Cloud Polystores
Overview and Open Research Questions

Daniel Glake, Felix Kiehn, Mareike Schmidt
Use Case I: E-Commerce

- Example: Otto Group, Amazon, ...
- What do they need:
  - Availability
  - Consistency only for subset of entities
    - E.g. full consistency for completed orders
    - E.g. partial consistency for ratings and stock
  - High read throughput when querying for products
  - Complex Analytics (e.g. market basket analysis, ...)
Use Case II: Simulation and decision support

- Example: Large scale traffic simulation, Game Engines, ...

- What do they need:
  - High Write throughput (persisting results)
  - Realtime Data Visualization
  - Analytics with aggregated results
  - Support for standardized spatial formats
Use Case III: Medical Data Management

- Example: Hospital, Care facility, research centers, ...

- What do they need:
  - High demand for consistency and data quality
  - Support for multimedia formats
  - Need for full-text search in previous diagnostics
  - Privacy and restricted access to data
  - Availability
Concluded Requirements I

- Multiple kinds of queries
  - Point and range queries
  - Filtering data
  - Full text search
  - ...

- Special complex analytic queries
  - Reading and writing spatiotemporal data
  - Making time series analysis
  - Graph analysis
  - ...
Concluded Requirements II

- Integrity constraints
- Ability to handle multiple data models
- Different degrees of consistency
- Different demands for availability of certain entities or subsets
- Different throughputs
  - High/Low reads
  - High/Low writes
- ...

June 2019
Cloud & Big Data Landscape

- Easy to get lost
- No "one size fits all"
- No standard
- Keeps evolving

Multi-model vs. Multi-modal

Query interface(s)

Multi-model DB System

DS

Graph
Relational
Document
...

Query interface(s)

Multi-modal DB System

DS

Heterogenous domain-specific objects

Graph

Relational
Document
...

Cloud Polystores - Kiehn
20th June 2019
How to solve this with multiple DS?

- Creates Overhead
  - Querying different databases
  - Managing intermediate results
  - Delivering (e.g. sorting) the final results
- Can performance be further enhanced despite the additional overhead?
Federated system vs. Polyglot/-lingual system

Federated DB System

Single query interface

Homogenous data stores

Polyglot/ Polylingual DB System

Multiple query interfaces

Homogenous data stores

DS1  DS2  DS3

DS1  DS2  DS3
MultiStore vs. PolyStore

MultiStore

- Single query interface
- Heterogenous data stores

PolyStore

- Multiple query interfaces
- Heterogenous data stores
Problems and Challenges

- How to decide where to save the data?
- How to do the query planning and processing?
- How to design the schema and optimize it?
- Which query interfaces to provide and how many?
- How to provide full functionality of the underlying systems?
- How to handle schema evolution?
- How to realize extensibility?
- How to enable data stores to communicate to each other and work together?
- What means adaptability?
- How to enable a system to be self-adaptive?
- How to realize logging and recovery over multiple systems?
- How to do internal optimization and benchmarking?
- How to handle different consistency levels within a multi-polystore system?
- How to handle different semantics?
- How can replication and partitioning mechanisms be used?
- Which data models should/can be supported by the system?
- How to enable the user to define requirements for his/her data?
- How to enable the system to learn requirements on its own?
- How to handle CAP/PACELC/BASE over different systems?
- How to handle efficient data migration? (Data is slow!)
- How to carry out distributed transactions over different systems?
- How to provide full functionality of the underlying systems?
- Is there a suitable common data model? How to work with one?

Huge and diverse set of questions and challenges! Thus, focus on subset of problems.
## Explored Systems

<table>
<thead>
<tr>
<th>Design objective</th>
<th>System type</th>
<th>DI architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL++/Forward</td>
<td>Unified relational and JSON-models</td>
<td>Query language and processor</td>
</tr>
<tr>
<td>CloudMdsQL</td>
<td>Query relational and NoSQL stores with Python extension</td>
<td>Query interface</td>
</tr>
<tr>
<td>Estocada</td>
<td>Extensible model integration</td>
<td>Multistore (Delegation system)</td>
</tr>
<tr>
<td>Polybase</td>
<td>Querying Hadoop Cluster over Microsoft-SQL RDBMS</td>
<td>Multistore (HDFS bridge)</td>
</tr>
<tr>
<td>BigDAWG</td>
<td>Unification of relational, NoSQL and NewSQL models</td>
<td>Polystore</td>
</tr>
<tr>
<td>Polypheny-DB</td>
<td>Self-Adapting Polystore</td>
<td>Polystore</td>
</tr>
</tbody>
</table>

