MonetDB:
Reaching the stars step by step

Martin Kersten
Milena Ivanova
Arjen de Rijke
http://www.monetdb.org/

2010 NGSS Sky Survey Data Management Workshop, Edinburgh
“We can't solve problems by using the same kind of thinking we used when we created them.”
The world of column stores
Functionality and performance of MonetDB
Roadmap for a Science Database System
The landscape
The world of column stores

Motivation

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

- Column stores dominate product development since 2008
  - Datawarehouses and business intelligence applications
  - Startups: Infobright, Aster Data, Greenplum, LucidDB,..
  - Commercial: Microsoft, IBM, SAP,..

MonetDB, the pioneer
## The world of column stores

### Workload changes: Transactions (OLTP) vs ...

<table>
<thead>
<tr>
<th>contract</th>
<th>client</th>
<th>date</th>
<th>name</th>
<th>price</th>
<th>city</th>
<th>product</th>
</tr>
</thead>
<tbody>
<tr>
<td>12302346</td>
<td>10042334</td>
<td>Eno</td>
<td></td>
<td></td>
<td>Redmond</td>
<td>Car</td>
</tr>
<tr>
<td>37611373</td>
<td>10987097</td>
<td>Gotz</td>
<td></td>
<td></td>
<td>Berkeley</td>
<td>Redmond</td>
</tr>
<tr>
<td>95371001</td>
<td>10032112</td>
<td>Chen</td>
<td>Seattle</td>
<td>House</td>
<td>lookup query</td>
<td></td>
</tr>
<tr>
<td>51213123</td>
<td>10032423</td>
<td>Jones</td>
<td>Washington</td>
<td>Travel</td>
<td></td>
<td></td>
</tr>
<tr>
<td>54535545</td>
<td>10087823</td>
<td>Smith</td>
<td>New York</td>
<td>House</td>
<td></td>
<td></td>
</tr>
<tr>
<td>45447894</td>
<td>10013232</td>
<td>Doe</td>
<td>Boston</td>
<td>Car</td>
<td>insert query</td>
<td></td>
</tr>
</tbody>
</table>

OLTP queries: access all columns of just one row.
The world of column stores

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

<table>
<thead>
<tr>
<th>contract</th>
<th>client</th>
<th>date</th>
<th>claim</th>
<th>city</th>
<th>product</th>
</tr>
</thead>
</table>

OLAP query: accesses only a few columns of almost all rows.

- select those tuples sold after march 21
- sum claims while grouping by city and product
The world of column stores

Databases hit The Memory Wall


- CPU is 60%-90% idle, waiting for memory:
  - L1 data stalls
  - L1 instruction stalls
  - L2 data stalls
  - TLB stalls
  - Branch mispredictions
  - Resource stalls

![Bar chart showing execution time for different processing types: Ideal, seq. scan, index scan, DSS, OLTP. BUSY and IDLE (STALLED) are compared.](image)
The world of column stores

Hardware Changes: The Memory Wall

Trip to memory = 1000s of instructions!
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)
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
- Design focus on large memory
  - 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 (Binary Association Table)
  • 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, SciQL, RDF/SPARQL
  • front-ends map data models and query languages
How is MonetDB Different

• optimized for large memory hierarchies
  • cache-conscious algorithms
  • exploit the persistence storage (disk, network, SSD)

• 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

Front-ends
- XQuery
- SQL 03
- SciQL
- RDF
- Optimizers

Back-end(s)
- MonetDB 4
- MonetDB 5

Kernel
- MonetDB kernel
Storing Relations in MonetDB

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)

Hash tables, T-trees, R-trees, ...

