Evolution Management of Multi-Model Data
(Position Paper)

Irena Holubova
Charles University Prague, Czech Republic

Meike Klettke
University of Rostock, Germany

Uta Störl
University of Applied Sciences Darmstadt, Germany
Motivation /1

- Big Data movement changes many technologies
- Most challenging issue: **Variety** of data
  - Variety within one system (heterogeneous data)
  - Data in multiple types and formats (structured, semi-structured, and unstructured)

→ multi-model systems
→ Polystores
Motivation / 2

"Software aging will occur in all successful products"
(David Lorge Parnas, 1994)

- Continuous redevelopment necessary
- Evolution of the databases in Polystores
  - One of the most challenging tasks
  - In Polystores new requirements arise from the variety of data
Motivation /3

• Avoid: manually written scripts for data migration
Motivation /4

• We need the "alter table" for Polystores
• Lingua franca for all models in Polystores

State of the Art:
• Relational databases: (simple) schema evolution operations available
• XML: some research prototypes are available *(Challenge: Separation of schema and data)*
• NoSQL: first research activities (e.g. Darwin) *(Challenge: No schema information and Variety of data)*
Our Background

• Evolution for NoSQL databases
• Considering variety (within one database) and volume

- Uta Störl, Daniel Müller, Julian Stenzel, Alex Tekleab, Stephane Tolale, Meike Klettke, Steffi Scherzinger: Curating Variational Data in Application Development, Demo presentation, ICDE 2018, Paris
Challenges of Evolution in Polystores

1. Modeling of Multi-Model Data
2. Defining Evolution Operations
   1. Intra vs. Inter-Model Operations
   2. Single-Type vs. Multi-Type Operations
   3. Global vs. Local Evolution Operations
3. Executing Data Migration Operations (Eager, Lazy and Hybrid Approaches)
4. Inference of a Multi-Model Schema
5. ...
1. Modeling of Multi-Model Data

- Requirement: **Unified model** of a Polystore as starting point for evolution
- Object **modeling language** is needed
  - for relational data, objects, documents, graphs, streams, ..
  - representing **links** between different models
- Additionally
  - Translation from global object model into specific model (design)
  - Translation of model changes to a specific model (evolution)

→ **Same methods for design/ evolution**
# Modeling of Multi-Model Data

<table>
<thead>
<tr>
<th>Challenge</th>
<th>State of the Art</th>
<th>Multi-Model Open Issues</th>
</tr>
</thead>
</table>
| Modelling | • Conceptual modeling languages (UML, ER, ...) are tailored for certain models  
• Translations between different models | • Formal definition of **intra-/ inter-model links**  
• **Multi-model conceptual modeling language**  
• Multi-model **schema definition language**  
• Support for **inter-model links, cross-model redundancy** |

- Irena Holubova, Martin Svoboda, Jiaheng Lu: *Unified Management of Multi-Model Data*. ER '19, Salvador, Bahia, Brazil, 2019
2. Schema Evolution Operations

- Schema evolution language for defining evolution operations **in all data models**

- Single-type operations
  - add, delete, rename

- Multi-type operations
  - move/copy
  - split/merge

- Details in the article

```plaintext
evolutionop = typeop | propertyop;

propertyop = add | delete | rename | move | copy;
add = "add" [datatype] property [defaultValue];
delete = "delete" property;
rename = "rename" property "to" pname;
move = "move" property "to" ( kind | property ) complexcond;
copy = "copy" property "to" ( kind | property ) complexcond;

complexcond = "where" joinconds ["and" conds];
joinconds = joincond {"and" joincond};
joincond = property =  "=
manda = cond ["and" cond];
cond = property =  "=
manda;

pname = pname ["as" pname] {"," pname ["as" pname];
property = kind = pname;
kind = [pname = ""] kname;
name = identifier;
mname = identifier;
 pname = identifier;
defaultvalue = value;
```
2.1. Intra vs. Inter-Model Operations

Different kinds of evolution operations:

- **Intra-model operations** are translated into evolution operations within **one model**
- **Inter-model operations** change data in **different models**:
  - Evolving **links** between different models
  - Evolving data which are **redundant in different data models**
  - **Refactoring** a Polystore
    - Reorganization: moving data into another data model
    - Duplication of data (redundant storage for performance reasons)
    - Introduction of new models in a Polystore
## Intra vs. Inter-Model Operations

<table>
<thead>
<tr>
<th>Challenge</th>
<th>State of the Art</th>
<th>Multi-Model Open Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intra-/Inter-Model Operations</td>
<td>• Intra-model operations supported in some, but not all models</td>
<td>• Implementation of <strong>intra-model operations</strong> in all models/systems</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Implementation of <strong>inter-model evolution operations</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Discovering inter-model evolution operations</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Propagate them in a Polystore</td>
</tr>
</tbody>
</table>

