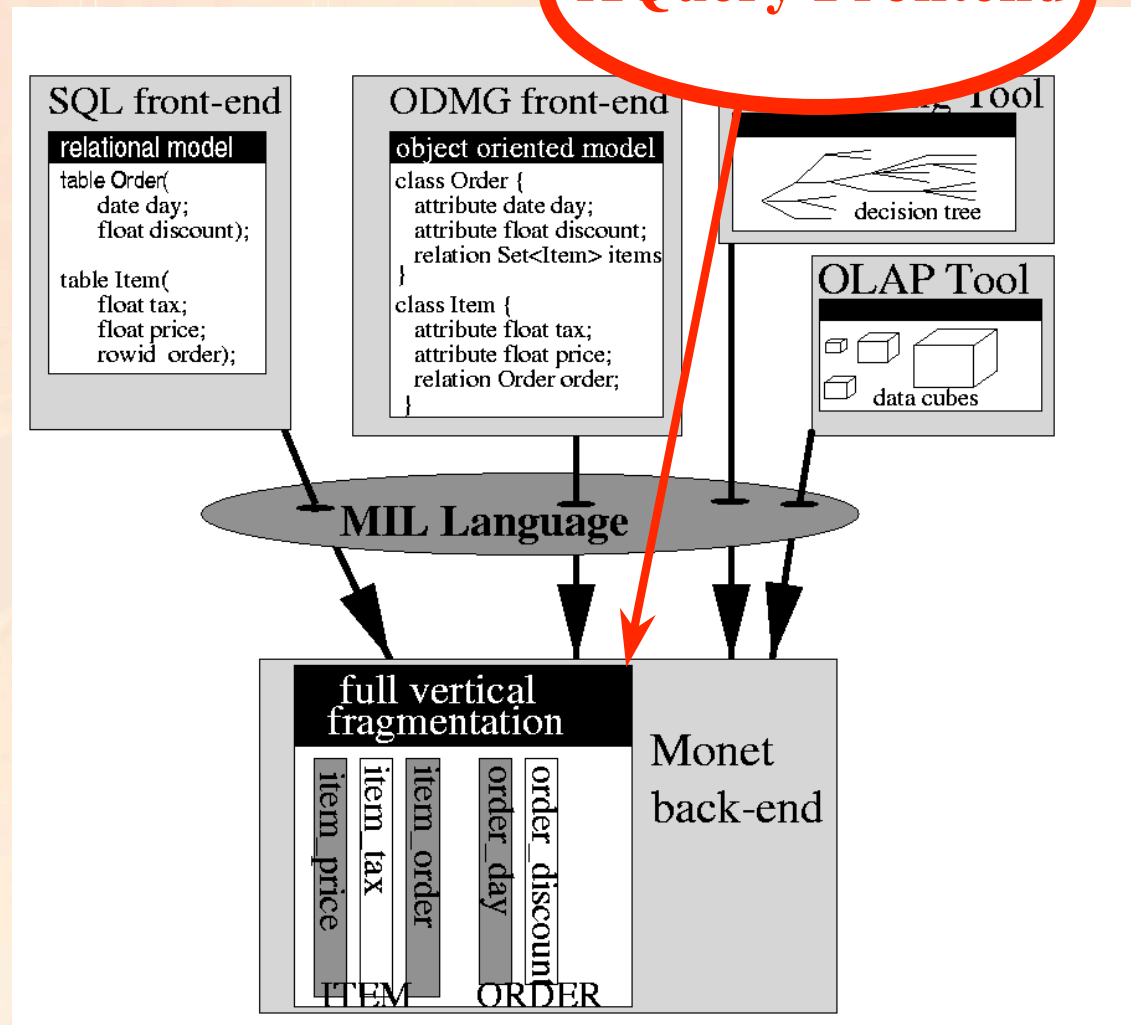


MonetDB: extensible architecture

XQuery Frontend

Front-end/back-end:

- support multiple data models
- support multiple end-user languages
- support diverse application domains