Various search accelerator structures
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
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
Heavy use of code expansion to reduce cost

1 algebra operator

select()

select(“=“, value) select(“between”, L, H) select (”fcn”, parm)

3 overloaded operators

scan hash-lookup bin-search bin-tree pos-lookup

10 operator algorithms

~1500(!) routines

(macro expansion)

• ~1500 selection routines
• 149 unary operations
• 335 join/group operations
• ...
The MonetDB Software Stack

Front-ends
- XQuery
- SQL 03
- Optimizers

Back-end(s)
- MonetDB 4
- MonetDB 5

Kernel
- MonetDB kernel

Optimizers:
- Strategic optimization
  - MAL
- Tactical optimization: MAL -> MAL rewrites
  - MAL
- Runtime operational optimization
  - MAL
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)
EXPLAIN SELECT a, z FROM t, s WHERE t.c = s.x;

function user.s2_1():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);
_11 := algebra.join(_2, _10);
_13 := algebra.markT(_11,0@0);
_14 := bat.reverse(_13);
_15:bat[:oid,:int]  := sql.bind("sys","t","a",0);
_17 := algebra.leftjoin(_14, _15);
_18 := bat.reverse(_11);
_19 := algebra.markT(_18,0@0);
_20 := bat.reverse(_19);
_21:bat[:oid,:int]  := sql.bind("sys","s","z",0);
_23 := algebra.leftjoin(_20, _21);
exit _73;
_24 := sql.resultSet(2,1,_17);
sql.rsColumn(_24,"sys.t","a","int",32,0,_17);
sql.rsColumn(_24,"sys.s","z","int",32,0,_23);
_33 := io.stdout();
sql.exportResult(_33,_24);
end s2_1;
MonetDB: 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
  - Reordering to optimize intermediate result usage
  - Reordering of linear (projection-) join chains
- Parallelization:
  - Dataflow analysis
  - Horizontal partitioning
  - Remote execution
- Cracking
- Recycling
- ...
MonetDB Front-end: SQL

- SQL 2003
- 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
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.

---

**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
- SkyQuery
- Images of RC3 Galaxies

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

---

**For Astronomers**
- A separate branch of this website for professional astronomers (English)

**More...**

---

**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...
Recycler
motivation & idea

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
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);
...

Run time comparison of

- instruction types
- argument values

Exact matching

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Data type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>10</td>
<td>:bat[:oid,:date]</td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td>“sys”</td>
<td>:str</td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td>“orders”</td>
<td>:str</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Y3 := sql.bind("sys","orders","o_orderdate",0);

X1 := sql.bind("sys","orders","o_orderdate",0);

...
Recycler
instruction subsumption

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Data type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>10</td>
<td>:bat[:oid,:int]</td>
<td>2000</td>
</tr>
<tr>
<td>X3</td>
<td>130</td>
<td>:bat[:oid,:int]</td>
<td>700</td>
</tr>
<tr>
<td>X5</td>
<td>150</td>
<td>:bat[:oid,:int]</td>
<td>350</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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
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)
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

Project portfolio

Commercial: 0.5 PB telco warehouse
Commercial: DNA warehouses
LOFAR: Transient detection
Emili: Streaming in sensor-based systems
TELEIOS: Remote sensing virtual observatory
Planetdata, Lod2: Semantic web, linked open data
NWO: Biased sampling for science database
SciLens: Dissemination and coordination
Datacyclotron: Novel distributed architectures
Remote Direct Memory Access (RDMA)

- Remote Memory at Your Finger Tips.

- RDMA Benefits.
  - CPU Load
  - Reduced Memory Bus Traffic
Road-map for RDMA
The topology.

- Swiss one (LHC)
- Dutch one (DaCy)
The data acceleration.

- A chunk is loaded by a node into the ring.
- It flows clockwise...
- It continuously hops from node to node... until it is removed...
MonetDB SciQL

SciQL (*pronounced ‘cycle’*)
- A backward compatible extension of SQL’03
- Symbiosis of relational and array paradigm
- Flexible structure-based grouping
- Capitalizes the MonetDB array storage
  - Recycling, an adaptive ‘materialized view’
  - Zero-cost attachment contract for cooperative clients
MonetDB Vaults

A contract between MonetDB and file repository of volumeous scientific data

• provide seamless SQL access to foreign file formats using SciQL views

• zero cost, adaptive loading and replication

• Capitalize libraries as UDFs (linpack, R,..)