Cloud Polystores - Kiehn  
20th June 2019
<table>
<thead>
<tr>
<th>Feature</th>
<th>SQL++/Forward</th>
<th>Cloud MdsQL</th>
<th>Estocada</th>
<th>Polybase</th>
<th>BigDAWG</th>
<th>Polypheny-DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema modeling (mult. DM)</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>(✔️)</td>
<td>?</td>
</tr>
<tr>
<td>Query language</td>
<td>Single</td>
<td>Single</td>
<td>Single (QBT\textsuperscript{XM})</td>
<td>Single (T-SQL)</td>
<td>Multi (SQL, AFL, D4M with SQL)</td>
<td>Multi (SQL, Cypher, CRUD)</td>
</tr>
<tr>
<td>Write operations</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✔️</td>
<td>✗</td>
<td>✔️</td>
</tr>
<tr>
<td>Query optimization</td>
<td>Cost-based</td>
<td>Cost-based</td>
<td>Cost-based</td>
<td>Cost-based</td>
<td>Heuristic</td>
<td>Cost-based</td>
</tr>
<tr>
<td>Query execution</td>
<td>Query splitting</td>
<td>Bind join</td>
<td>View-based rewriting</td>
<td>Query splitting</td>
<td>Query splitting + Data shipping</td>
<td>Query splitting + data shipping</td>
</tr>
<tr>
<td>Semantic</td>
<td>Manual</td>
<td>Hybrid</td>
<td>Fixed</td>
<td>Fixed</td>
<td>Fixed</td>
<td>?</td>
</tr>
<tr>
<td>Feature</td>
<td>SQL++/Forward</td>
<td>Cloud MdsQL</td>
<td>Estocada</td>
<td>Polybase</td>
<td>BigDAWG</td>
<td>Polypheny-DB</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>---------------</td>
<td>-------------</td>
<td>----------</td>
<td>----------</td>
<td>---------</td>
<td>--------------</td>
</tr>
<tr>
<td>Monitoring</td>
<td>❌ ✔</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>(✔)</td>
<td>(Benchmarking)</td>
</tr>
<tr>
<td>Migration</td>
<td>❌ ✗</td>
<td>❌</td>
<td>❌</td>
<td>✔</td>
<td>✔</td>
<td>Online, Offline</td>
</tr>
<tr>
<td>Adaptable Topology</td>
<td>❌ ✔</td>
<td>❌</td>
<td>❌</td>
<td>(Hadoop)</td>
<td>❌</td>
<td>✔</td>
</tr>
<tr>
<td>Automatic Replication</td>
<td>❌ ✗</td>
<td>❌</td>
<td>❌</td>
<td>(✔)</td>
<td>❌</td>
<td>✔</td>
</tr>
<tr>
<td>Automatic Partitioning</td>
<td>❌ ✔</td>
<td>❌</td>
<td>❌</td>
<td>✔</td>
<td>❌</td>
<td>✔</td>
</tr>
<tr>
<td>Logging + Recovery</td>
<td>❌ ✔</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
</tr>
<tr>
<td>Data Models</td>
<td>SQL++/Forward</td>
<td>Cloud MdsQL</td>
<td>Estocada</td>
<td>Polybase</td>
<td>BigDAWG</td>
<td>Polypheny DB</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------</td>
<td>-------------------</td>
<td>---------------------</td>
<td>-------------------</td>
<td>------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>NoSQL*, NewSQL,</td>
<td>Relational,</td>
<td>JSON, Key-Value,</td>
<td>Relational</td>
<td>Array, Relational,</td>
<td>Assoc. Arrays</td>
<td></td>
</tr>
<tr>
<td>Relational</td>
<td>JSON-based, Array</td>
<td>Relational,</td>
<td>JSON-based Relational, XML</td>
<td>Text</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Common Data Model</td>
<td>JSON-based</td>
<td>JSON-based Relational</td>
<td>Relational</td>
<td>Relational</td>
<td></td>
<td>Assoc. Arrays</td>
</tr>
<tr>
<td>Annotations</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>✔</td>
</tr>
</tbody>
</table>

* except graph & key-value
Summary

- Wide variety of use cases with diverse persistency requirements
- No „one fits all“ solution
- Multiple ways of implementing intermediate layer
- A handful of solutions available (5+1 shown)
- Solutions are very rudimentary and tailored for very specific use cases
- We see a need for intelligent and feature-rich Multi-/Polystores
Conclusions for future Multi-/Polystores

- Able to adapt its data store topology
  - Provide “optimal” ecosystem for current requirements
- Routing data to “optimal” data store (Mediation)
  - based on user requirement, user behavior and queries
- Live Migration to adapt to changing topology + requirements
- Using flexible nature of the cloud to orchestrate adaptive topology
Thank you for your attention.
Sources