Performance: scalability

- **Create large binary table from 1k “seed”**
 - Permutate tail and insert recursively
- **SQL script of up to 50k lines**
- **Athlon 1400 Mhz, 1 GB RAM, 3.5 GB free disk space**

	MonetDB/SQL	MySQL 4.0.18	PostgreSQL 7.4	SQLite 2.8.12
1K	0.2 s	0.2 s	2.0 s	5.0 s
100K	0.9 s	12.1 s	69.6 s	30.7 s
1M	6.1 s	122.9 s	1060.4 s	335.2 s
10M	59.9 s	1582.7 s	4377.0 s	> 7200.0 s
20M	124.6 s	3190.6 s	full disk	aborted
30M	295.2 s	4740.6 s		



Performance: C't magazine (issue 13/2006)

Benchmark-Ergebnisse

Einsender	Betriebs-system	Datenbank	Programmiersprache	Messergebnis (opm)
MySQL Benchmark Team	Linux	MySQL 5	PHP 5	3664
Alexander Burger	Linux	Pico Lisp 2.2.1.pre	Pico Lisp 2.2.1.pre	2600 ¹
MonetDB-Entwicklerteam	Linux	MonetDB/SQL 4.9	Java	1833
Jan Mönnich	Windows	MySQL 5	Java	1798
USU ValueCoders	Windows	MySQL 4.1	Java	1564
Martin Winandy	Windows	MySQL 4.1	PHP 5	1542
IBM	Linux	DB2 Express 8.2	Java	1537
Frank Berger	Linux	Oracle 10g Express	Java	1412
Niklas Mehner	Linux	SiebenGeisslein	Java	600 ¹
exedio	Linux	MySQL 5	Java	587
FH Ulm	Windows	MySQL 5	PHP 5	444 ¹
Gerhard Kalab	Windows	MySQL 5	Python 2.4	137
bluegap	Linux	PostgreSQL 8.1	PHP 5	120
tdb Software	Linux	tdbengine	EASY	— ¹
Tommi Mäkitalo	Linux	MySQL 4.1	C++	— ¹
Alvar C.H. Freude	Linux	PostgreSQL 8.x	Perl	— ¹
Rolf Stakemann	Windows	MySQL 3.23	PHP 4	— ¹

¹ siehe Text



Portability & (nightly) Testing

OS		CPU	MHz	RAM [MB]	Bits	Compiler
1	Windows2000	Pentium III	450	256	32	Microsoft
2	Cygwin 1.5.7 (on WindowsXP)	Pentium III	550	256	32	GNU
3	RedHat 9 (Linux 2.4.20)	Athlon XP	1400	1,024	32	GNU
4	RedHat 9 (Linux 2.4.20)	Athlon XP	1400	1,024	32	Intel
5	SuSE 8.1 (Linux 2.4.21)	Pentium 4 Xeon	2200	2,048	32	GNU
6	RedHat 2.1AW (Linux 2.4.18)	Itanium 2	1300	512	64	GNU
7	RedHat 2.1AW (Linux 2.4.18)	Itanium 2	1300	512	64	Intel
8	Debian 3.0 (Linux 2.4.24)	Opteron	1600	16,384	32 & 64	GNU
9	Debian 3.0 (Linux 2.4.24)	Opteron	1600	16,384	32	Intel
10	Debian 3.0 (Linux 2.4.24)	Opteron	1600	16,384	32 & 64	PGI
11	MacOS X 10.3 (Darwin 7.3.0)	PowerPC G4	733	1,024	32	GNU
12	AIX 5.1	Power4	1100	32,768	32 & 64	GNU
13	AIX 5.1	Power4	1100	32,768	32 & 64	IBM
14	IRIX64 6.5	MIPS R10k	250	65,536	32 & 64	GNU
15	IRIX64 6.5	MIPS R12k	300	65,536	32 & 64	SGI
16	Solaris 8 (SunOS 5.8)	UltraSPARC II	450	4,096	32 & 64	GNU
17	Solaris 8 (SunOS 5.8)	UltraSPARC II	450	4,096	32 & 64	Sun
18	OpenZaurus 3.2 (Linux 2.4.6)	StrongARM-1110	206	64	32	GNU (cross)



