MonetDB:

Open-source Database Technology Beyond Textbooks

http://monetdb.cwi.nl/

Stefan Manegold
Stefan.Manegold@cwi.nl
http://homepages.cwi.nl/~manegold/

Vacancies

2010-09-03:
The CWI Database Architectures group is hiring. We are looking for 1 post-doc and 3 PhD students. Check the descriptions of the vacancies for details.

New release

2010-08-30:
The second "Jun 2010" bug fix release of MonetDB is available for Download.

Publications

2009-08-23:
Database Architecture Evolution: Mammals Flourished long before Dinosaurs became Extinct VLDB 10-year Best Paper award, cf., paper, press release

2009-06-21:
An Architecture for Recycling Intermediates in a Column-store ACM SIGMOD Best Paper Runner-up Award, cf., paper, press release

SOURCEFORGE.NET

Query Processing at Light Speed

MonetDB is a open-source database system for high-performance applications in data mining, OLAP, GIS, XML Query, text and multimedia retrieval. MonetDB often achieves a significant speed improvement for SQL and XQuery over other open-source systems. Use the product strands as your roadmap for exploration.

Products

- MonetDB/SQL our relational database solution.
- MonetDB/XQuery our XML database solution.
- MonetDB Server our multi-model database server.

Science & Technology at Work

MonetDB achieves its goal by innovations at all layers of a DBMS, e.g. a storage model based on vertical fragmentation, a modern CPU-tuned query execution architecture, automatic and self-tuning indexes, run-time query optimization, and a modular software architecture. In-depth information on the technical innovations in the design and implementation of MonetDB can be found in our science library.

Commercial support

An independent company has been set up to facilitate the dissemination of the MonetDB code base as an open source solution worldwide and to provide expert consultancy, participation in projects, and joint ventures on a commercial basis to realize its mission.

Who?

Database Architecture Group @ CWI

*(**C**entrum **W**iskunde & **I**nformatica)*

- People
  - 1 Professor
  - 1 Scientific programmer
  - 3 Tenure researchers
  - 1 Tenure-track researcher
  - 3 Postdocs
  - 4 PhD students
  - Master students
  - *(6 nationalities)*
  - **4 Vacancies**

What?

Database Architecture Group @ CWI

*(**C**entrum **W**iskunde & **I**nformatica)*

- Research Focus
  - Database Architecture
    - Hardware Awareness, Self-*, Performance, Multi-modal
- Research Approach
  - R & D
  - Strong Engineering
  - System-oriented
  - Experimental
- Research Products
  - Scientific Papers, PhD Theses, ...
  - Open-Source MonetDB

Motivation

- Relational DBMSs dominate since the late 1970's / early 1980's
  - Ingres, Oracle, SQLserver, DB2, ...
  - Transactional workloads (OLTP, row-wise access)
  - I/O based processing

Why?

Workload changes: Transactions (OLTP) vs ...

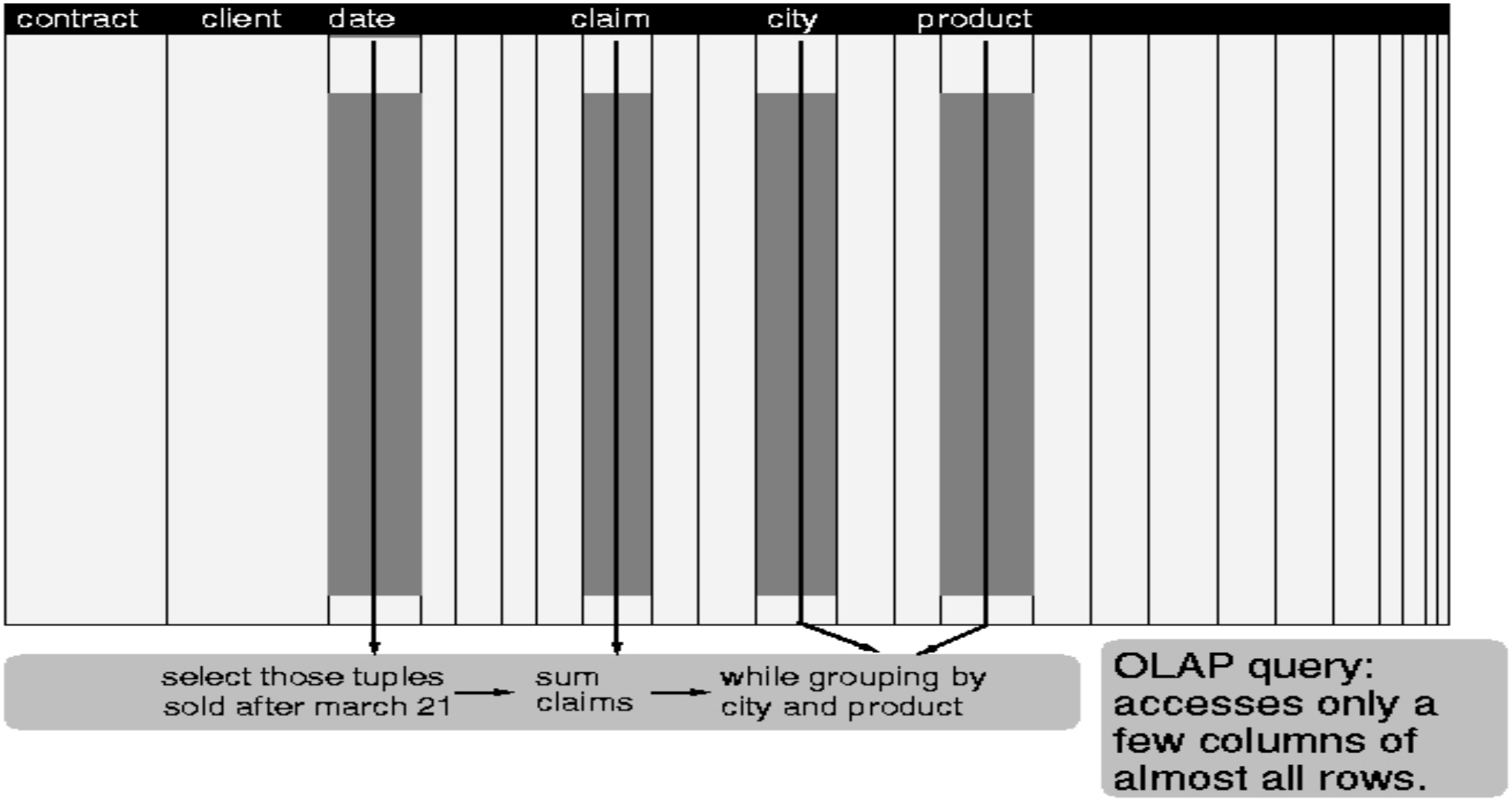
contract	client	date	name	price		city	product
12302346	10042334		Eno			Redmond	Car
37611373	10987097		Gotz			~~Berkeley~~ ← Redmond	update query
95371001	10032112		Chen			Seattle	House lookup query
51213123	10032423		Jones			Washington	Travel
54535545	10087823		Smith			New York	House
45447894	10013232		Doe			Boston	Car insert query

find client 10032112

OLTP queries: access all columns of just one row.

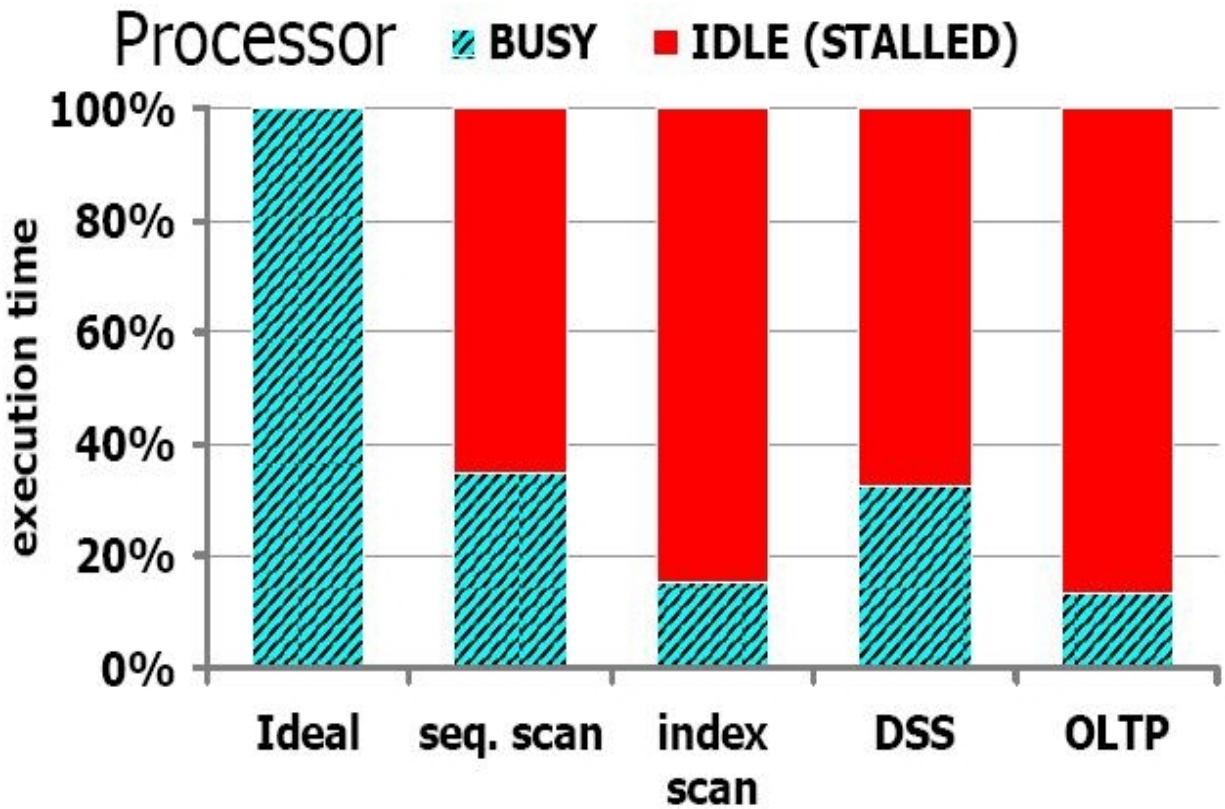
Why?

Workload changes: ... vs OLAP, BI, Data Mining, ...



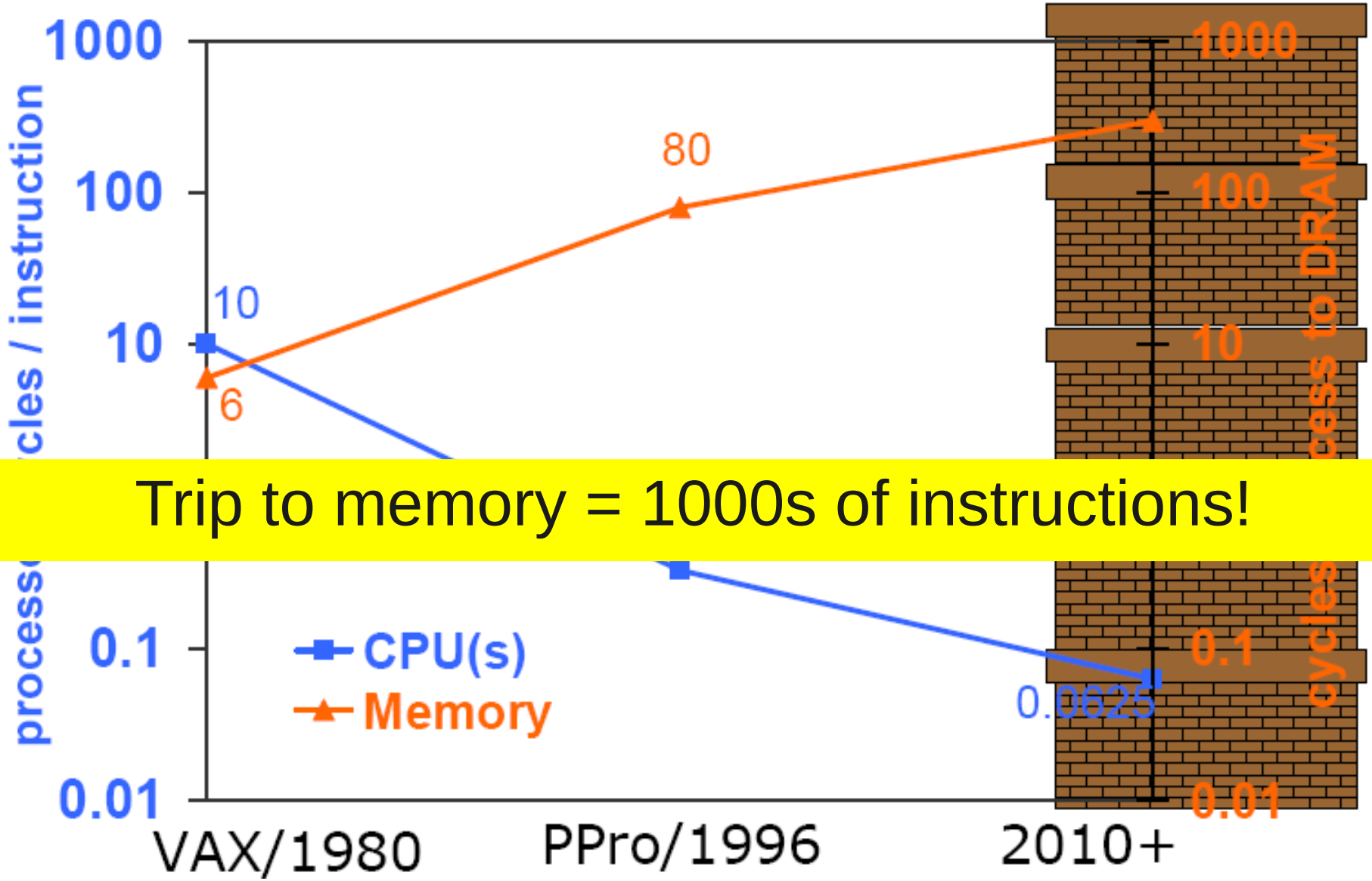
Databases hit The Memory Wall

- Detailed and exhaustive analysis for different workloads using 4 RDBMSs by Ailamaki, DeWitt, Hill,, Wood in VLDB 1999: "*DBMSs On A Modern Processor: Where Does Time Go?*"
- CPU is 60%-90% idle, waiting for memory:
  - L1 data stalls
  - L1 instruction stalls
  - L2 data stalls
  - TLB stalls
  - Branch mispredictions
  - Resource stalls



Why?

