Enterprise Applications – OLTP and OLAP – Share One Database Architecture

Prof. Dr. Hasso Plattner
History of OLTP and OLAP
Motivation

• Today’s data management systems are separated into transactional and analytical systems storing their data along rows or columns.

• Modern ERP systems are challenged by a mixed workload including OLAP-style queries, e.g.,
  • Dunning-run,
  • Available-to-promise, and
  • Real-time operational reporting
Enterprise Data is Sparse Data

- Many columns are not used even once
- Many columns have a low cardinality of values
- NULL values/default values are dominant
- Sparse distribution facilitates high compression
Sparse Data

55% unused columns per company on average
40% unused columns across all companies

combined distinct value distribution (BKPF, BSAD, BSAK, BSAS, BSID, BSIK, BSIS, VBAP, VBUK, VBUP, GTLO, KNA1, LFC1)
Column Store is Best Suited for Modern CPUs
## Row vs. Column Store

(Compressed)

<table>
<thead>
<tr>
<th>Document Number</th>
<th>Document Date</th>
<th>Sold-To Party</th>
<th>Order Value</th>
<th>Status</th>
<th>Sales Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>95769214</td>
<td>2009-10-01</td>
<td>584</td>
<td>10.24</td>
<td>CLOSED</td>
<td>Germany Frankfurt</td>
</tr>
<tr>
<td>95769215</td>
<td>2009-10-01</td>
<td>1215</td>
<td>124.35</td>
<td>CLOSED</td>
<td>Germany Berlin</td>
</tr>
<tr>
<td>95779216</td>
<td>2009-10-21</td>
<td>584</td>
<td>47.11</td>
<td>OPEN</td>
<td>Germany Berlin</td>
</tr>
<tr>
<td>95779217</td>
<td>2009-10-21</td>
<td>454</td>
<td>21.20</td>
<td>OPEN</td>
<td>Germany Frankfurt</td>
</tr>
</tbody>
</table>

### Row Store

<table>
<thead>
<tr>
<th>Row</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
</table>

### Column Store

- Doc Num
- Doc Date
- Sold-To
- Value
- Status
- Sales Organization

![Diagram of Row Store](image1)
![Diagram of Column Store](image2)
OLTP vs. OLAP Queries

SELECT *
FROM Sales Orders
WHERE Document Number = '95779216'

SELECT SUM(Order Value)
FROM Sales Orders
WHERE Document Date > 2009-01-20

<table>
<thead>
<tr>
<th>Doc Num</th>
<th>Doc Date</th>
<th>Sold-To</th>
<th>Value</th>
<th>Status</th>
<th>Sales Org</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Column Store

Row Store

Row 1
Row 2
Row 3
Row 4
Column Stores for Modern Enterprise Applications

• Single object instance vs. set processing on attributes of nodes of objects

• Enterprise applications perform **set processing** (items for an order, orders for a customer)

• Bring application logic closer to the storage layer using stored procedures
Object Data Guides

- Enterprise systems make heavy use of objects - objects must be mapped to relations
- Often, objects are distributed sparsely over all tables representing nodes
- Relevant tables can now be queried in parallel
- When adding new tables, only add another bit

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>ODG</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Order</td>
<td>(1,1,1,0,0)</td>
</tr>
</tbody>
</table>

1 = table is relevant
0 = table not relevant
Dynamic Views

Presentation Layer

- Excel
- SAP
- Business Objects Explorer
- Any Software

View Layer (Calculations, Filter, ...)