- Stefanie Scherzinger, Meike Klettke, and Uta Störl: Managing Schema Evolution in NoSQL Data Store, The 14th International Symposium on Database Programming Languages DBPL@VLDB, Italy, 2013
2.2. Single-type vs. Multi-type Evolution Operations

- **Single-type operations**
  - covering one table/object/document
  - e.g. add, delete, rename

- **Multi-type operations**
  - cover two or more tables/objects/documents
  - e.g. move/copy, or split/merge
  - for complex operations or refactoring within one model
2.3. Global vs. Local Evolution Operations

Evolution operation can be specified
- on the **global model**
- on a certain **local model** of the Polystore

After executing evolutions: **Consistent state** of a Polystore has to be guaranteed
## Global vs. Local Evolution Operations

<table>
<thead>
<tr>
<th>Challenge</th>
<th>State of the Art</th>
<th>Multi-Model Open Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global/Local Operations</td>
<td>• Local single-type evolution operations available in most systems&lt;br&gt;• Multi-type operations (like split, merge, copy, move) only in research prototypes</td>
<td>• Propagation of global operations</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Detection and propagation of local operations</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Discovering schema changes from schema snapshots</td>
</tr>
</tbody>
</table>

![Diagram showing the flow of global and local operations](image-url)
3. Basic Strategies of Data Migration

**Eager Migration**
- after introduction of a new schema version, all datasets are immediately migrated

**Lazy Migration**
- evolution operations are stored and data migration is done on request

**Hybrid Approaches**
- combination of both, intelligent prediction for data migration
# Eager vs. Lazy Migration in Polystores

<table>
<thead>
<tr>
<th>Challenge</th>
<th>State of the Art</th>
<th>Multi-Model Open Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eager/Lazy Migration</td>
<td>➡️ Eager/lazy migration is supported by some systems for single-type operations</td>
<td>➡️ Realizing <em>lazy migration</em> over different models / systems in case of inter-model operations</td>
</tr>
<tr>
<td></td>
<td>➡️ Eager/lazy migration for multi-type operations only in research prototypes</td>
<td>➡️ <strong>Synchronizing eager migration</strong> over different models / systems</td>
</tr>
<tr>
<td></td>
<td></td>
<td>➡️ needs <em>version numbers</em> in all data sets or <em>timestamps</em></td>
</tr>
</tbody>
</table>

- Uta Störl, Alex Tekleab, Meike Klettke, Steffi Scherzinger: *In for a Suprise When Migrating NoSQL Data*, Lightning Talk on the ICDE 2018, Paris
- Andrea Hillenbrand, Maksym Levchenko, Uta Störl, Stefanie Scherzinger, Meike Klettke: *MigCast: Putting a Price Tag on Data Model Evolution in NoSQL Data Stores*. SIGMOD Conference 2019, Amsterdam
4. Schema Inference: Reverse Engineering

- Available for all data models, e.g. JSON
  - Deriving implicit structure
  - Deriving integrity constraints
  - Deriving changes over time (schema versions and evolution operations)
# Schema Inference: Reverse Engineering

<table>
<thead>
<tr>
<th>Challenge</th>
<th>State of the Art</th>
<th>Multi-Model Open Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema Inference</td>
<td>➡️ Single-model inference approaches (XML, JSON, RDF hierarchies, ...)</td>
<td>➡️ Multi-model (i.e., target) schema definition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>➡️ Multi-model inference approaches (heuristic / grammar-inferring)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>➡️ Mutual enrichment of single model subschemas (using, e.g., <strong>inter-model links, redundancy</strong>, ..)</td>
</tr>
</tbody>
</table>

- Meike Klettke, Uta Störl, Stefanie Scherzinger: Schema Extraction and Structural Outlier Detection for NoSQL Data Stores. BTW 2015
- Meike Klettke, Hannes Awolin, Uta Störl, Daniel Müller, Stefanie Scherzinger: Uncovering the Evolution History of Data Lakes, 6th Scalable Cloud Data Management Workshop (SCDM) @ IEEE Big Data Conference, Boston, USA, 2017
- Irena Holubova, Stefanie Scherzinger: Unlocking the Potential of NextGen Multi-Model Databases for Semantic Big Data Projects, SBD@SIGMOD, Amsterdam, Netherlands, 2019
Conclusion and Future Work

• Handling schema evolution and data migration in Polystores
• Most technologies are available for single-model systems
  – Either as state-of-the-art or
  – Research projects
• Most challenging:
  – Adding "inter-model features"
  – Guarantee data consistency for evolution
Literature

Literature