Hardware Changes: The Memory Wall



Trip to memory = 1000s of instructions!

Why?

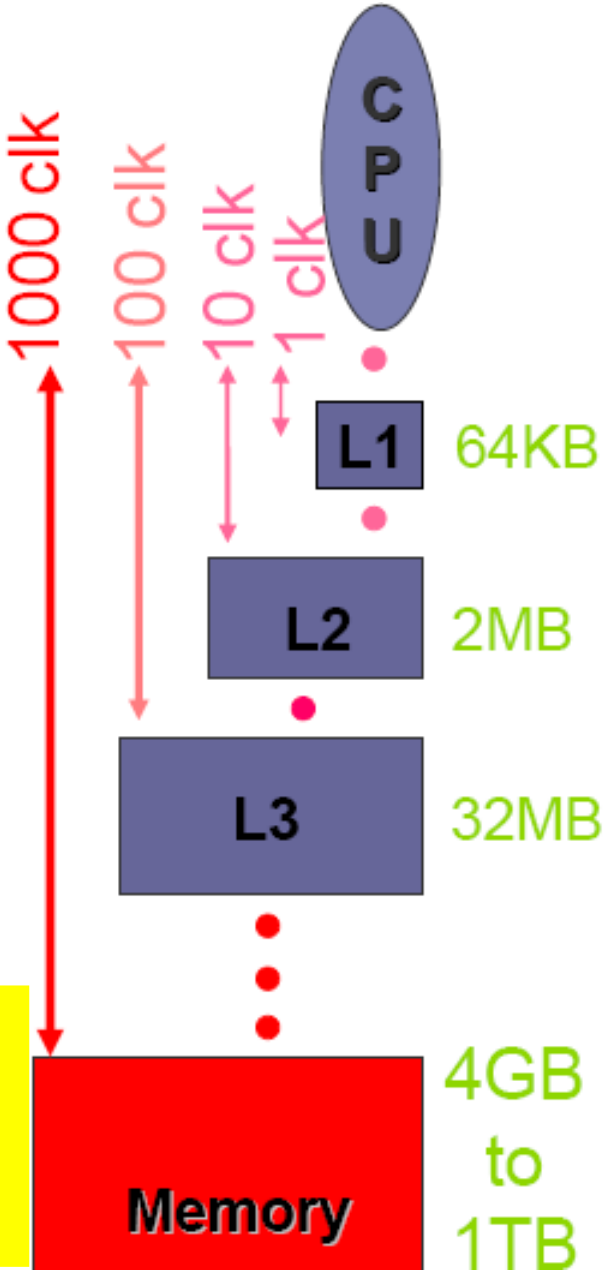
Hardware Changes: Memory Hierarchies

- Caches trade off capacity for speed
- Exploit instruction/data locality
- Demand fetch/wait for data

[ADH99]:

- Running top 4 database systems
- **At most 50% CPU utilization**

+Transition Lookaside Buffer (TLB)
Cache for VM address translation ➔
only 64 entries!



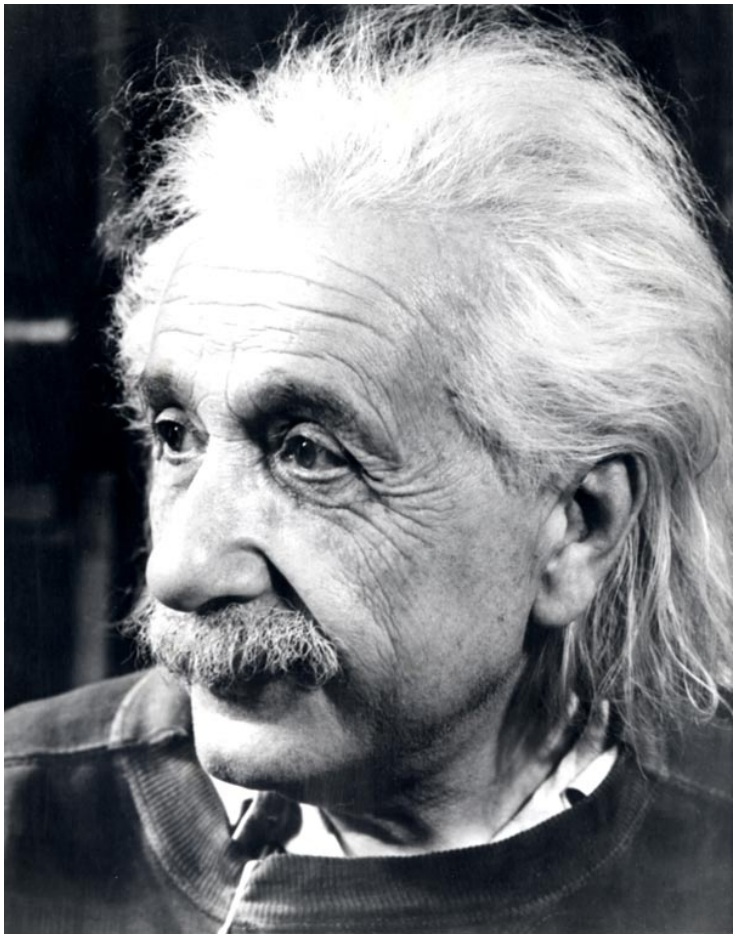
Evolution

It is not the strongest of the species that survives, nor the most intelligent, but the one most responsive to change.

Charles Darwin (1809 - 1882)

Solution

"We can't solve problems by using the same kind of thinking we used when we created them."



What?

MonetDB

- Database kernel developed at CWI since 1993
  - *Research prototype turned into open-source product*
- Pioneering columnar database architecture
  - *Complete Relational/SQL & XML/XQuery DBMS*
- Focusing on in-memory processing
  - *Data is kept persistent on disk and can exceed memory limits*
- Aiming at OLAP, BI & Data Mining workloads (“read-dominated”)
  - *Supporting ACID transactions (WAL, optimistic CC)*
- Platform for database architecture research
  - *Used in academia (research & teaching) & commercial environments*
- Back-end for various DB research projects:

  Multi-Media DB & IR (“Tijah”), XML/XQuery (“Pathfinder”),

  Data Mining (“Proximity”), Digital Forensics (“XIRAF”), GIS (“OSM”), ...

How is MonetDB Different

- full vertical fragmentation: always!
  - everything in binary (2-column) tables (**B**inary **A**ssociation **T**able)
  - saves you from table scan hell in OLAP and Data Mining
- RISC approach to databases
  - simple back-end data model
  - simple back-end query language (binary/columnar relational algebra)
  - don't need (to pay for) a buffer manager => manage virtual memory
  - explicit transaction management => DIY approach to ACID
- Multiple user data models & query languages
  - SQL, XML/XQuery, (RDF/SPARQL)
  - front-ends map data models and query languages

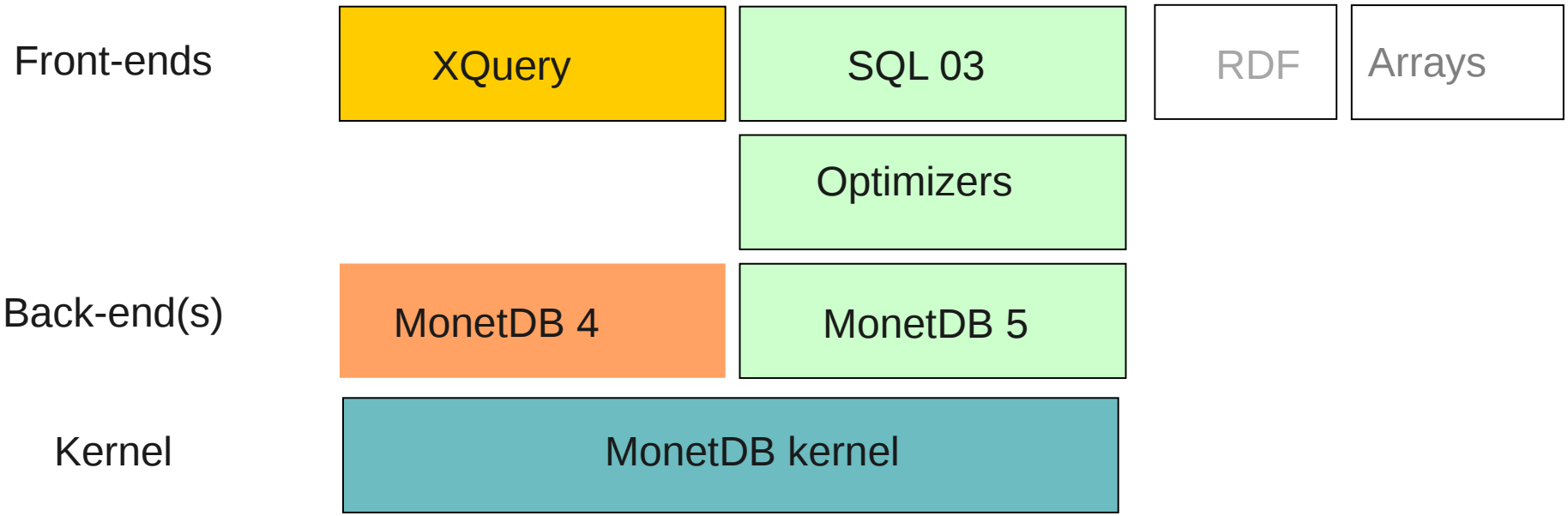
How is MonetDB Different

- optimized for in-memory processing
  - efficiently exploit large main-memories
  - data is persistent, though
- operator-at-a-time bulk processing
  - avoids tuple-at-a-time management overhead
- CPU and memory cache optimized
  - programming team experienced in main memory DBMS techniques
  - use of scientific programming optimizations (loop unrolling)

MonetDB vs Traditional DBMS Architecture

- **Architecture-Conscious Query Processing**
  - **vs Magnetic disk I/O conscious processing**
  - *Data layout, algorithms, cost models*
- **RISC Relational Algebra (operator-at-a-time)**
  - **vs Tuple-at-a-time Iterator Model**
  - *Faster through simplicity: no tuple expression interpreter*
- **Multi-Model: ODMG, SQL, XML/XQuery, ..., RDF/SPARQL**
  - **vs Relational with Bolt-on Subsystems**
  - *Columns as the building block for complex data structures*
- **Decoupling of Transactions from Execution/Buffering**
  - **vs ARIES integrated into Execution/Buffering/Indexing**
  - *ACID, but not ARIES.. Pay as you need transaction overhead.*
- **Run-Time Indexing and Query Optimization**
  - **vs Static DBA/Workload-driven Optimization & Indexing**
  - *Extensible Optimizer Framework;*
  - *cracking, recycling, sampling-based runtime optimization*

The MonetDB Software Stack



Storing Relations in MonetDB

Front-End

contract	client	date	name	price	city	product
12302346	10042334		Eno		Redmond	Car
37611373	10987097		Gotz		Redmond	House
51213123	10032423		Jones		Washington	Travel
54535545	10087823		Smith		New York	House
45447894	10013232		Doe		Boston	Car
95371001	10032112		Chen		Seattle	House

logical data model

mapping rules

physical data model (BATs)

Void	contract
1000	12302346
.	37611373
.	51213123
.	54535545
.	45447894
.	95371001

Void	client
1000	10042334
.	10987097
.	10032423
.	10087823
.	10013232
.	10032112

Void	name
1000	Eno
.	Gotz
.	Jones
.	Smith
.	Doe
.	Chen

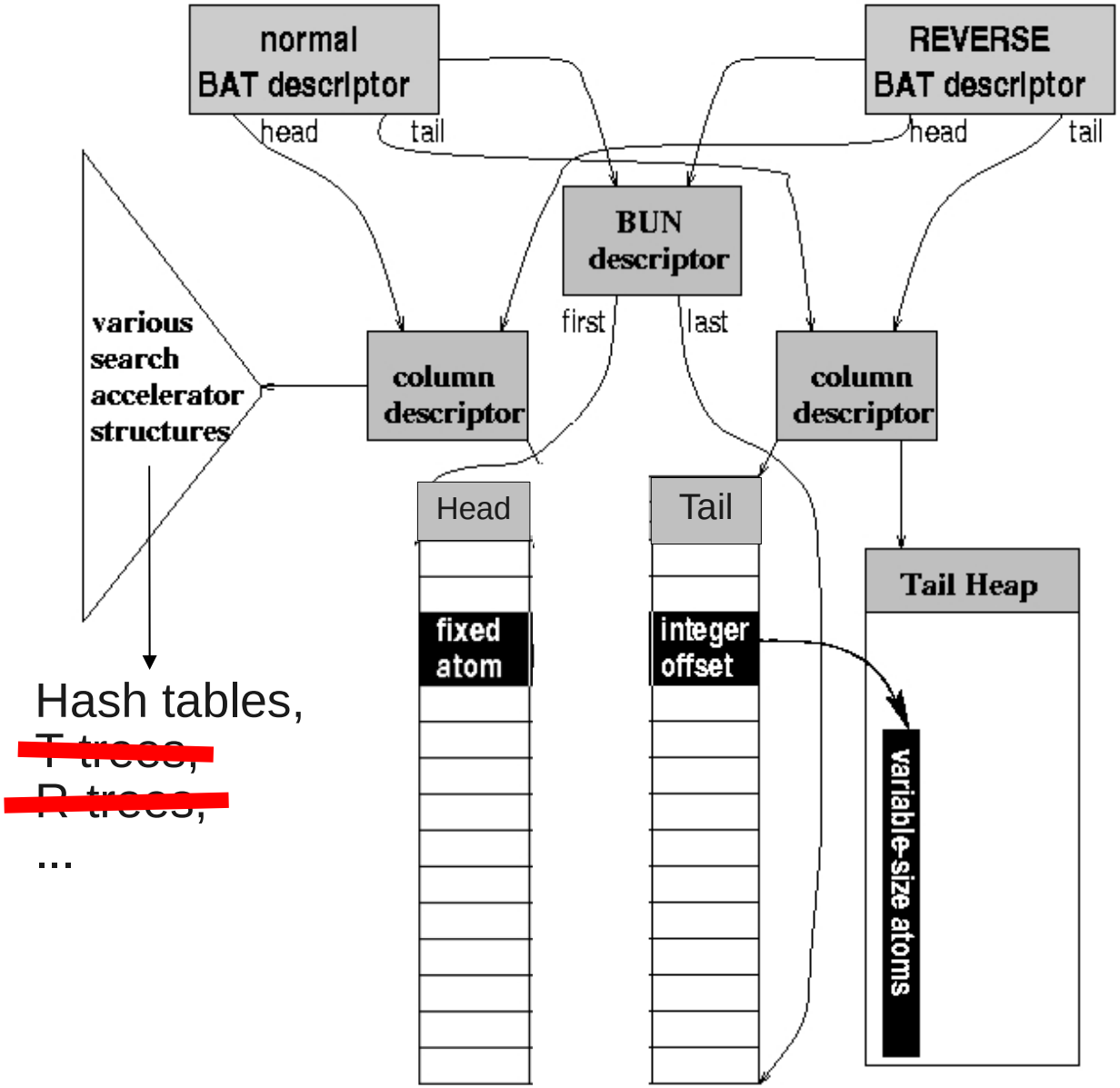
Void	city
1000	Redmond
.	Redmond
.	Washington
.	New York
.	Boston
.	Seattle