MonetDB: Usage

- a decade of “query-intensive” application experience
 - *image retrieval*: Peter Bosch → ImageSpotter
 - *audio/video retrieval*: Alex van Ballegooij → RAM
 - *XML text retrieval*: de Vries / Hiemstra → TIJAH
 - *XML databases*: Albrecht Schmidt → XMARK
 - Grust / vKeulen → Pathfinder
 - *GIS*: Wilco Quak → MAGNUM
 - *data warehousing / OLAP / data mining*
 - SPSS → DataDistilleries
 - Univ. Massachussetts → PROXIMITY

MonetDB / XQuery

*Using relational technology
to query XML documents*



*Technical University Munich
(Torsten Grust, Jens Teubner, Jan Rittinger)*



*Technical University Twente
(Maurice van Keulen, Jan Flokstra)*



*Center for Mathematics and Computer Science
Amsterdam
(Peter Boncz, Stefan Manegold, Sjoerd Mullender)*



Background

- W3C
- XML
- XPath
- XQuery



W3C (<http://www.w3.org>)

The World Wide Web Consortium (W3C) is an international consortium where Member organizations, a full-time staff, and the public work together to develop Web standards.

W3C Mission:

To lead the World Wide Web to its full potential by developing protocols and guidelines that ensure long-term growth for the Web.