- View
- View
- View
- View

Persistency Layer (Main Memory)

- Node Tables
- Object Hierarchy

Logical Log

DB Persistence

Store

Write Complete Objects

Restart

Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam
Multi-Core Usage
Parallelization in Column Stores

- Columns are optimal for dynamic range partitioning
- One sequential block can be easily split into many (as number of cores) blocks
Stored Procedures

• New enterprise data management requires rethinking of how application logic is written
• Identify common application logic
• Rethink how applications are developed
Claim:
Columnar storage is suited for update-intensive applications
Nowadays Financials

Base Tables

- Accounting Document Header
- Accounting Document Items

Materialized Aggregates

- General Ledger
  - Accounts Payable
  - Accounts Receivable
  - Material Ledger
- Sales Ledger
  - Tax Ledger
  - Fixed Asset
  - Cash Ledger

Materialized Views

- General Ledger Items
  - Accounts Payable Items
  - Accounts Receivable Items
  - Material Ledger Items
  - Dunning
- Sales Ledger Items
  - Tax Ledger Items
  - Fixed Asset Items
  - Cash Ledger Items
  - Payments

Change History

Reporting Cubes

Indices
Simplified Financials System (Target)

Only base tables, algorithms, and some indices
Insert Only

- Tuple visibility indicated by timestamps (POSTGRES-style time-travel*)
- Additional storage requirements can be neglected due to low update frequency (5 – 15%)
- Timestamp columns are not compressed to avoid additional merge costs
- Snapshot isolation
- Application-level locks

Status Updates

- When updates of status fields are changed by replacement, do we need to insert a new version of the tuple?
- Most status fields are binary
- Idea: uncompressed in-place updates with row timestamp

Unpaid \( \leadsto \) Paid

\[ t = \text{NULL} \quad \text{and} \quad t = 2009/06/30 \]
Optimizing Write

- OLTP workload requires many appends
- Instantly applying compression has a severe impact on the performance
- New values are written transactionally safe to a special write optimized storage
- Asynchronous re-compression of all values
- Current binary representation is stored on secondary storage (Flash) for faster recovery
Memory Consumption

- Experiments show a general factor 10 in compression (using dictionary compression and bit-vector encoding)
- Additional storage savings by removing materialized aggregates, save ~2x
- Keep only the active partition of the data in memory (based on fiscal year), save ~5x
- In total 100x is possible
Aging = Partitioning

- Each enterprise object has a dedicated lifecycle - modeled using a state-transition diagram
- Events determine the status of an object
- Map states to partitions
- Multiple partitions = parallel queries
Memory Consumption (contd.)

• Arrays of 100 blades already available
• Next generation of rack servers will allow up to 2TB RAM
• 50 TB main memory will easily allow to cover the majority of SAP Business Suite customers
Customer Study: Dunning Run in < 1s?

• Dunning run determines all open and due invoices
• Customer defined queries on 250M records
• Current system: 20 min
• New logic: 3 sec
  • In-memory column store
  • Parallelized stored procedures
  • Simplified Financials
Why?

• Being able to perform the dunning run in such a short time lowers TCO

• Add more functionality!

• Run other jobs in the meantime! - in a multi-tenancy cloud setup hardware must be used wisely
Next: Hybrid Storage

• **Coarse**-grained hybrid - a single table can be either stored all rows or all columns

• **Fine**-grained hybrid - a single table will be vertically partitioned into groups of columns which are stored independently

• Enterprise workload is mixed workload and the hybrid provides best performance
Hybrid Storage

OLTP

Column Store

Row Store

Hybrid

OLAP
Recovery in On-Demand Systems

- Recovery must be handled differently in on-demand scenarios
- Multiple tenants per system
  - Should all tenants be reloaded at the same time?
  - Prioritization inside a single tenant?
- Use parallelization
Transition

- Millions of “old” unoptimized lines of code at the customers’ site
- Transition required
  - Row-store replacement
  - Part-for-part replacement with bypass
  - Transform row-store to column-store on the fly
- Change of application code
Conclusion

• Technology improvements allow re-thinking of how we build enterprise apps:
  • A combined OLTP and OLAP system can share the same in-memory column store data base
  • Our experiments with real applications and data prove it

• Open research challenges:
  Disaster recovery, extension for unstructured data, life cycle based data management
Outlook