Void	product
1000	Car
.	House
.	Travel
.	House
.	Car
.	House

Monet

Virtual OID: seqbase=**1000** (increment=1)

BAT Data Structure



BAT:
binary association table

BUN:
binary unit

Head & Tail:
- consecutive memory blocks (arrays)
- memory-mapped files

Tail Heap:
- best-effort duplicate elimination for strings (~ dictionary encoding)

BAT Property Management

type	- (physical) type number
dense	- dense ascending range (OID only)
(rev)sorted	- column sorted ascending (descending)
constant	- all equal values
align	- unique sequence id
key	- no duplicates on column
set	- no duplicates in BAT
hash	- accelerator flag
mirrored	- head=tail value
count	- cardinality
nonil	- no NIL (NULL) values in column

RISC Relational Algebra

```
SELECT   id, name, (age-30)*50 as bonus
FROM     people
WHERE    age > 30
```

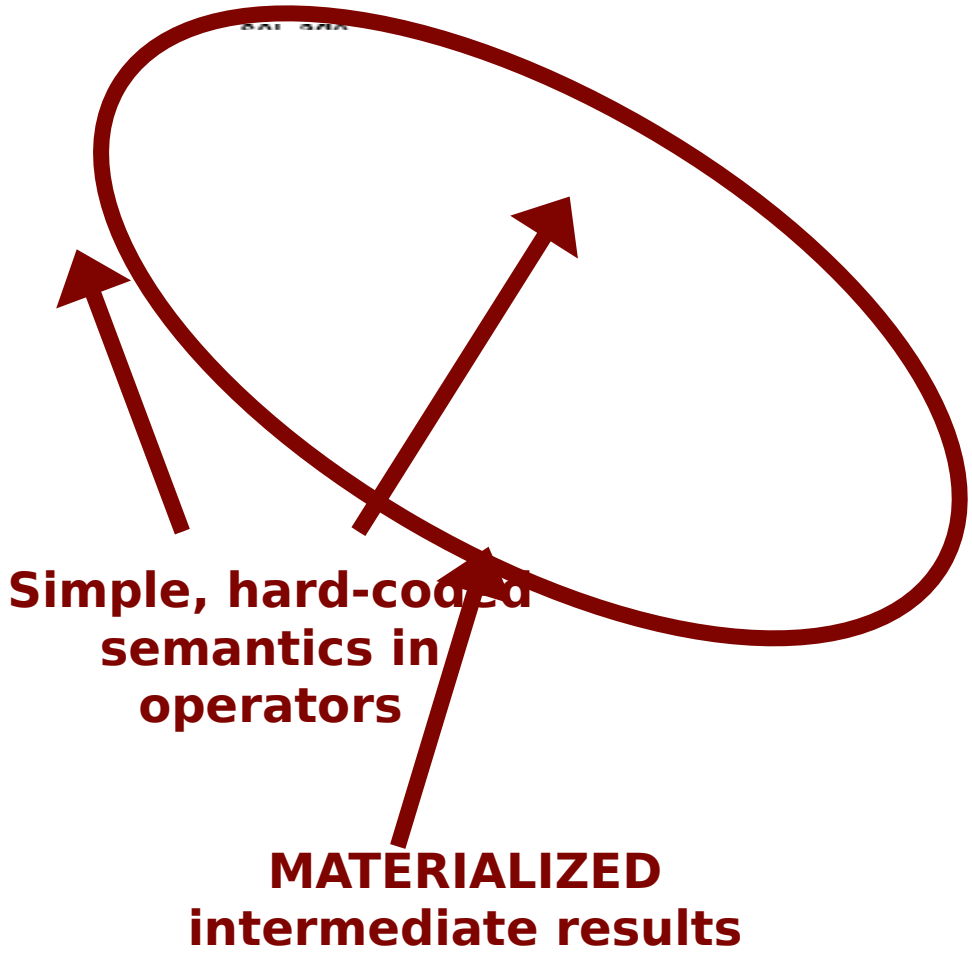
```
batcalc_minus_int(int* res,
                  int* col,
                  int  val,
                  int n)
{
    for(i=0; i<n; i++)
        res[i] = col[i] - val;
}
```

CPU ☺? Give it "nice" code !

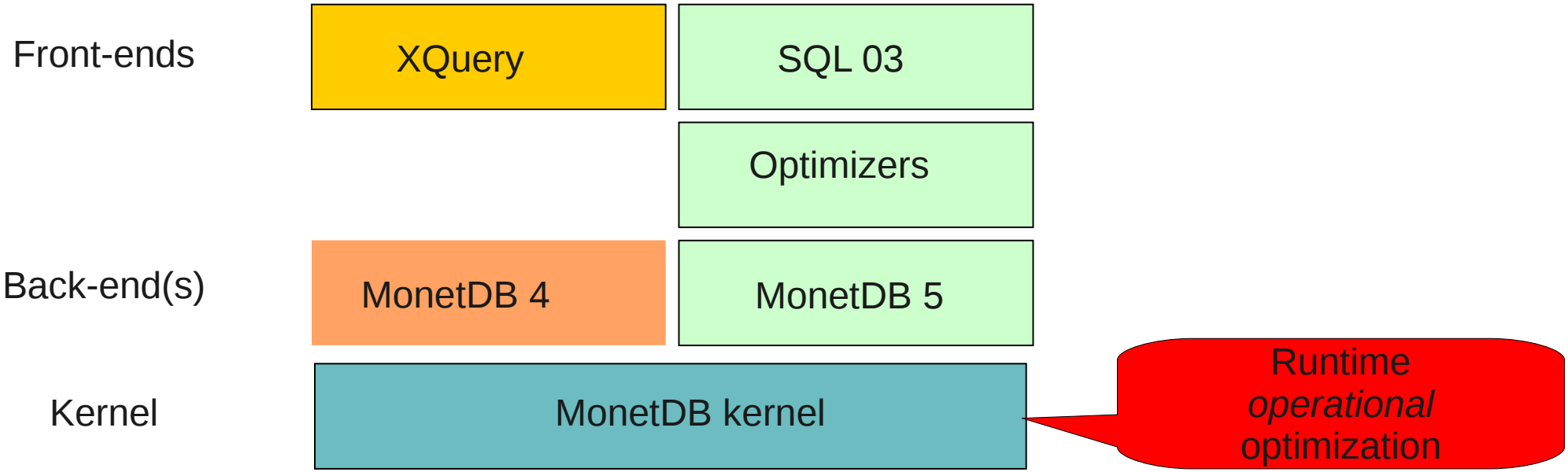
- few dependencies (control,data)
- CPU gets out-of-order execution
- compiler can e.g. generate SIMD

One loop for an entire column

- no per-tuple interpretation
- arrays: no record navigation
- better instruction cache locality



The MonetDB Software Stack



Processing Model (MonetDB Kernel)

- **Bulk processing:**
  - full materialization of all intermediate results
- **Binary (i.e., 2-column) algebra core:**
  - select, join, semijoin, outerjoin
  - union, intersection, diff (BAT-wise & column-wise)
  - group, count, max, min, sum, avg
  - reverse, mirror, mark
- **Runtime *operational* optimization:**
  - Choosing optimal algorithm & implementation according to input properties and system status

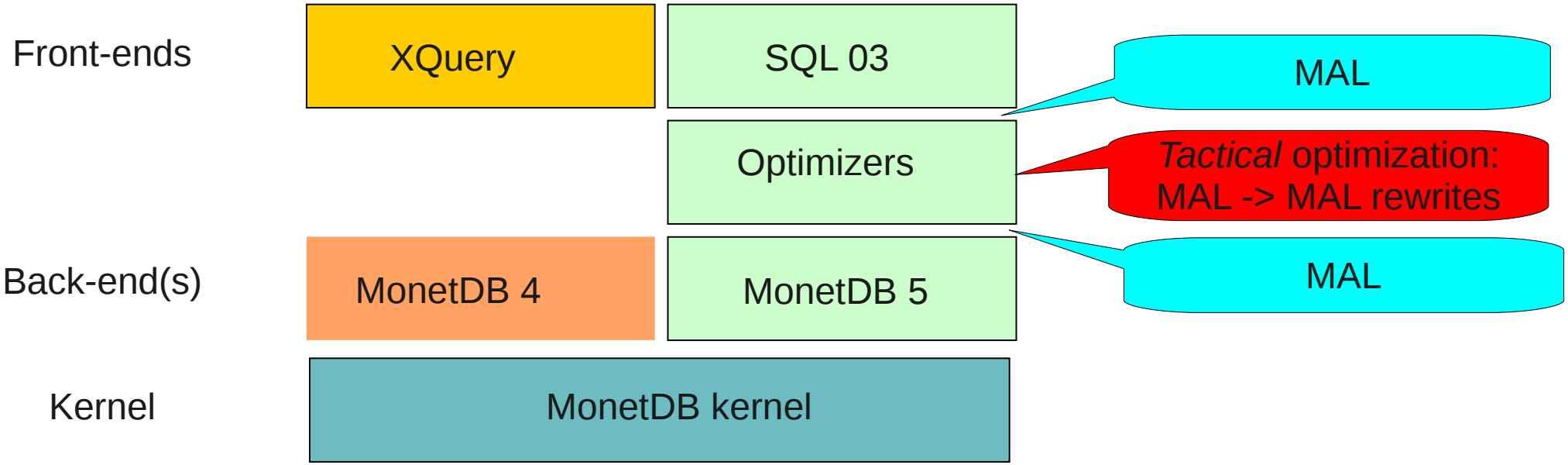
Processing Model (MonetDB Kernel)

- Heavy use of code expansion to reduce cost

1 algebra operator	select()
3 overloaded operators	select(“=“,value) select(“between”,L,H) select(“fcn”,parm)
10 operator algorithms	scan hash-lookup bin-search bin-tree pos-lookup
~1500(!) routines (macro expansion)	scan_range_select_oid_int(), hash_equi_select_void_str(), …

- ~1500 selection routines
- 149 unary operations
- 335 join/group operations
- ...

The MonetDB Software Stack



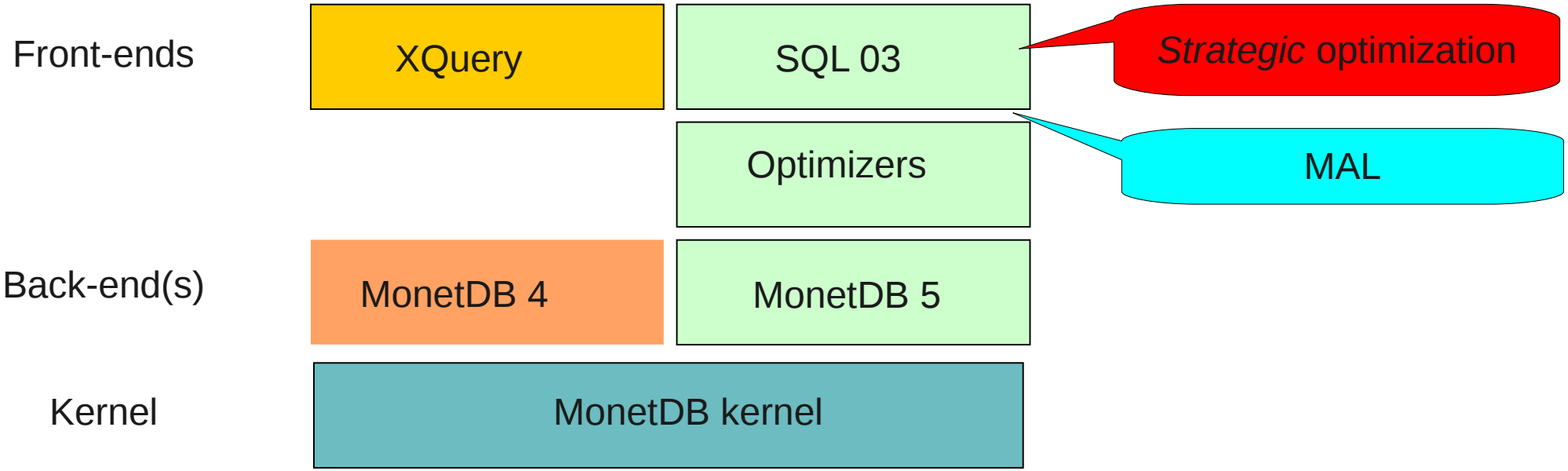
MonetDB/5 Back-end: MAL

- **MAL: Monet Assembly Language**
  - textual interface
  - Interpreted language
- **Designed as system interface language**
  - Reduced, concise syntax
  - Strict typing
  - Meant for automatic generation and parsing/rewriting/processing
  - Not meant to be typed by humans
- **Efficient parser**
  - Low overhead
  - Inherent support for *tactical* optimization: MAL -> MAL
  - Support for optimizer plug-ins
  - Support for runtime schedulers
- **Binary-algebra core**
- **Flow control** (MAL is computational complete)