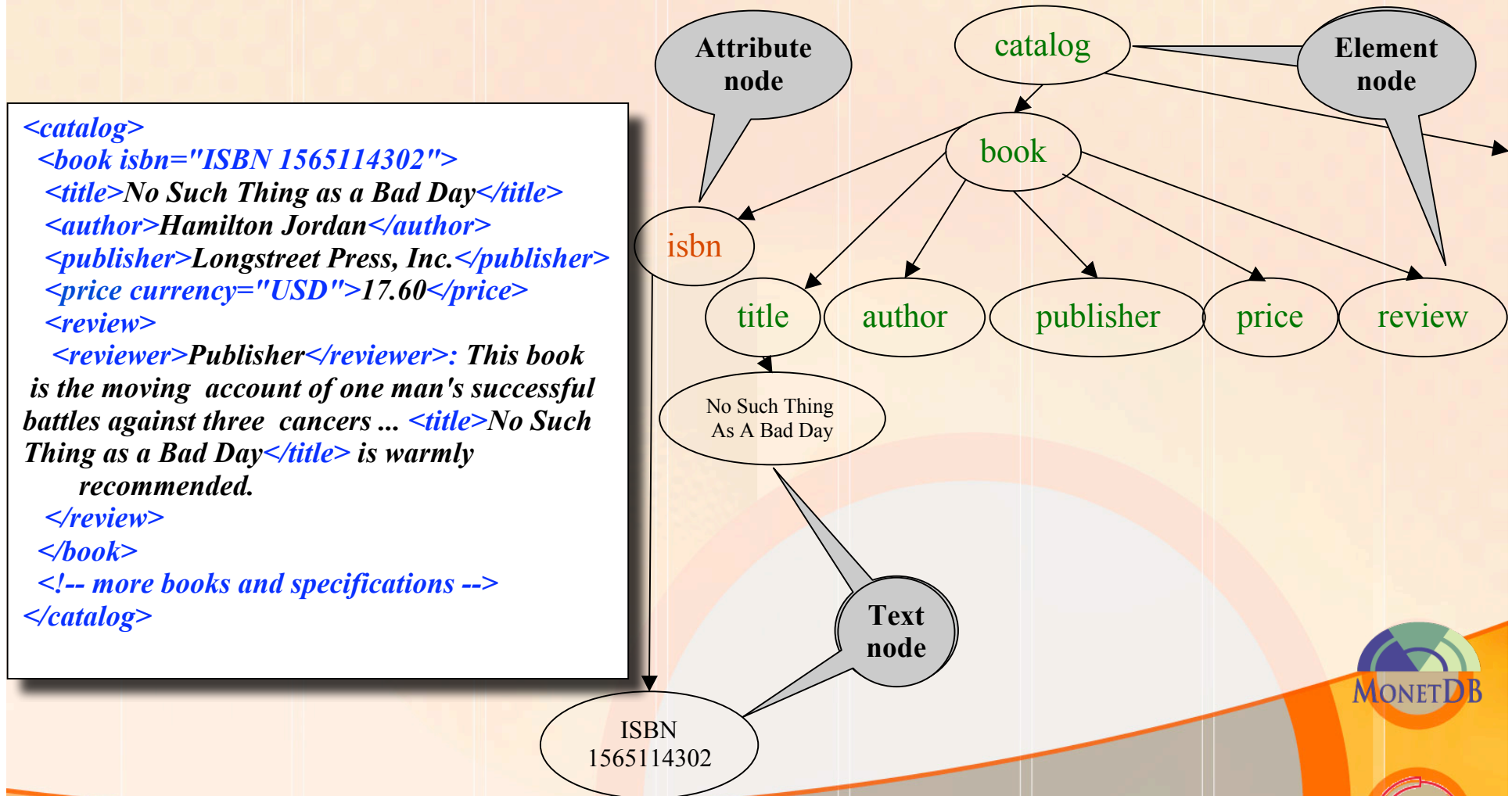


XML

```
<catalog>
  <book isbn="ISBN 1565114302">
    <title>No Such Thing as a Bad Day</title>
    <author>Hamilton Jordan</author>
    <publisher>Longstreet Press, Inc.</publisher>
    <price currency="USD">17.60</price>
    <review>
      <reviewer>Publisher</reviewer>: This book
      is the moving account of one man's successful
      battles against three cancers ... <title>No Such
      Thing as a Bad Day</title> is warmly
      recommended.
    </review>
  </book>
  <!-- more books and specifications -->
</catalog>
```



XML as a tree



XPath

- XML Path Language
- XPath is a language for addressing parts of an XML document = XML Navigation.



XPath: XML Navigation

- *Titles of all books published by Longstreet Press*

```
$cat/catalog/book[publisher="Longstreet Press"]/title
```

```
<title>No Such Thing As A Bad Day</title>
```

- *Publications with Don Chamberlin as author or editor*

```
$cat//*[(author|editor) = "Don Chamberlin"]
```

```
<book><title>XQuery from the Experts</title>...</book>
```

```
<spec><title>XQuery Formal Semantics</title>...</spec>
```



XQuery 1.0, a W3C Recommendation

- **Functional, strongly-typed query language**
- ***XQuery 1.0* =**
 - XPath 2.0* for navigation, selection, extraction**
 - + A few more expressions**
 - For-Let-Where-Order By-Return (FLWOR)
 - XML construction
 - Operators on types
 - + User-defined functions & modules**
 - Modularize large queries
 - Process recursive data
 - + Strong typing**
 - Checks values of required type (operator, function)
 - Guarantees result value instance of output type
 - Enforced statically or dynamically



FLWR (“Flower”) Expressions

XQuery

F OR ...	sequence expression
L ET...	variable definition
W HERE...	condition
R ETURN...	result expression



XQuery Example

- *For each author, return number of books and receipts for books published in the past 2 years, ordered by name*

```
let $cat := fn:doc("www.bn.com/catalog.xml"), ☺Documents☺
    $sales := fn:doc("www.publishersweekly.com/sales.xml")

for $author in distinct-values($cat//author) ☺Grouping☺
let $books := $cat//book[@year >= 2003 and author = $author], ☺Sel.☺
    $receipts := $sales/book[@isbn = $books/@isbn]/receipts ☺Join☺

order by $author ☺Ordering☺

return

<sales> ☺XML Construction☺
  { $author }
  <count> { fn:count($books) } </count> ☺Aggregation☺
  <total> { fn:sum($receipts) } </total>
</sales>
```



Why XML Databases ?

