Multi-Model databases
Multi-Model databases

**WHY**
Alternatives, problems

**WHAT**
Definition, basic usage

**HOW**
Internals, pros & cons, tips
Do we need another DB?

There are already a lot of relational, graph, column, RDF, key-value, distributed, SQL, noSQL, newSQL databases (and variation of those)
Trends in DBMS

<table>
<thead>
<tr>
<th>Own</th>
<th>SQL</th>
<th>NoSQL</th>
<th>CQRS</th>
<th>Multi-Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red-Black, binary tree</td>
<td>Tables</td>
<td>Collections</td>
<td>Event, Command, Projection</td>
<td>Relations</td>
</tr>
<tr>
<td>R-tree</td>
<td>Joins</td>
<td>Nesting</td>
<td></td>
<td>Nesting Indexes</td>
</tr>
<tr>
<td>Custom structures</td>
<td>ACID</td>
<td>Scaling</td>
<td>Speed</td>
<td>Choice for structures and ACID</td>
</tr>
<tr>
<td>Speed</td>
<td>Transactions</td>
<td>Async</td>
<td>Reports</td>
<td></td>
</tr>
<tr>
<td>Reinventing the wheel</td>
<td>Sharding</td>
<td>Relations</td>
<td>Technology</td>
<td>Not silver bullet</td>
</tr>
<tr>
<td></td>
<td>Replication</td>
<td></td>
<td>zoo, tools</td>
<td></td>
</tr>
</tbody>
</table>

**WHY**

**HOW**

**WHAT**
Events vs Multimodel

CQRS versus multimodel database

**Events**
- Projection as database
  - ✔ History
  - ✔ Referer
  - Solving conflicts later

**WAL**
- Accessible data
  - ✔ Simple

E.g. on mobile, offline JS
CQRS with Multimodel

Implementation of snapshots in Event database. Made my choice

Snapshots are graph edges

Event has inner structure
Polyglot Persistence

CQRS is very powerful only if data can be split correctly

<table>
<thead>
<tr>
<th>WHAT</th>
<th>HOW</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Session</td>
<td>User Session</td>
</tr>
<tr>
<td>KeyValue</td>
<td>ArangoDB</td>
</tr>
<tr>
<td>Shopping Cart</td>
<td>Shopping Cart</td>
</tr>
<tr>
<td>KeyValue</td>
<td>ArangoDB</td>
</tr>
<tr>
<td>Product Catalog</td>
<td>Product Catalog</td>
</tr>
<tr>
<td>Document</td>
<td>ArangoDB</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Recommendations</td>
</tr>
<tr>
<td>Graph</td>
<td>ArangoDB</td>
</tr>
<tr>
<td>Financial Data</td>
<td>Financial Data</td>
</tr>
<tr>
<td>RDBMS</td>
<td>ArangoDB</td>
</tr>
<tr>
<td>Reporting</td>
<td>Reporting</td>
</tr>
<tr>
<td>Column</td>
<td>PostgreSQL</td>
</tr>
<tr>
<td>Analytics</td>
<td>Analytics</td>
</tr>
<tr>
<td>Column</td>
<td>Cassandra</td>
</tr>
<tr>
<td>User activity log</td>
<td>User activity log</td>
</tr>
<tr>
<td>Column</td>
<td>Cassandra</td>
</tr>
</tbody>
</table>
I chose MMD because

1. **Multiple data structures** and operations used in application (especially graph based)
2. Needed common data pattern for desktop, web, offline frontend and mobile: **common implementation of basic functionality, easy to synchronize**
3. Needed a way to track changes/history (CQRS)
4. Stable and **reliable**
5. Possibilities for **scaling/distributing**
6. Split data by user: **autonomous database**
Definition

... designed to support multiple data models against a single, integrated backend
**Single backend**

“We want to prevent a deadlock where the team is forced to switch the technology in the middle of the project because it doesn’t meet the requirements any longer.”

- Martin Schönert and Frank Celler (ArangoDB)

- Fulltext - Search field could be optimised
- Graph - Services and Projects have bidirectional relation
- Key-value - SEO urls need high performance
- Table - Category has predefined structure, multiple fields
- Nested - Structure of projects differs (Facebook link, services, client, multiple paragraphs)
ORM vs MultiModel

Software structure for easy use, database structure for fast read. ORM work on DB side

Software data structures

Complex change

DB

Simple change

DB
### Multiple models

<table>
<thead>
<tr>
<th>HOW</th>
<th>WHAT</th>
<th>WHY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph</td>
<td>Relations in HDD</td>
<td>Object: Faster development or migration</td>
</tr>
<tr>
<td>Tables</td>
<td>Query language for JOINS</td>
<td>Key-value: Id/hash calculation per cluster</td>
</tr>
<tr>
<td>Events</td>
<td>Append-only, GC later</td>
<td>Object: Faster write on SSD</td>
</tr>
<tr>
<td>Object</td>
<td>Schema for each document</td>
<td>Tables: Query language for JOINS</td>
</tr>
<tr>
<td>Key-value</td>
<td>Id/hash calculation per cluster</td>
<td>Events: Faster development or migration</td>
</tr>
</tbody>
</table>
# Limitation for Multi-Model

<table>
<thead>
<tr>
<th>Why</th>
<th>What</th>
<th>How</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACID</td>
<td>Distributed</td>
<td>Full text</td>
</tr>
<tr>
<td>Transaction must fit in memory</td>
<td>Low stability Depending on other tools</td>
<td>Depending on other tokenizers</td>
</tr>
<tr>
<td>MVCC Multiversion Concurrency Control</td>
<td>Hazelcast Etc</td>
<td>LibCU Lucene</td>
</tr>
</tbody>
</table>
DEMO

1. PHP clients
2. Multimodel: OrientDB, ArangoDB
3. Async MVCC: @id, _id, @version, _rev

https://github.com/aurelijusb/demo-multimodel-databases

Real world query examples:
Conclusion

Single-model
- Solving common problem
- Data structures are stable

Multi-Model
- Exploring new markets
- Relations intensive data

Many (CQRS)
- High load or big data
- Dedicated SysOps / Cloud
Questions?

WHY
Alternatives, problems

WHAT
Definition, basic usage

HOW
Internals, pros & cons, tips

Feedback is always welcome:
https://docs.google.com/forms/d/1qLHPIA4GlZSI5MuBEyFhBMQiTMn4_RtJ89oMbyDrBg/viewform
References and useful links

- https://en.wikipedia.org/wiki/Multi-model_database
- http://orientdb.com/orientdb/
- https://www.arangodb.com/
- https://www.arangodb.com/key-features/
- https://lostechies.com/jimmybogard/2013/06/06/acid-2-0-in-action/
- http://www.slideshare.net/arangodb/multi-model-databases-41917934
- http://www.slideshare.net/LuigiDellAquila/orientdb-time-representation
- https://youtu.be/JHGkaShoyNs?t=57m7s
- https://en.wikipedia.org/wiki/Entity%E2%80%93attribute%E2%80%93value_model
- http://www.jamesserra.com/archive/2015/07/what-is-polyglot-persistence/
- http://de.slideshare.net/MichaelHackstein/multi-modeldatabases