MonetDB/5 Back-end: MAL Optimizers

- **General front-end independent MAL -> MAL rewriting**
  - Implemented once, shared by all (future) front-ends
- **Examples**:
  - Constant propagation
  - Scalar expression evaluation
  - Dead-code elimination
  - Common sub-expression elimination
  - Empty result set removal
  - Reordering to optimize intermediate result usage
  - Reordering of linear (projection-) join chains
  - Parallelization:
    - Dataflow analysis
    - Horizontal partitioning
    - Remote execution
  - *Cracking*
  - *Recycling*
  - *...*

The MonetDB Software Stack



MonetDB Front-end: SQL

- **SQL 2003**
  - Stick to the standard; avoid “private” extensions, renaming, etc.
- **Parse SQL into logical n-ary relational algebra tree**
- **Translate n-ary relational algebra into logical 2-ary relational algebra**
- **Turn logical 2-ary plan into physical 2-ary plan (MAL program)**
  - Generate internal tree representation, not textual MAL program
- **Front-end specific *strategic* optimization:**
  - Heuristic optimization during all three previous steps
- **Primary key and distinct constraints:**
  - Create and maintain hash indices
- **Foreign key constraints**
  - Create and maintain foreign key join indices
- **Exploit both above indices during query evaluation**

MonetDB Front-end: SQL

```
PLAN SELECT a FROM t WHERE c < 10;

project (
  select (
    table(sys.t) [ t.a, t.c, t.%TID% NOT NULL ]
  ) [ t.c < convert(10) ]
) [ t.a ]
```

MonetDB Front-end: SQL

```
EXPLAIN SELECT a FROM t WHERE c < 10;

function user.s1_1{autoCommit=true}():void;
barrier _55 := language.dataflow();
    _2:bat[:oid,:int]  := sql.bind("sys","t","c",0);
    _7 := algebra.thetauselect(_2,10,"<");
    _2:bat[:oid,:int]  := nil:BAT;
    _10 := algebra.markT(_7,0@0);
    _7 := nil:BAT;
    _11 := bat.reverse(_10);
    _10 := nil:BAT;
    _12:bat[:oid,:int]  := sql.bind("sys","t","a",0);
    _14 := algebra.leftjoin(_11,_12);
    _11 := nil:BAT;
    _12:bat[:oid,:int]  := nil:BAT;
exit _55;
    _15 := sql.resultSet(1,1,_14);
    sql.rsColumn(_15,"sys.t","a","int",32,0,_14);
    _14 := nil:BAT;
    _21 := io.stdout();
    sql.exportResult(_21,_15);
end s1_1;
```

MonetDB Front-end: SQL

```
PLAN SELECT a, z FROM t, s WHERE t.c = s.x;


project (
  join (
    table(sys.t) [ t.a, t.c, t.%TID% NOT NULL ],
    table(sys.s) [ s.x, s.z, s.%TID% NOT NULL ]
  ) [ t.c = s.x ]
) [ t.a, s.z ]
```

MonetDB Front-end: SQL

```
EXPLAIN SELECT a, z FROM t, s WHERE t.c = s.x;

% .explain # table_name
% mal # name
% clob # type
% 0 # length
function user.s2_1{autoCommit=true}():void;
barrier _73 := language.dataflow();
    _2:bat[:oid,:int]  := sql.bind("sys","t","c",0);
    _7:bat[:oid,:int]  := sql.bind("sys","s","x",0);
    _10 := bat.reverse(_7);
    _7:bat[:oid,:int]  := nil:BAT;
    _11 := algebra.join(_2,_10);
    _2:bat[:oid,:int]  := nil:BAT;
    _10 := nil:BAT;
    _13 := algebra.markT(_11,0@0);
    _14 := bat.reverse(_13);
    _13 := nil:BAT;
    _15:bat[:oid,:int]  := sql.bind("sys","t","a",0);
    _17 := algebra.leftjoin(_14,_15);
    _14 := nil:BAT;
    _15:bat[:oid,:int]  := nil:BAT;
    _18 := bat.reverse(_11);
    _11 := nil:BAT;
    _19 := algebra.markT(_18,0@0);
    _18 := nil:BAT;
    _20 := bat.reverse(_19);
    _19 := nil:BAT;
    _21:bat[:oid,:int]  := sql.bind("sys","s","z",0);
    _23 := algebra.leftjoin(_20,_21);
    _20 := nil:BAT;
    _21:bat[:oid,:int]  := nil:BAT;
exit _73;
    _24 := sql.resultSet(2,1,_17);
    sql.rsColumn(_24,"sys.t","a","int",32,0,_17);
    _17 := nil:BAT;
    sql.rsColumn(_24,"sys.s","z","int",32,0,_23);
    _23 := nil:BAT;
    _33 := io.stdout();
    sql.exportResult(_33,_24);
end s2_1;
```

MonetDB Front-end: SQL

- **Updates / transactions:**
  - Write ahead logging
  - Optimistic concurrency control
  - “delta” BATs

TPC-H

	MonetDB	PostgreSQL	MySQL
	5.2.3	8.2.6	5.0.45
SF	0.01		
oad	1097	1734	4409
	ms		
1	29	558	366
2	9	15	35
3	12	39	141
4	10	18	34
5	12	21	223
6	4	44	156
7	13	31	59
8	9	35	30
9	14	124	75
10	12	20	132
11	7	21	27
12	9	61	74
13	47	51	112
14	7	43	301
15	6	67	152
16	9	46	80
17	4	12	10
18	11	92	18
19	18	64	
20	12	1810	11
21	26	199	27
22	8	420	13

TPC-H
60K rows line_item table
Comfortably fit in memory
Performance in milliseconds

ATHLON X2 3800+ (2000mhz) 2 disks in raid 0, 2G main memory

TPC-H

	MonetDB 5.2.3	PostgreSQL 8.2.6	MySQL 5.0.45	MonetDB 5.2.3	PostgreSQL 8.2.6	MySQL 5.0.45
SF	0.01			1		
oad	1097	1734	4409	96466	342103	140888
	ms			ms		
1	29	558	366	4004	61253	36305
2	9	15	35	104	3344	152870
3	12	39	141	935	18504	34137
4	10	18	34	663	3273	3187
5	12	21	223	867	758	19931
6	4	44	156	171	14011	6897
7	13	31	59	902	17395	5967
8	9	35	30	353	17101	2940
9	14	124	75	855	49769	9233
10	12	20	132	692	1014	12424
11	7	21	27	65	2491	39164
12	9	61	74	404	14303	6023
13	47	51	112	5532	6461	8128
14	7	43	301	350	13129	32832
15	6	67	152	110	15093	18023
16	9	46	80	487	8972	9086
17	4	12	10	176	> 1 hour	1042
18	11	92	18	698	35273	> 1 hour
19	18	64		1060	15208	297
20	12	1810	11	538	> 1 hour	16936
21	26	199	27	2483	47391	99153
22	8	420	13	246	> 1 hour	718

Scale-factor 1
6M row line-item table

Out of the box performance
Queries produce empty
or erroneous results

ATHLON X2 3800+ (2000mhz) 2 disks in raid 0, 2G main memory

TPC-H

	MonetDB 5.2.3	PostgreSQL 8.2.6	MySQL 5.0.45	MonetDB 5.2.3	PostgreSQL 8.2.6	MySQL 5.0.45	MonetDB 5.2.3	PostgreSQL 8.2.6	MySQL 5.0.45
SF	0.01			1			2		
oad	1097	1734	4409	96466	342103	140888	211	733	303
	ms			ms			sec		
1	29	558	366	4004	61253	36305	6	121	80
2	9	15	35	104	3344	152870	1	6	308
3	12	39	141	935	18504	34137	3	31	91
4	10	18	34	663	3273	3187	1	6	17
5	12	21	223	867	758	19931	3	2	5755
6	4	44	156	171	14011	6897	0	23	27
7	13	31	59	902	17395	5967	2	37	2655
8	9	35	30	353	17101	2940	121	32	504
9	14	124	75	855	49769	9233	3	256	57
10	12	20	132	692	1014	12424	3	4	91
11	7	21	27	65	2491	39164	1	5	175
12	9	61	74	404	14303	6023	2	27	23
13	47	51	112	5532	6461	8128	20	15	132
14	7	43	301	350	13129	32832	2	27	15193
15	6	67	152	110	15093	18023	1	31	49
16	9	46	80	487	8972	9086	1	17	19
17	4	12	10	176	> 1 hour	1042	4	> 1 hour	27
18	11	92	18	698	35273	> 1 hour	13	125	> 1 hour
19	18	64		1060	15208	297	4	31	50
20	12	1810	11	538	> 1 hour	16936	1	> 1 hour	162
21	26	199	27	2483	47391	99153	131	107	1492
22	8	420	13	246	> 1 hour	718	3	> 1 hour	1755

ATHLON X2 3800+ (2000mhz) 2 disks in raid 0, 2G main memory

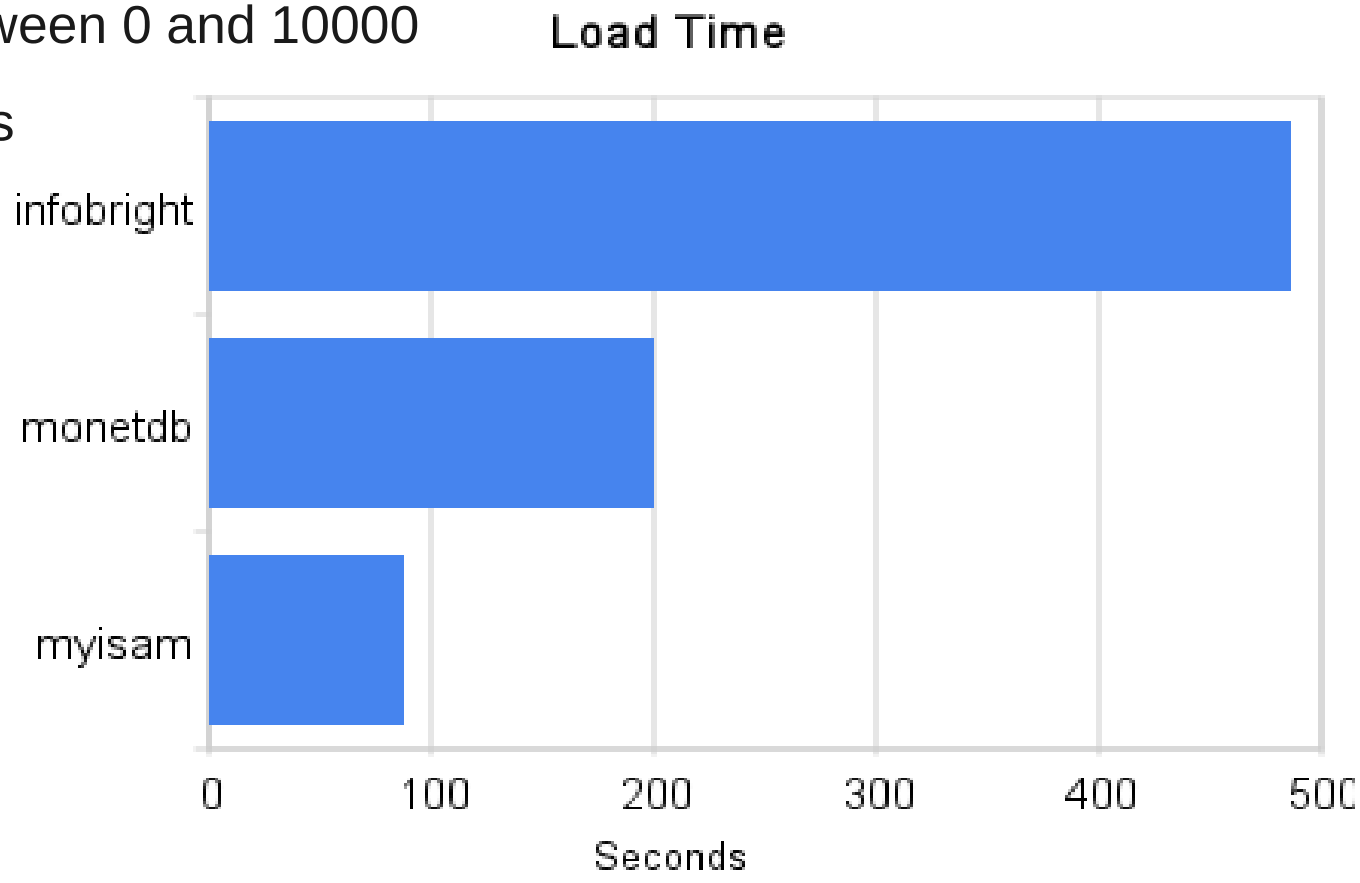
TPC-H

	MonetDB 5.2.3	PostgreSQL 8.2.6	MySQL 5.0.45	MonetDB 5.2.3	PostgreSQL 8.2.6	MySQL 5.0.45	MonetDB 5.2.3	PostgreSQL 8.2.6	MySQL 5.0.45	MonetDB 5.2.3	PostgreSQL 8.2.6	MySQL 5.0.45
SF	0.01			1			2			5		
oad	1097	1734	4409	96466	342103	140888	211	733	303	793	3570	1233
	ms			ms			sec			sec		
1	29	558	366	4004	61253	36305	6	121	80	74	293	201
2	9	15	35	104	3344	152870	1	6	308	88	25	1028
3	12	39	141	935	18504	34137	3	31	91	29	79	421
4	10	18	34	663	3273	3187	1	6	17	21	13	112
5	12	21	223	867	758	19931	3	2	5755	425	13	> 1 hour
6	4	44	156	171	14011	6897	0	23	27	10	75	58
7	13	31	59	902	17395	5967	2	37	2655	47	99	> 1 hour
8	9	35	30	353	17101	2940	121	32	504	58	82	2818
9	14	124	75	855	49769	9233	3	256	57	85	538	> 1 hour
10	12	20	132	692	1014	12424	3	4	91	34	108	344
11	7	21	27	65	2491	39164	1	5	175	56	24	504
12	9	61	74	404	14303	6023	2	27	23	19	77	60
13	47	51	112	5532	6461	8128	20	15	132	40	61	802
14	7	43	301	350	13129	32832	2	27	15193	20	73	> 1 hour
15	6	67	152	110	15093	18023	1	31	49	5	60	113
16	9	46	80	487	8972	9086	1	17	19	26	36	49
17	4	12	10	176	> 1 hour	1042	4	> 1 hour	27	402	>1 hour	185
18	11	92	18	698	35273	> 1 hour	13	125	> 1 hour	21	69	>> 1 hour
19	18	64		1060	15208	297	4	31	50	14	571	> 1 hour
20	12	1810	11	538	> 1 hour	16936	1	> 1 hour	162	17	>1hour	489
21	26	199	27	2483	47391	99153	131	107	1492	188	302	> 1 hour
22	8	420	13	246	> 1 hour	718	3	> 1 hour	1755	84	>1hour	4