- Managing
 - Large XML Documents
 - Many XML Documents
- Manage =
 - Query (/Transform) + Update
 - Multiple users, Complex Queries, ACID Properties, ..



MonetDB/XQuery

My own XML DBMS with (almost..) full XQuery support.

- *Built purely on an RDBMS, namely MonetDB*

Pathfinder compiler & “staircase join” (see later):



Technical University Munich (Torsten Grust, et al.)



Technical University Twente (Maurice van Keulen)

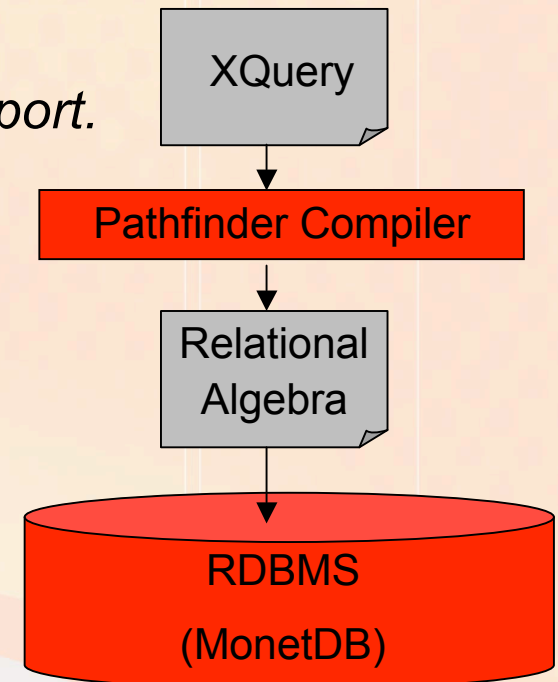
MonetDB High-Performance DBMS



CWI Amsterdam (Peter Boncz, Stefan Manegold)

Useful for:

- **Large XML databases!**
- *Querying XML annotations (multimedia, forensic → NFI)*

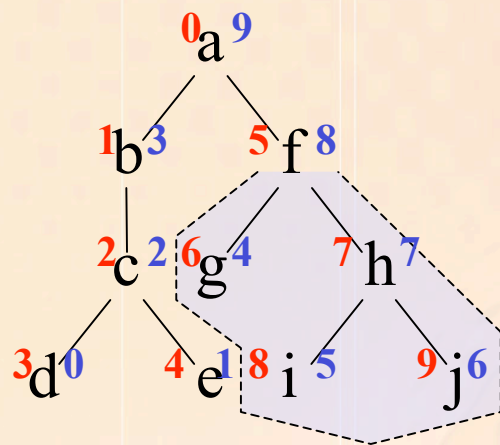


Main ideas

- Relational XQuery
 - Path steps in the pre/post plane
 - Translating for-loops, and beyond
- Query Optimization
 - Order prevention
 - Join recognition
 - Loop-Lifted Staircase join



XML tree



Preorder/postorder rank assignment

Figure 1: A scatter plot showing the relationship between 'pre' (x-axis) and 'post' (y-axis) values. The plot is divided into two regions by a vertical dashed line at pre=5 and a horizontal dashed line at post=8. The region where pre < 5 and post < 8 is shaded orange. The region where pre > 5 and post < 8 is shaded purple. The region where pre > 5 and post > 8 is shaded yellow. A red line connects points a, b, c, d, e, f, g, h, i, j. The points are labeled a through j. The origin is labeled (0,0). The x-axis is labeled 'pre' and the y-axis is labeled 'post'.