• Short term targets:
  • MSEED, FITS, NETCDF, csv
MonetDB Octopus

- Distributed SQL processing without a DBA
- Merovingian, managing a cluster of servers
- Cloud-based infrastructure with fail-over
- Partial/full replication adaptive to query needs.
- Recycling, a basis for distribution and load scheduling
eScience- landscape

Data acquisition systems
Data scrubbing, cleaning
Data refinement, enrichment
Data catalogues, meta-data
Data exploration, mining
Data visualisation
MonetDB for Science

Science Domain Workbench

General Workflow Systems

JDBC, ODBC, MAPI, C, Python, Ruby

General Purpose Database Management

Science Exploration Kernels Systems

Mult-cores, SSD, Flash, RDMA, GPU

UDF Library

UDF Library
## Science DBMS

<table>
<thead>
<tr>
<th>Feature</th>
<th>MonetDB 5.23</th>
<th>SciDB 0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open source</td>
<td>Mozilla License</td>
<td>GPL 3.0 + Commercial</td>
</tr>
<tr>
<td>Downloads</td>
<td>&gt;12,000 /month</td>
<td>Tens up to now</td>
</tr>
<tr>
<td>SQL compliance</td>
<td>SQL 2003</td>
<td>??</td>
</tr>
<tr>
<td>Interoperability</td>
<td>JDBC, ODBC, MAPI, C, Python, Ruby, C++</td>
<td>C++ UDF</td>
</tr>
<tr>
<td>Array model</td>
<td>SciQL</td>
<td>AQL</td>
</tr>
<tr>
<td>Science support</td>
<td>Linked libraries</td>
<td>Linked libraries</td>
</tr>
<tr>
<td>Foreign files</td>
<td>Vaults to FITS, NETCDF, MSEED</td>
<td>??</td>
</tr>
<tr>
<td>Distribution</td>
<td>50 node cluster, 200 node cluster, Octopus, Cyclotron</td>
<td>4 node cluster</td>
</tr>
<tr>
<td>Distribution tech</td>
<td>Dynamic partial replication</td>
<td>Static fragmentation</td>
</tr>
<tr>
<td></td>
<td>Distributed query, map-reduce, streaming, multi-core</td>
<td>Map-reduce</td>
</tr>
<tr>
<td>Largest local demo</td>
<td>Skyserver SDSS 6, 3TB</td>
<td>---</td>
</tr>
</tbody>
</table>
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
<table>
<thead>
<tr>
<th>Test Suite</th>
<th>Notes</th>
<th>OS</th>
<th>Processor</th>
<th>gcc/optimization</th>
<th>Languages</th>
<th>Translators</th>
<th>SDL Ver</th>
<th>SDB Ver</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>monodb-api-objc</td>
<td></td>
<td>iOS</td>
<td>A13</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Success</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Android</td>
<td>Snapdragon</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Success</td>
</tr>
<tr>
<td>monodb-api-cocoa</td>
<td></td>
<td>macOS</td>
<td>M1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Success</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Windows</td>
<td>Intel</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Success</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Linux</td>
<td>Intel</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Success</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FreeBSD</td>
<td>AMD</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Success</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Solaris</td>
<td>Sun</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Success</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OpenBSD</td>
<td>Intel</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Success</td>
</tr>
</tbody>
</table>

*Notes: "Success" = Test passed. "Failure" = Test failed. Empty cells indicate tests were not run.*
MonetDB, a full-functional open-source product with a mature modern column-store, has a proven track record in (science) applications. It features a strong and committed development team and close interaction with application developers.

*Reaching the stars step by step*