ATHLON X2 3800+ (2000mhz) 2 disks in raid 0, 2G main memory

MySQL Performance Blog

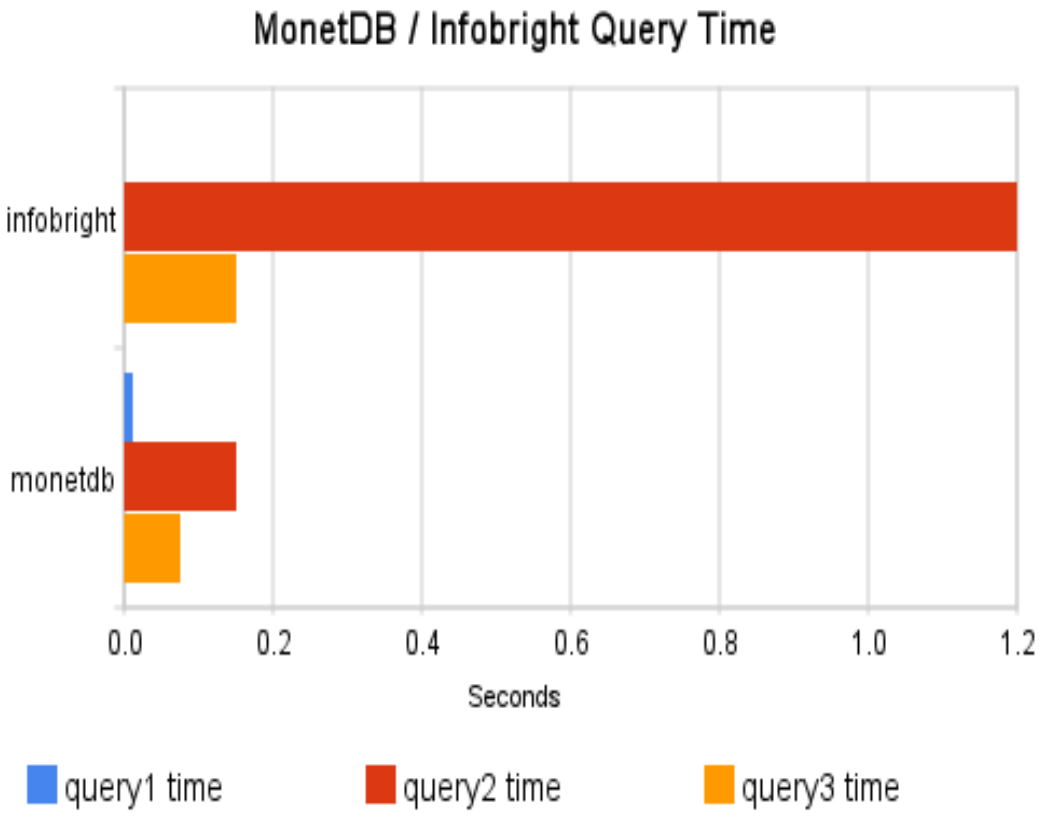
- http://www.mysqlperformanceblog.com/2009/09/29/quick-comparison-of-myisam-infobright-and-monetdb/
- table with 200 integer columns
- random numbers between 0 and 10000
- Loading 1 million rows



MySQL Performance Blog

1. SELECT sum(c19), sum(c89), sum(c129) FROM t;
2. SELECT sum(c19), sum(c89), sum(c129) FROM t WHERE c11> 5;
3. SELECT sum(c19), sum(c89), sum(c129) FROM t WHERE c11 <5;

	myisam	monetdb	infobright
size (bytes)	826000000	829946723	332497242
load time (seconds)	88	200	486
query1 time	3.4	0.012	0.0007
query2 time	3.4	0.15	1.2
query3 time	2.5	0.076	0.15

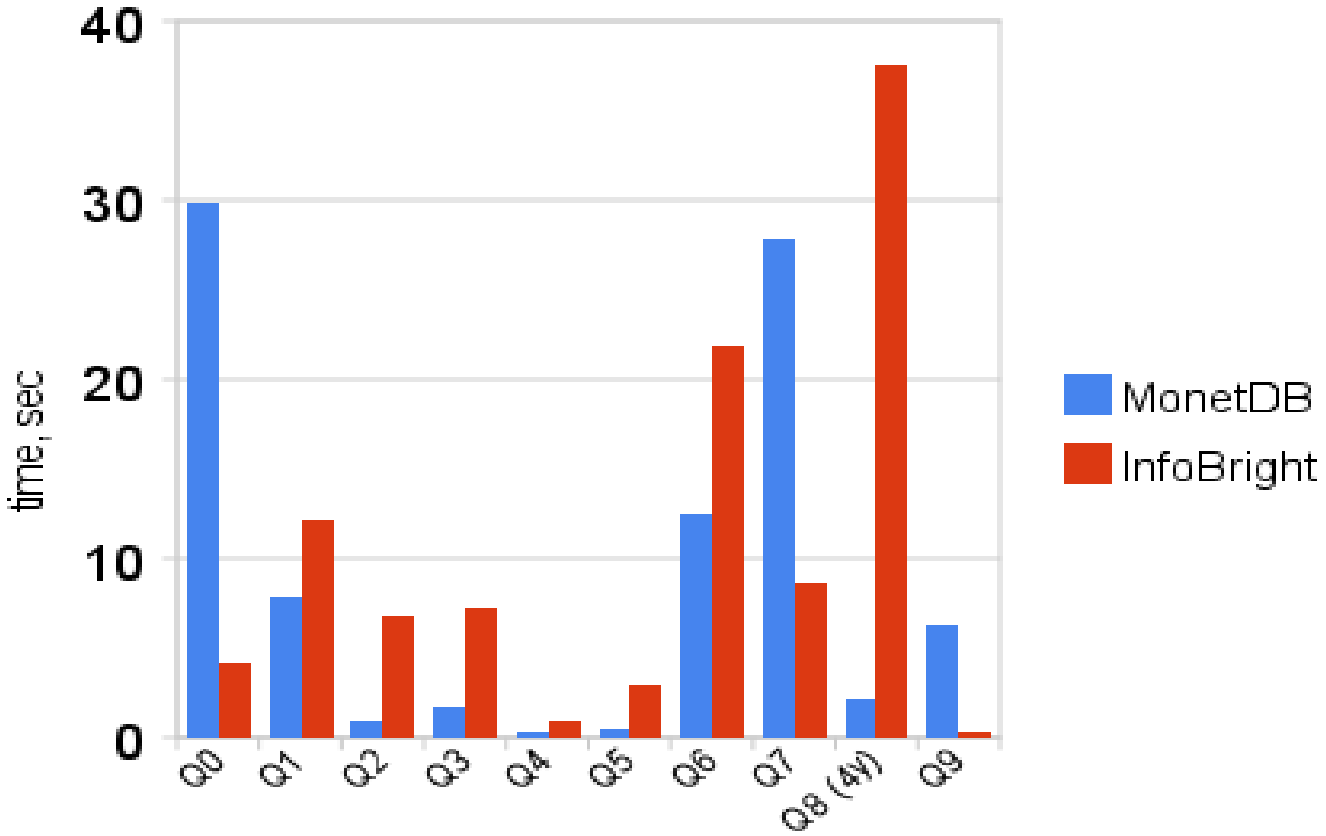


MySQL Performance Blog

http://www.mysqlperformanceblog.com/2009/10/02/analyzing-air-traffic-performance-with-infobright-and-monetdb/

I tried to analyze a bigger dataset and I took public available data http://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236&DB_Short_Name=On-Time about USA domestic flights with information about flight length and delays.

In total raw data is about 55GB.



Open-Source Development

- Feature releases: 3-4 per year
  - Research results
  - User requests
- Bug-fix releases: monthly
- QA
  - Automated nightly testing on >20 platforms
  - Ensure correctness & stability
  - Ensure portability
  - Bug reports become test cases
  - Semi-automatic performance monitoring
  - Passed static code verification by Coverity with only minor problems

sql Stable test Sun Jan 17 10:33:20 CET 2010 - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://monetdb.cwi.nl/testing/projects/monetdb/Stable/sql/.mTests103/index.html

Most Visited Release Notes Fedora Project Red Hat Free Content

clients Stable t... | MonetDB4 Sta... | MonetDB5 Sta... | template/4 Sta... | template/5 Sta... | sql Stable test ... | geom Stable t... | pathfinder/ Sta... | pathfinder/g St...

Testing results of 2010.01.17_00-09-02 (on various platforms):

"stdout stderr": "o" = No differences. "x" = **Minor differences**. "X" = ***Major differences***. "K" = ***Killed***. "T" = ***Timeout***. "S" = ***Socket not released***. "C" = ***Crash***. "-" = No output.

Below: Complete list showing all tests. (*Alternative:* Shortened list showing only tests with unexpected outcome on at least one platform.)

	Debian4.0 G.32.32.d.1	Fedora10 G.32.32.d.1	Fedora10 G.64.32.d.1	Fedora10 G.64.64.d.0	Fedora10 G.64.64.d.1	Fedora10 G.64.64.s.1	Fedora10 I.64.32.d.1	Fedora10 I.64.64.d.1	Fedora11 G.64.64.d.0	Gentoo2.0.1 G.64.64.d.1	SunOS5.10 G.32.32.d.0	SunOS5.10 G.64.32.d.0	SunOS5.10 G.64.64.d.0	SunOS5.11 G.32.32.d.1	SunOS5.11 G.64.64.d.1	SunOS5.11 S.32.32.d.1	Windows5.1 I.32.32.d.1	Windows6.0 I.64.32.d.1	Windows6.0 I.64.64.d.1	Windows6.0 M.32.32.d.1	Windows6.0 M.64.32.d.1	Windows6.0 M.64.64.d.1	
conf	Debian4.0 G.32.32.d.1	Fedora10 G.32.32.d.1	Fedora10 G.64.32.d.1	Fedora10 G.64.64.d.0	Fedora10 G.64.64.d.1	Fedora10 G.64.64.s.1	Fedora10 I.64.32.d.1	Fedora10 I.64.64.d.1	Fedora11 G.64.64.d.0	Gentoo2.0.1 G.64.64.d.1	SunOS5.10 G.32.32.d.0	SunOS5.10 G.64.32.d.0	SunOS5.10 G.64.64.d.0	SunOS5.11 G.32.32.d.1	SunOS5.11 G.64.64.d.1	SunOS5.11 S.32.32.d.1	Windows5.1 I.32.32.d.1	Windows6.0 I.64.32.d.1	Windows6.0 I.64.64.d.1	Windows6.0 M.32.32.d.1	Windows6.0 M.64.32.d.1	Windows6.0 M.64.64.d.1	*conf*
monetdb-sql-conds	o o	- -	o o	o o	o o	- -	o o	o o	o o	X o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	*monetdb-sql-conds*
src/backends/monet5	Debian4.0 G.32.32.d.1	Fedora10 G.32.32.d.1	Fedora10 G.64.32.d.1	Fedora10 G.64.64.d.0	Fedora10 G.64.64.d.1	Fedora10 G.64.64.s.1	Fedora10 I.64.32.d.1	Fedora10 I.64.64.d.1	Fedora11 G.64.64.d.0	Gentoo2.0.1 G.64.64.d.1	SunOS5.10 G.32.32.d.0	SunOS5.10 G.64.32.d.0	SunOS5.10 G.64.64.d.0	SunOS5.11 G.32.32.d.1	SunOS5.11 G.64.64.d.1	SunOS5.11 S.32.32.d.1	Windows5.1 I.32.32.d.1	Windows6.0 I.64.32.d.1	Windows6.0 I.64.64.d.1	Windows6.0 M.32.32.d.1	Windows6.0 M.64.32.d.1	Windows6.0 M.64.64.d.1	src/backends/monet5
optimizers	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	optimizers
Mbeddedsql5--help	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	Mbeddedsql5--help
src/benchmarks/ATIS	Debian4.0 G.32.32.d.1	Fedora10 G.32.32.d.1	Fedora10 G.64.32.d.1	Fedora10 G.64.64.d.0	Fedora10 G.64.64.d.1	Fedora10 G.64.64.s.1	Fedora10 I.64.32.d.1	Fedora10 I.64.64.d.1	Fedora11 G.64.64.d.0	Gentoo2.0.1 G.64.64.d.1	SunOS5.10 G.32.32.d.0	SunOS5.10 G.64.32.d.0	SunOS5.10 G.64.64.d.0	SunOS5.11 G.32.32.d.1	SunOS5.11 G.64.64.d.1	SunOS5.11 S.32.32.d.1	Windows5.1 I.32.32.d.1	Windows6.0 I.64.32.d.1	Windows6.0 I.64.64.d.1	Windows6.0 M.32.32.d.1	Windows6.0 M.64.32.d.1	Windows6.0 M.64.64.d.1	src/benchmarks/ATIS
cache	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	cache
load	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	load
select_simple_join	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	select_simple_join
select_join	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	select_join
select_key_prefix_join	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	select_key_prefix_join
select_distinct	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	select_distinct
select_group	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	select_group
src/benchmarks/arno	Debian4.0 G.32.32.d.1	Fedora10 G.32.32.d.1	Fedora10 G.64.32.d.1	Fedora10 G.64.64.d.0	Fedora10 G.64.64.d.1	Fedora10 G.64.64.s.1	Fedora10 I.64.32.d.1	Fedora10 I.64.64.d.1	Fedora11 G.64.64.d.0	Gentoo2.0.1 G.64.64.d.1	SunOS5.10 G.32.32.d.0	SunOS5.10 G.64.32.d.0	SunOS5.10 G.64.64.d.0	SunOS5.11 G.32.32.d.1	SunOS5.11 G.64.64.d.1	SunOS5.11 S.32.32.d.1	Windows5.1 I.32.32.d.1	Windows6.0 I.64.32.d.1	Windows6.0 I.64.64.d.1	Windows6.0 M.32.32.d.1	Windows6.0 M.64.32.d.1	Windows6.0 M.64.64.d.1	src/benchmarks/arno
create	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	create
check0	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	check0
insert_MODEL	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	insert_MODEL
insert_ATOM	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	insert_ATOM
insert_BOND	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	insert_BOND
check1	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	check1
01	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	01
02	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	02
03	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	03
04	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	04
05	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	05
06	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	06
07	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	07
08	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	08
09	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	09
10	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	10
11	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	11
12	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	12
13	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	13
14	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	14
15	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	15
16	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	16
17	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	17
18	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	18
19	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	19
20	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	20
21	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	21
22	o o	- -	o o	o o	o o	- -	o o	o o	o o	o o	o o	o o	o o	- -	o o	- -	o o	o o	o o	- -	- -	- -	22