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

Preorder coincides with order of opening tags
Postorder coincides with order of closing tags

XPath evaluation (SQL)

f/following:

```
SELECT * FROM pre_post WHERE pre > f.pre AND post > f.post
```

f/descendant:

```
SELECT * FROM pre_post WHERE pre > f.pre AND post < f.post
```

f/preceding:

```
SELECT * FROM pre_post WHERE pre < f.pre AND post < f.post
```

f/ancestor:

```
SELECT * FROM pre_post WHERE pre < f.pre AND post > f.post
```

Similar queries for all 13 XPath axes



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
      AND oa.name = "open_auction" AND oa.kind = "elem"
      AND b.pre > oa.pre AND b.post < oa.post
      AND b.level = oa.level + 1
      AND b.name = "bidder" AND b.kind < "elem"
      AND a.pre > oa.pre AND a.post < oa.post
      AND a.level = oa.level + 1
      AND a.name = "annotation" AND a.kind < "elem"
ORDER BY a.pre
```

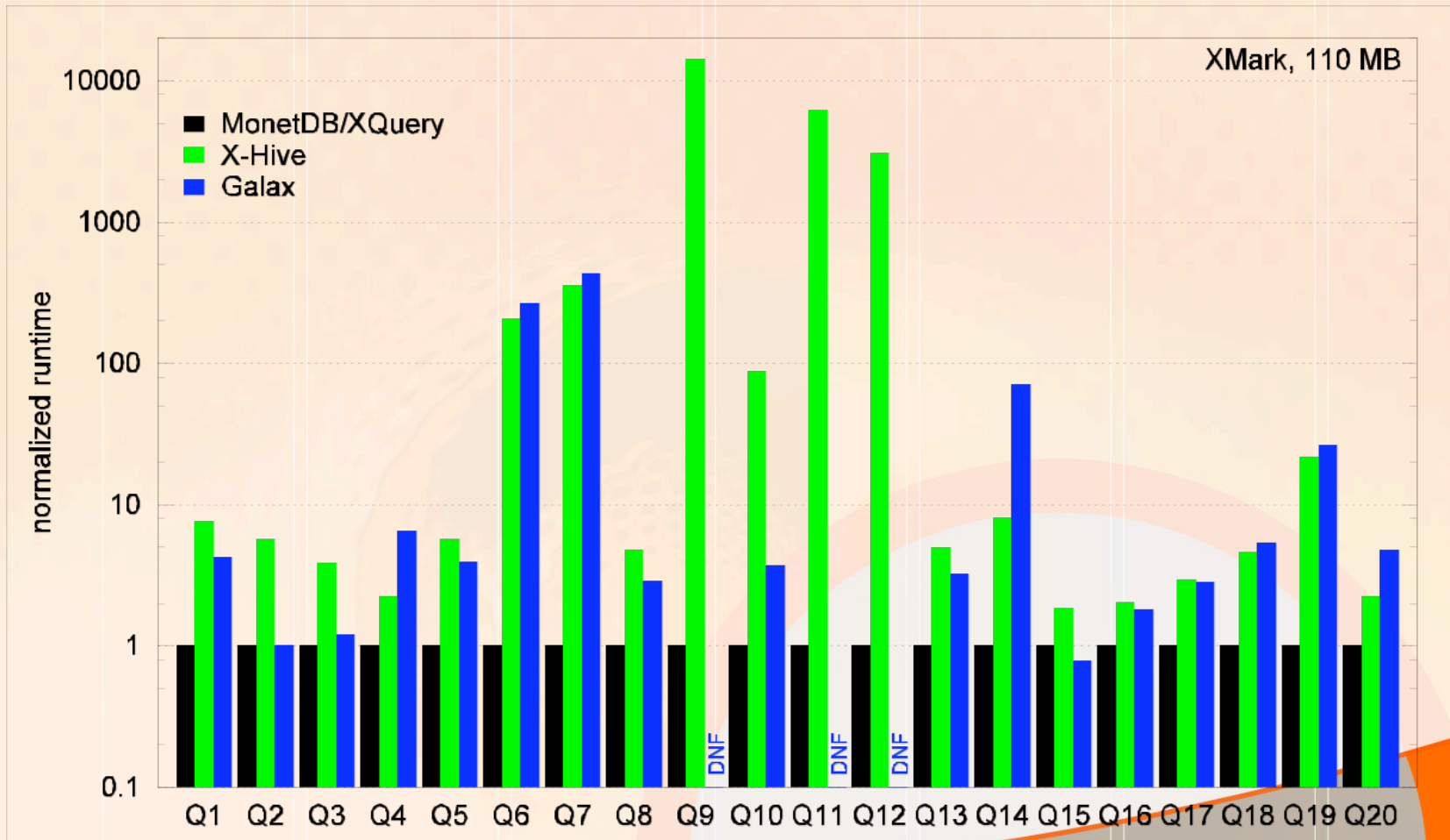