Done

Stefan Manegold | 9292ov.nl: OV-reisinfo... | sql Stable make Sun J... | sql Stable test Sun Jan ... | 2 °C 6 °C Sun Jan 17, 16:28:32

Usage

- CWI, Amsterdam:
  - Core DBMS Reseach
  - TIJAH: Multi-Media IR
  - Data Mining, GIS, Astronomy, RDF/SPARQL, Streams, ...
- Data Distilleries (CWI Spin-Off, now part of SPSS -> IBM), Amsterdam:
  - Commercial Data-Mining & CRM Software
  - Many banks & insurance companies in NL
- Knowledge Discovery Lab, UMass, Amherst:
  - Proximity: OpenSource relational knowledge discovery tool
- Universität Tübingen (with UTwente & CWI):
  - Pathfinder: XQuery compiler

Usage

- Dutch National Forensics Institute (NFI):
  - XIRAF: Digital Forensics
- University of Utrecht:
  - Data Mining & Bio-Informatics
- University of Twente:
  - Distributed Object-Oriented Data Management
  - Propabilistic databases
- Open-Source Community:
  - OpenStreetMap: Highperformance API
    - MonetDB, Cherokee, Sphinx, Django
  - Active user community (mailing lists)
  - (many more “anonymously”)
- >10000 downloads per month

File Edit View Go Bookmarks Tools Help

http://cas.sdss.org/dr6/en/

Go

Getting Started | Latest Headlines | MonetDB, Query Pro... | CWI

Sloan Digital Sky Survey / SkyServer

SDSS

Home | Tools | Schema | Projects | Astronomy | SDSS | Contact Us | Download | Site Search | Help

Welcome to the **DR6 site**!!
The Sixth Data Release is dedicated to **Jim Gray** for his fundamental contribution to the SDSS project and the extraordinary energy and passion he shared with everybody!

This website presents data from the Sloan Digital Sky Survey, a project to make a map of a large part of the universe. We would like to show you the beauty of the universe, and share with you our excitement as we build the largest map in the history of the world.

News

The site hosts data from **Data Release 6 (DR6)**. **What's new in DR6**, **what's new on this site**, and **known problems**.
More...

For Astronomers

A separate branch of this website for professional astronomers (English)

More...

SDSS is supported by

Powered by

Microsoft

Site Traffic

Privacy Policy

SkyServer Tools

- Famous places
- Get images
- Visual Tools
- Explore
- Search
- Object upload
- CasJobs

Science Projects

- Basic
- Advanced
- Challenges
- For Kids
- Games and Contests
- Teachers
- Links to other projects

Info Links

- About Astronomy
- About the SDSS
- About the SkyServer
- SDSS Data Release 6
- SDSS Project Website
- Open SkyQuery
- Images of RC3 Galaxies

Help

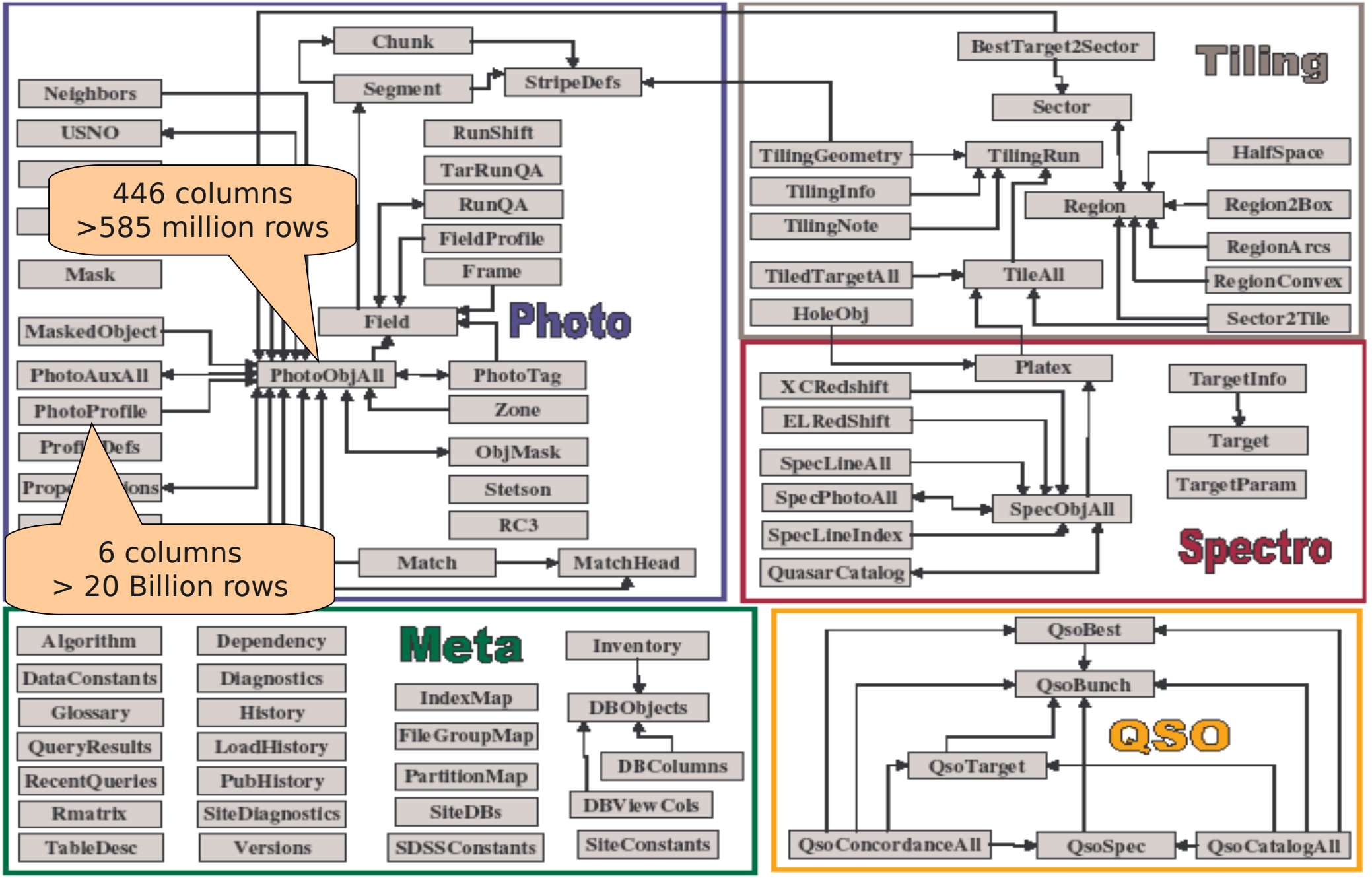
- Getting Started
- FAQ
- How To
- Glossary
- Schema Browser
- Sample SQL Queries
- Details of SDSS Data




Contact Us

Done

SkyServer Schema



Recycler
motivation & idea

"An architecture for recycling intermediates in a column-store". Ivanova, Kersten, Nes, Goncalves. ACM TODS 35(4), Dec. 2010

Motivation:

- scientific databases, data analytics
- Terabytes of data (observational , transactional)
- Prevailing read-only workload
- Ad-hoc queries with commonalities

Background:

- Operator-at-a-time execution paradigm
  - Automatic materialization of intermediates
- Canonical column-store organization
  - Intermediates have reduced dimensionality and finer granularity
  - Simplified overlap analysis

***Recycling idea*:**

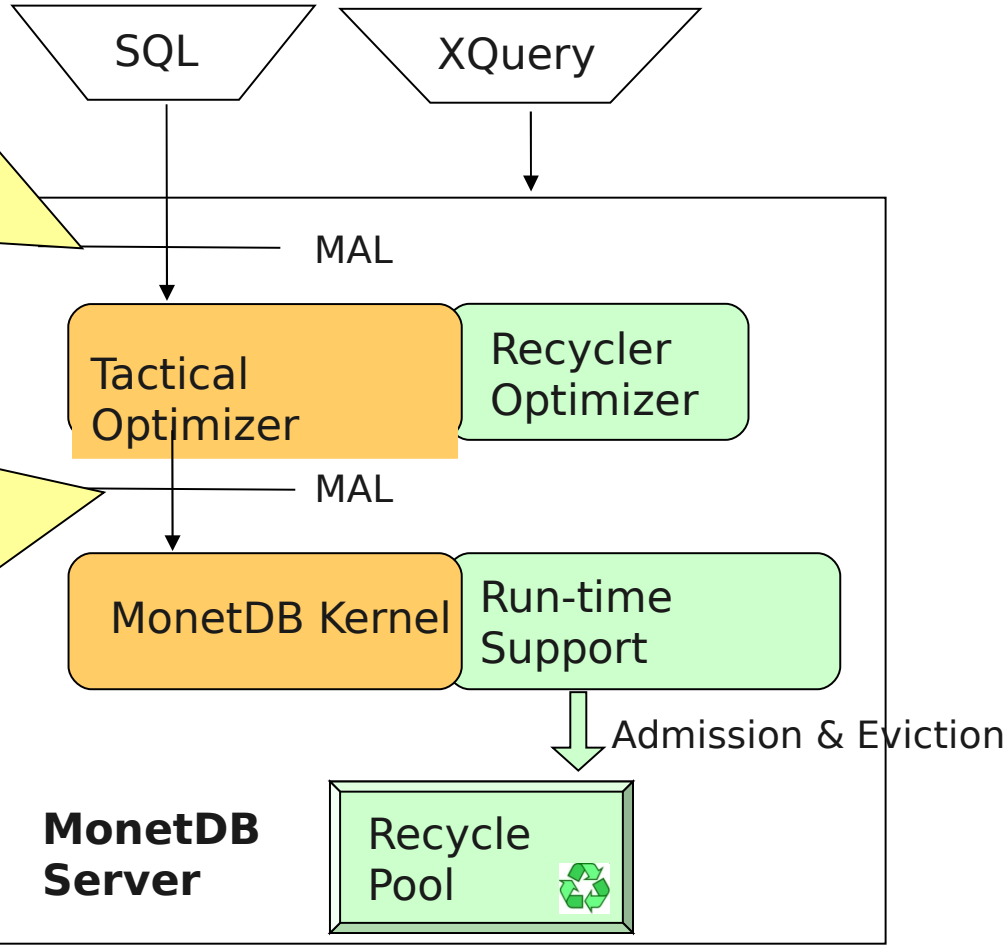
- instead of garbage collecting,
  keep the intermediates and reuse them
  - speed up query streams with commonalities
  - low cost and self-organization

Recycler
fit into MonetDB

"An architecture for recycling intermediates in a column-store". Ivanova, Kersten, Nes, Goncalves. ACM TODS 35(4), Dec. 2010

```
function user.s1_2(A0:date, ...):void;
   X5 := sql.bind("sys","lineitem",...);
   X10 := algebra.select(X5,A0);
   X12 := sql.bindIdx("sys","lineitem",...);
   X15 := algebra.join(X10,X12);
   X25 := mtime.addmonths(A1,A2);
   ...
```

```
function user.s1_2(A0:date, ...):void;
   X5 := sql.bind("sys","lineitem",...);
   X10 := algebra.select(X5,A0);
   X12 := sql.bindIdx("sys","lineitem",...);
   X15 := algebra.join(X10,X12);
   X25 := mtime.addmonths(A1,A2);
   ...
```



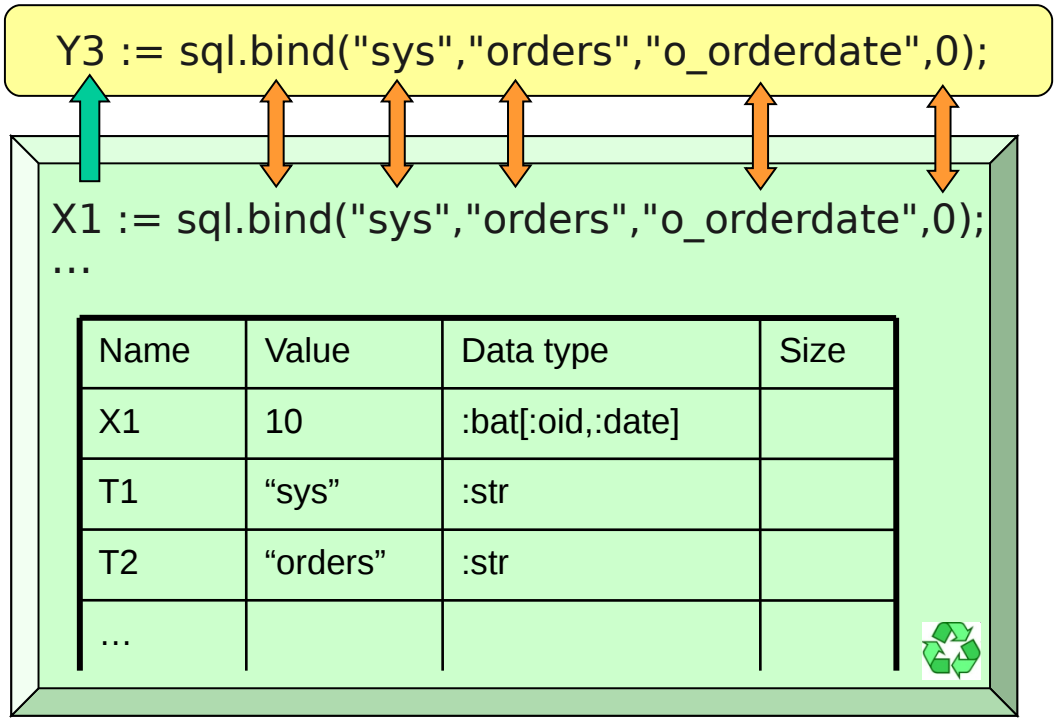
Recycler
instruction matching

"An architecture for recycling intermediates in a column-store". Ivanova, Kersten, Nes, Goncalves. ACM TODS 35(4), Dec. 2010

Run time comparison of

- instruction types
- argument values

Exact matching



Name	Value	Data type	Size
X1	10	:bat[:oid,:date]	
T1	"sys"	:str	
T2	"orders"	:str	
…			

Recycler
instruction subsumption

"An architecture for recycling intermediates in a column-store". Ivanova, Kersten, Nes, Goncalves. ACM TODS 35(4), Dec. 2010

Y3 := algebra.select(X1,20,45);

X3 := algebra.select(X1,10,80);
...
X5 := algebra.select(X1,20,60);

Name	Value	Data type	Size
X1	10	:bat[:oid,:int]	2000
X3	130	:bat[:oid,:int]	700
X5	150	:bat[:oid,:int]	350
...			

Recycler
admission policies

"An architecture for recycling intermediates in a column-store". Ivanova, Kersten, Nes, Goncalves. ACM TODS 35(4), Dec. 2010

Decide about storing the results

- KEEPALL
  - all instructions advised by the optimizer
- CREDIT
  - instructions supplied with credits
  - storage 'paid' with 1 credit
  - reuse returns credits
  - lack of reuse limits admission and resource claims

Recycler
cache policies

"An architecture for recycling intermediates in a column-store". Ivanova, Kersten, Nes, Goncalves. ACM TODS 35(4), Dec. 2010

Decide about eviction of intermediates

- Pick instructions with smallest utility
  - LRU : time of computation or last reuse
  - BENEFIT : estimated contribution to performance: CPU and I/O costs, recycling
- Triggered by resource limitations (memory or entries)

Recycler

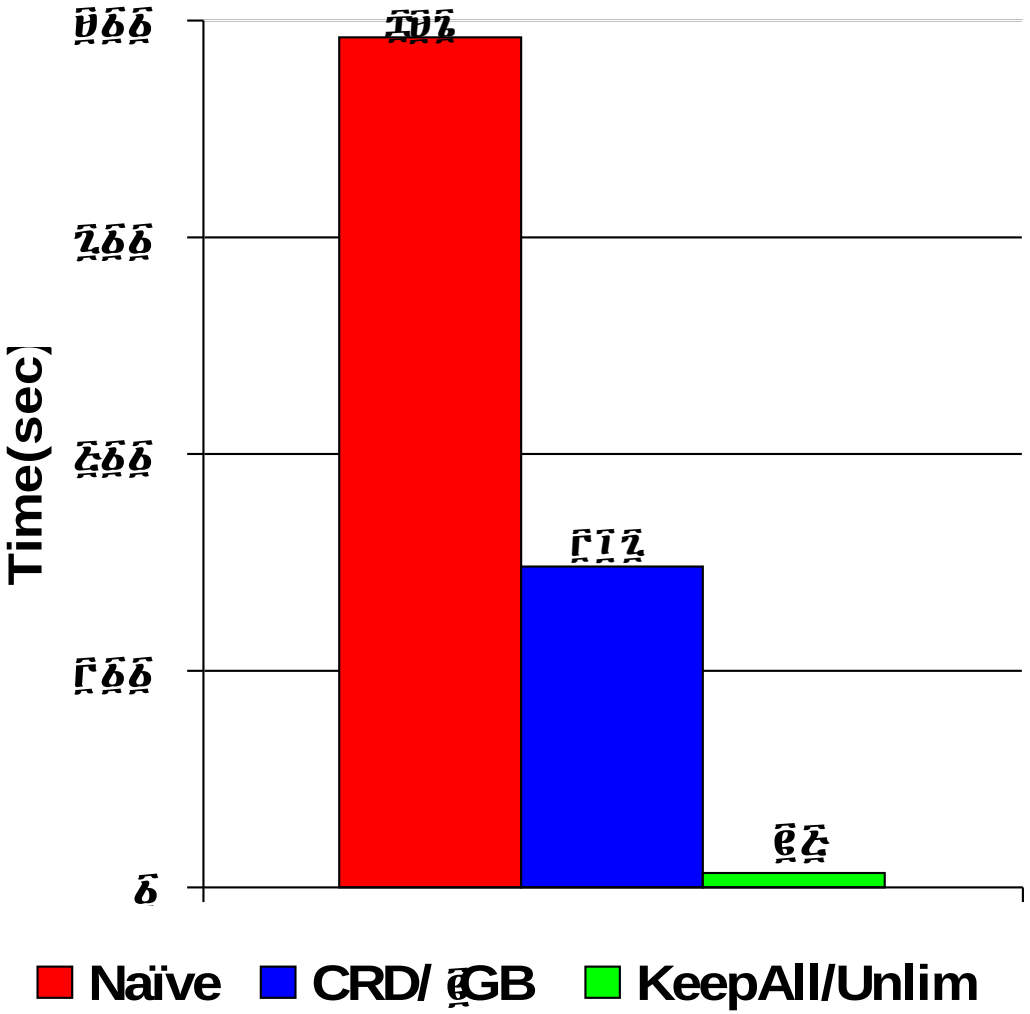
SkyServer evaluation

"An architecture for recycling intermediates in a column-store". Ivanova, Kersten, Nes, Goncalves. ACM TODS 35(4), Dec. 2010

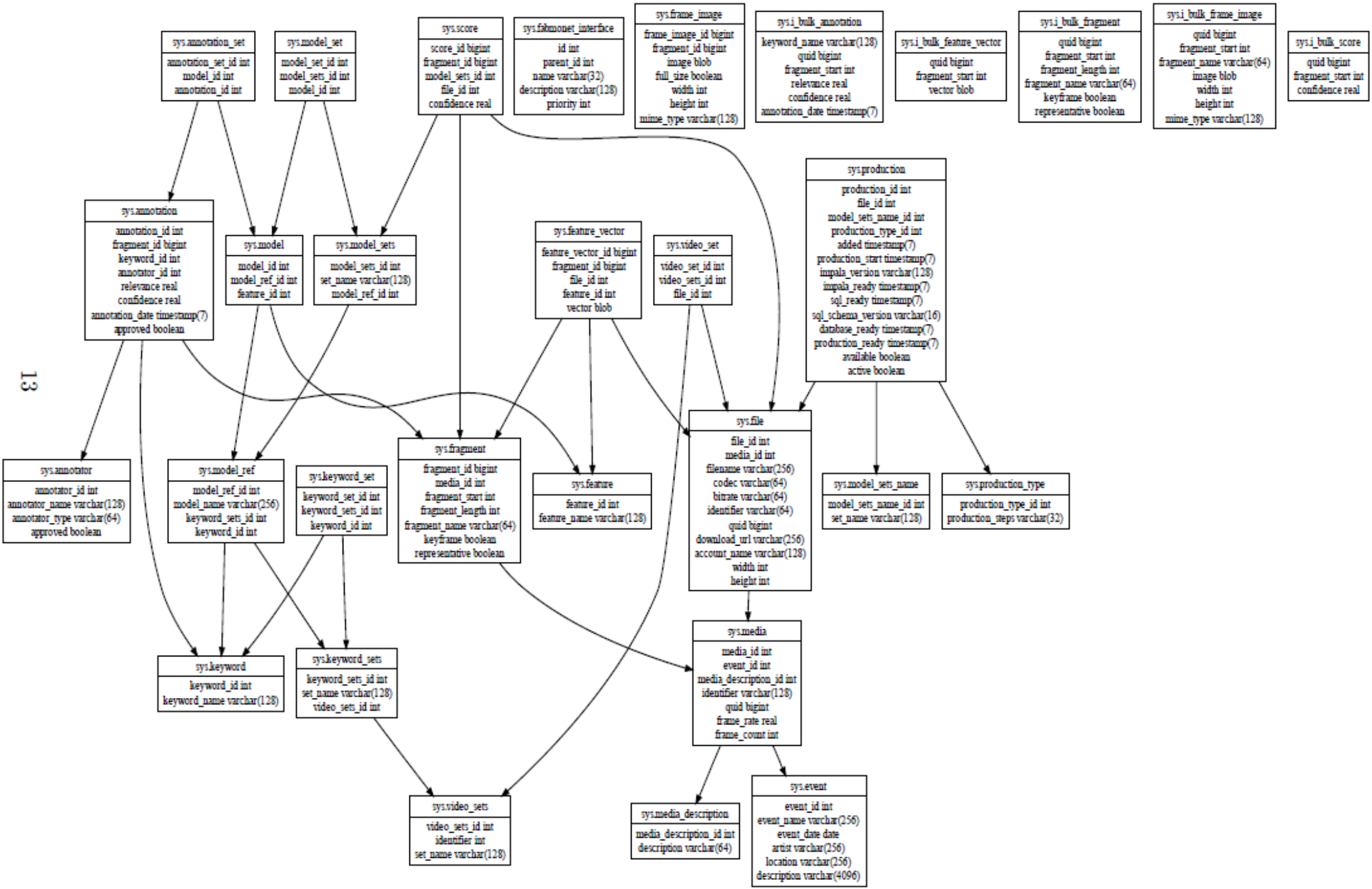
Sloan Digital Sky Survey / SkyServer

http://cas.sdss.org

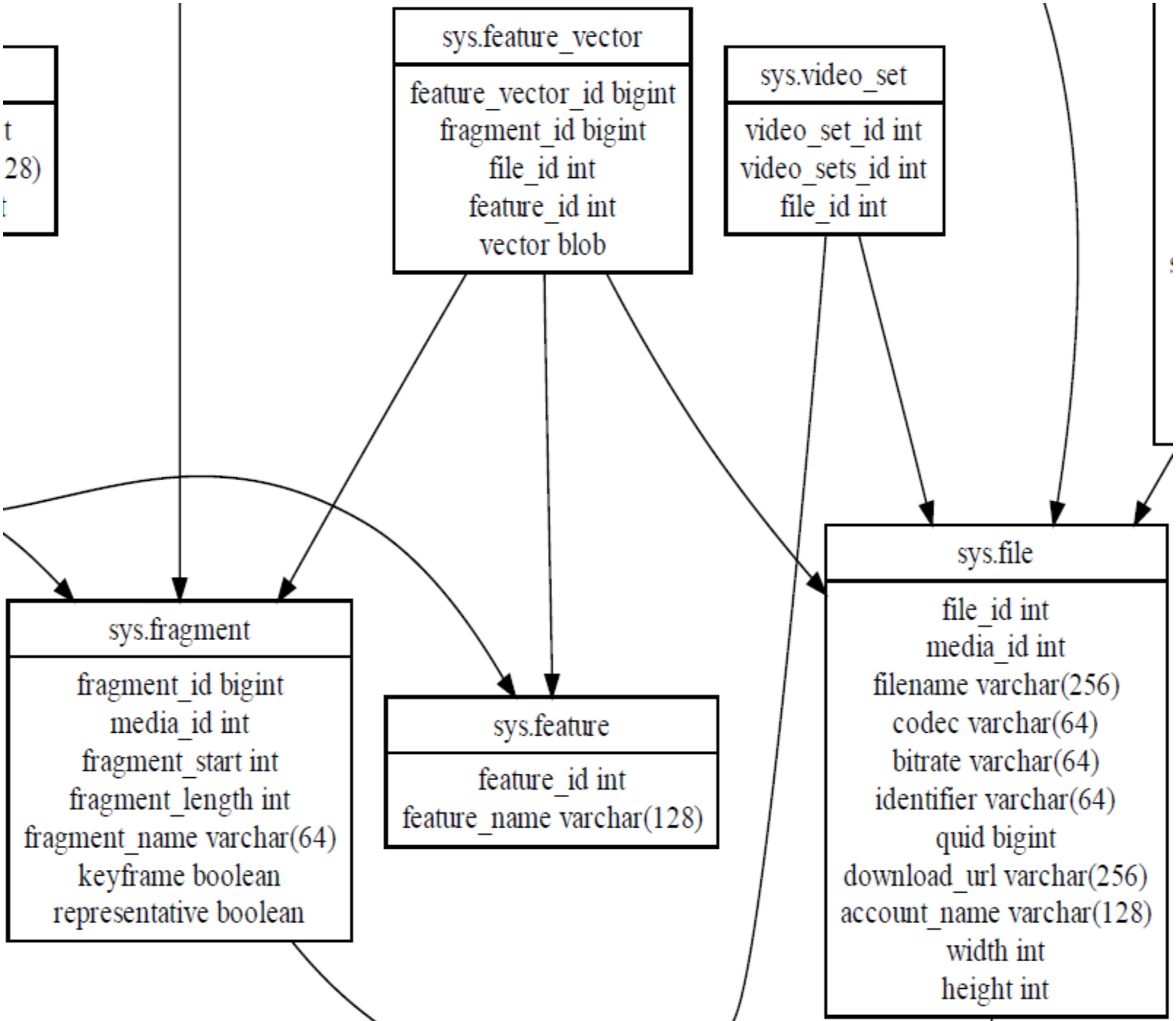
- 100 GB subset of DR4
- 100-query batch from January 2008 log
- 1.5GB intermediates, 99% reuse
- Join intermediates major consumer of memory and major contributor to savings