	1MB XML	1GB XML
Galax (GLX)		
X-Hive (XHV)		
Berkeley DB/XML (BDB)		
eXist (EXT)		



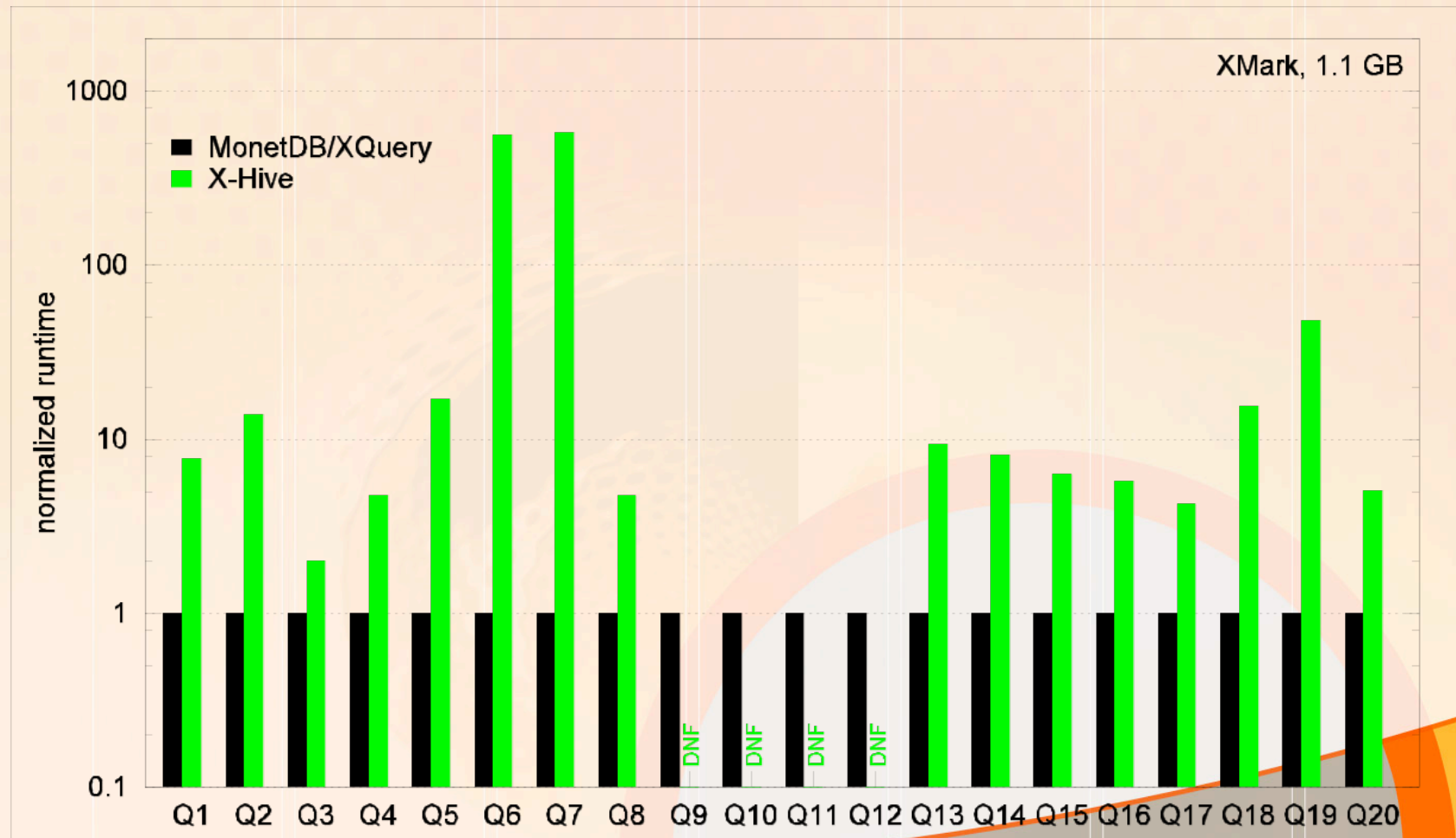
Performance (1)

XMark benchmark, 110 MB: MonetDB/XQuery vs. X-Hive & Galax



Performance (2)

XMark benchmark, 1.1 GB: MonetDB/XQuery vs. X-Hive

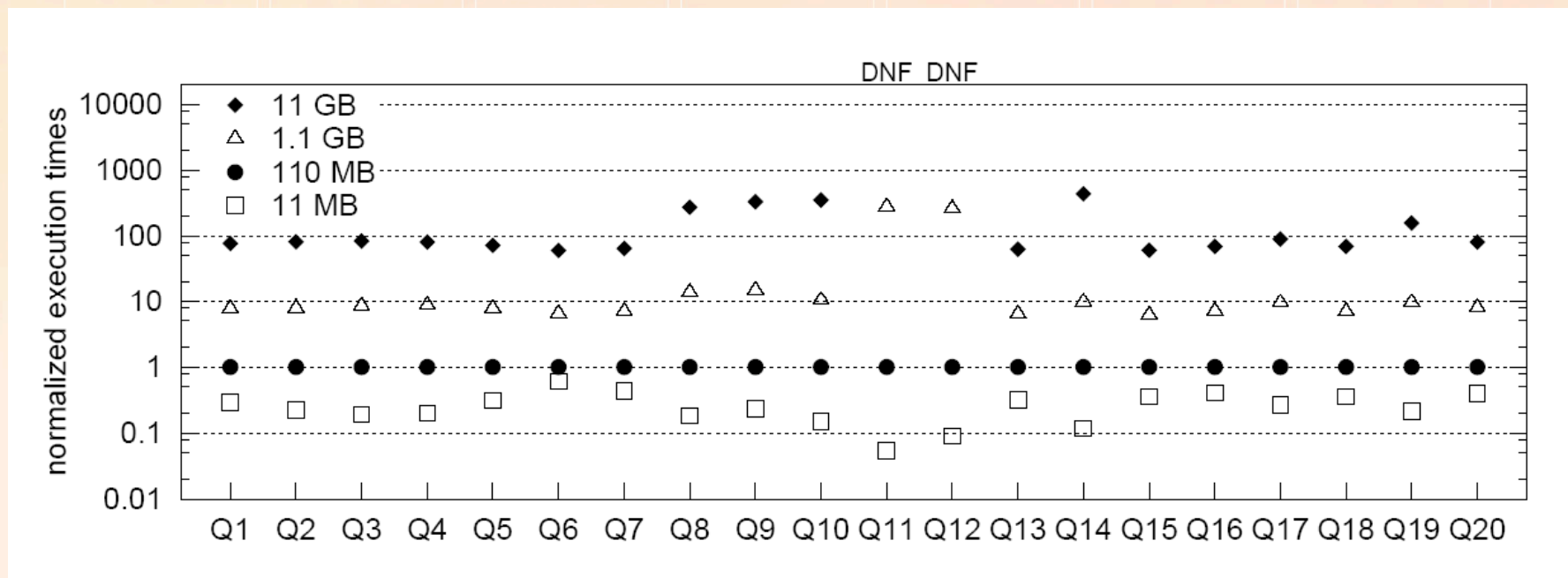


Scalability

Unsurpassed scalability

- Standard Opteron PC, 8GB RAM, 64-bit Linux
- Can process 11GB documents!

Mostly linear scaling with document size



Gràcies!