Schema for Video DB



Schema for Video DB



RDF in MonetDB

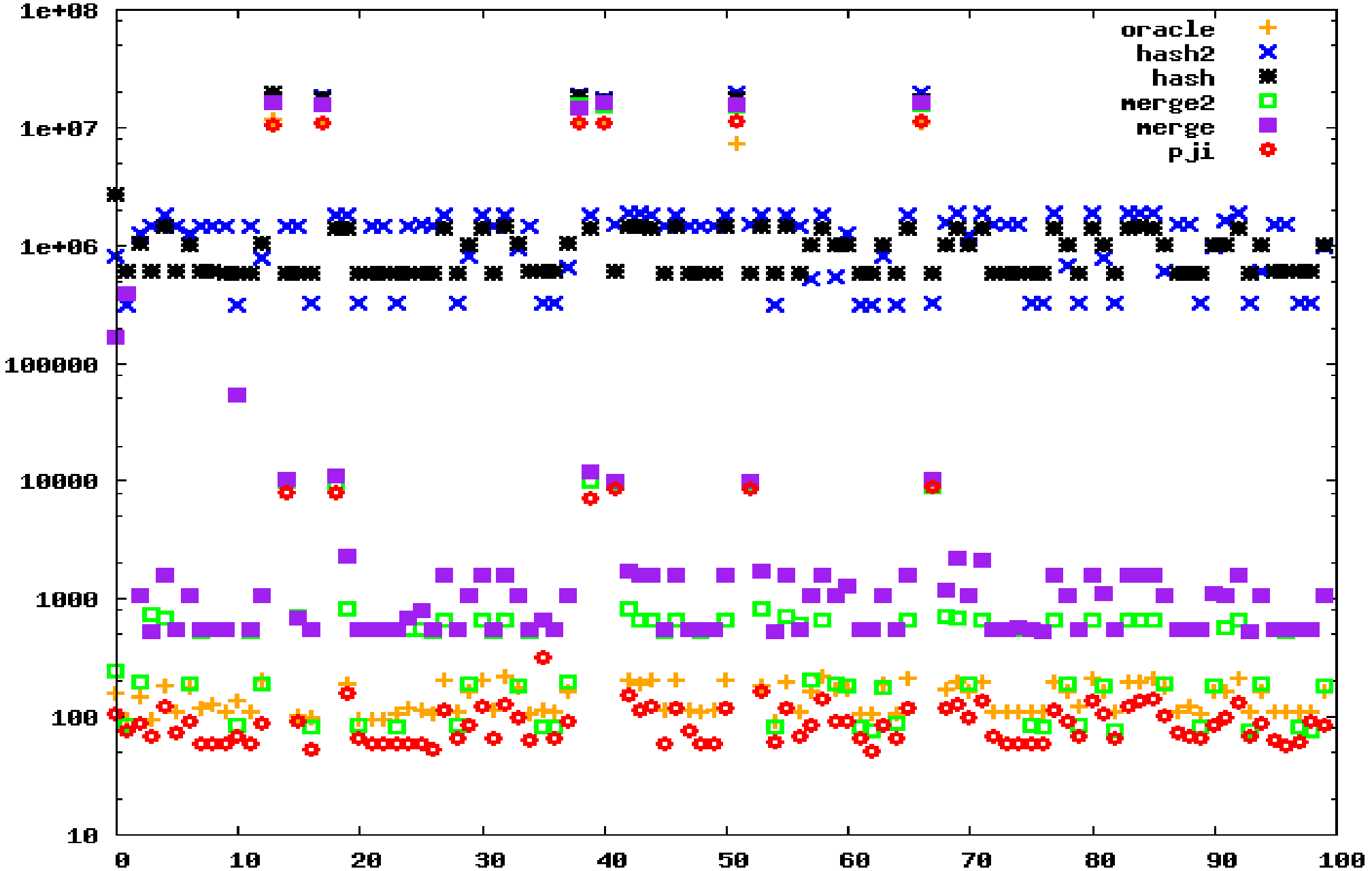
triple store

S	P	O
s[illegible]	p[illegible]	o[illegible]
s[illegible]	p[illegible]	o[illegible]
s[illegible]	p[illegible]	o[illegible]
s[illegible]	p[illegible]	o[illegible]
s[illegible]	p[illegible]	o[illegible]
s[illegible]	p[illegible]	o[illegible]
s[illegible]	p[illegible]	o[illegible]
s[illegible]	p[illegible]	o[illegible]
s[illegible]	p[illegible]	o[illegible]
s[illegible]	p[illegible]	o[illegible]
s[illegible]	p[illegible]	o[illegible]

- Triples store contain only numeric keys
- Keys point to dictionary encoding of URIs & literals
- (transitive closure) queries:
  - Selection (binary search) + series of (self-)joins

RDF in MonetDB

Gene ontology data (~19 M triples)



Cracking

Self Organizing Databases

"Database Cracking", Idreos, Manegold, Kersten, CIDR'07

Idea: each query can be used to re-arrange the data

Build a partial index, tailored to the encountered workload

- *Adapt to query workload as it comes*
- *Create indices/improve locality **during** query processing*
- *Make optimization decisions continuously, at run-time*
- *Let the system do this automatically*

Cracking
how it works

"Updating a cracked database ", Idreos, Manegold, Kersten, SIGMOD'07

Each query is treated as an advice on how data should be stored

The more we crack, the more we learn, the less we touch

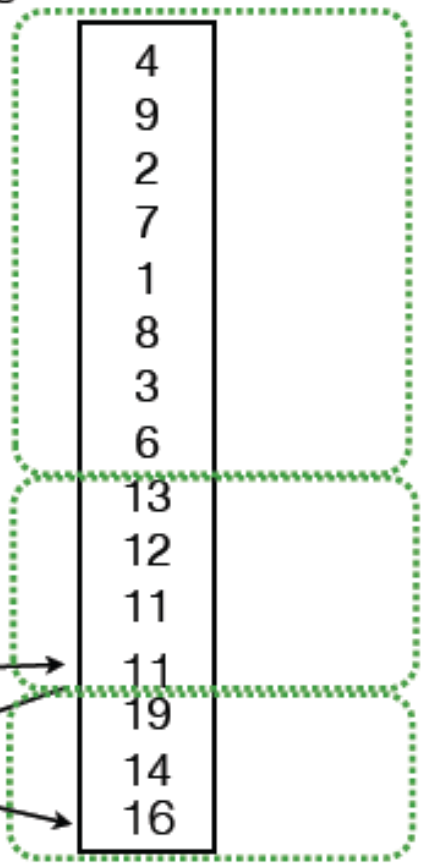
Updates become pending updates
Applied on demand
Goal: minimum physical actions

Pieces are ordered

Values in a piece are not ordered

New Value : 11

16



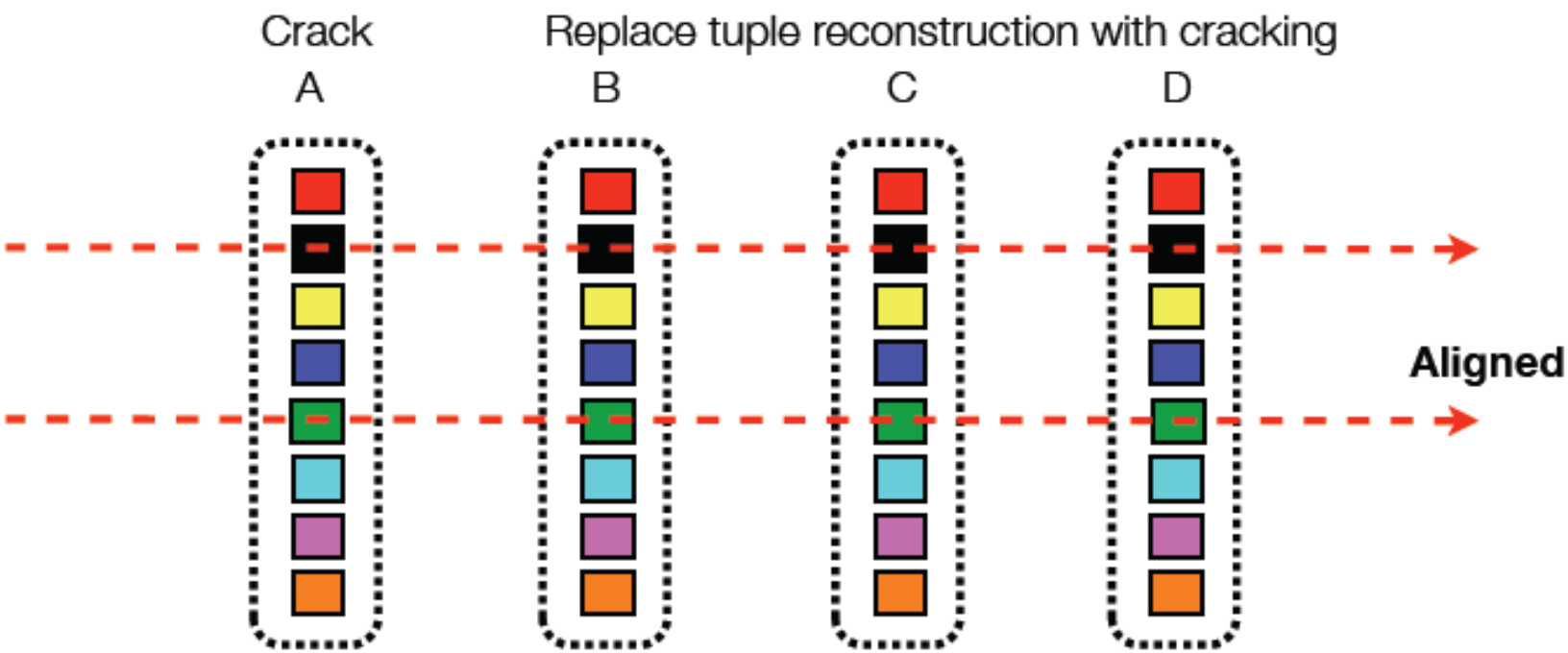
Piece1: A<=10

Piece2: 10<A<14

Piece3: 14<=A

Cracking
sideways cracking in column stores

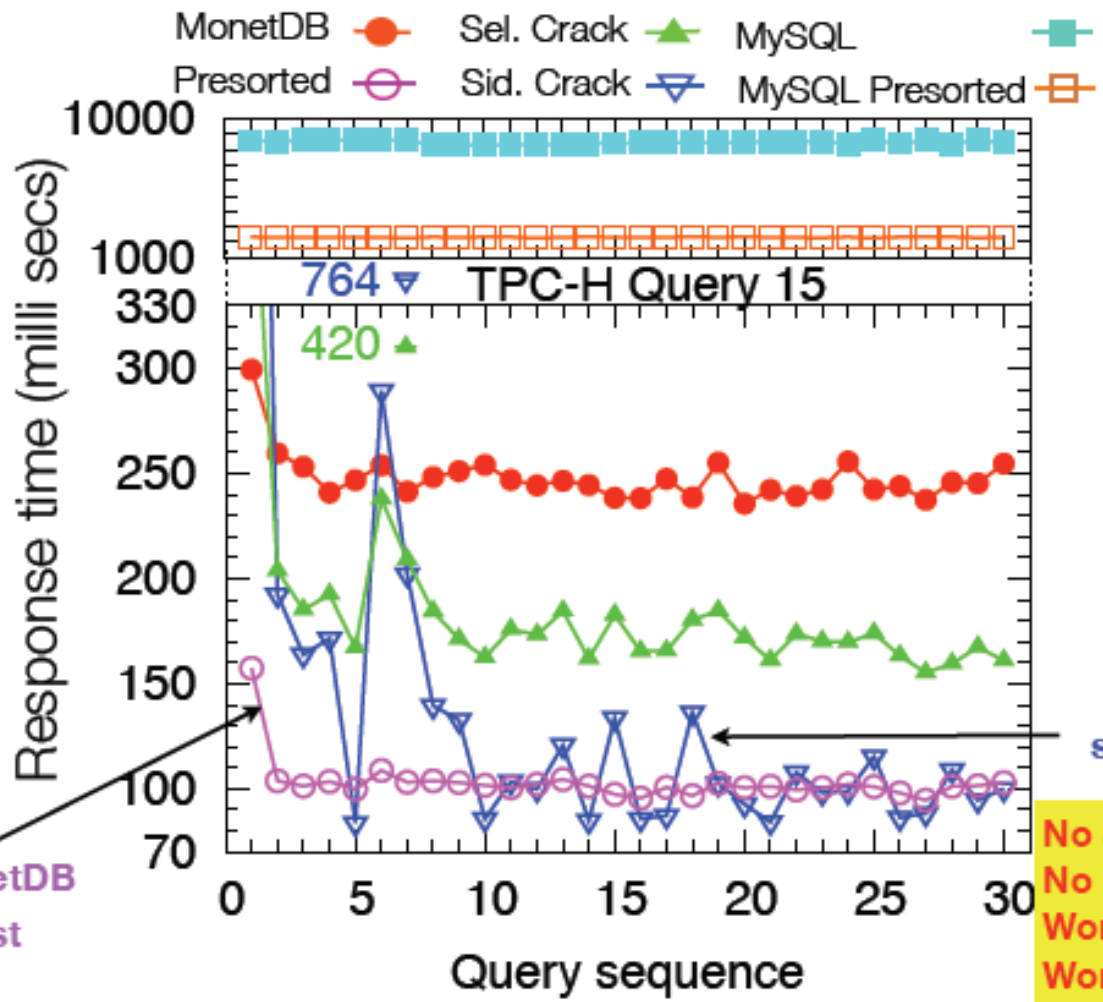
"Self-organizing tuple reconstruction in column-stores", Idreos, Manegold, Kersten, SIGMOD'09



Tuple reconstruction adopts the self-organizing properties of cracking
Everything happens on demand and on the fly
Driven by query needs
Only the (areas of the) columns needed are aligned

Cracking
TPC-H Performance

"Self-organizing tuple reconstruction in column-stores", Idreos, Manegold, Kersten, SIGMOD'09



MonetDB with sideways cracking

Presorted MonetDB
Preparation cost
3-14 minutes

No a priori knowledge needed
No idle time needed
Works for random workloads
Works for frequent updates

Cracking
Future Work

- More operators, joins, aggregation, ...
- External cracking
- Concurrency control
- Compression
- Distribution

Cracking for locality in large RDF graphs