The Theory behind Minimizing Research Data

Problem

Application
- Different applications dealing with growing amounts of data:
  - Research data management with measurement data
  - Sensor data management for smart (assisted) systems aiming at the derivation of activity and intention models by means of Machine Learning algorithms
- Aim: Describing traceability, reconstructibility and replicability of the path from data collection to publication

Aim of our research project
- Reducing the primary measurement or sensor data to an important kernel
- Calculating the kernel even after updating databases or database schemes
- Minimizing the sub-database that has to be stored to guarantee the reproducibility of the performed evaluation

Unification of Provenance and Evolution
- Goal: Performing provenance queries \( Q_{prov} \) after evolution \( \mathcal{E} \) of databases and database schemes
- Idea: Combination of provenance with schema and data evolution
- Wanted: New minimal sub-database to be archived \( J' \subseteq J \)
  - Calculation of a new query \( Q'(I(S_j)) \) from the old query \( Q(I(S_j)) \)

Example
- Schemas: \( S_1, S_2 \) and \( S_3 \)
- Query: \( Q = \text{AVG}(\text{grade}) \)
- Provenance Query \( Q_{prov} \) with input \( K^* \subseteq J \)
- Schema evolution: \( \mathcal{E} \) with minimal sub-database \( J' \subseteq J \)

Calculation of a minimal part of the database (minimal sub-database)
- Different constraints for the sub-database to be determined:
  - Number of tuples of the original relation remains unchanged.
  - The sub-database can be mapped homomorphically to the original database.
  - The sub-database is an intensional description of the original database.
- Question: Which additional information is required to be able to reconstruct the minimal part \( J' \) of the database \( J \) if the result and the evaluation query \( Q \) are both archived?

Example
- Schemas: \( S_1, S_2 \) and \( S_3 \)
- Query: \( Q = \text{AVG}(\text{grade}) \)
- Minimal sub-databases:
  - \( \mathcal{E}(S_1) \subseteq \mathcal{E}(S_1) \) without extension \( K'(S_3) \)
  - \( \mathcal{E}(S_1) = \mathcal{E}(S_1) \) with extension \( K'(S_3) \)
- Provenance Query \( Q_{prov} = \text{AVG}(grade) \)
- Input for \( Q_{prov} \): \( K^* = K(S_j) \)
  - existence of \( a \)
  - result equivalent CHASE-inverse for \( K' \)
  - tp-relaxed CHASE-inverse for \( K' \)
  - exact CHASE-inverse for \( K' \)

CHASE-inverse schema mappings

Types of CHASE-inverses
- CHASE-types:
  - Exact CHASE-inverse: Reconstructs the complete original database
  - Tuple preserving relaxed CHASE-inverse: Preserves the number of tuples
  - Result equivalent CHASE-inverse: \( \text{CHASE}(\mathcal{M}) = \text{CHASE}(\mathcal{M}') \)
  - Reduction result equivalent \( \leq \) relaxed \( \leq \) tp-relaxed \( \leq \) exact
- Conditions for the existence of CHASE inverse:

Combining the techniques
- CHASE:
  - CHASE incorporates dependencies \( \leftrightarrow \) in an object \( \odot \), i.e.
    - \( \text{CHASE}(\odot) = \odot \)
  - Source-to-target tuple-generating dependency (s-t tgd):
    - \( \forall (x) \rightarrow (y) \text{ s.t. } \langle x, y \rangle \)
  - Express the evaluation query \( Q \) as a schema mapping \( M = (S_1, S_2, \Sigma) \) with source and target schemas \( S_1 \) and \( S_2 \) and a set of dependencies \( \Sigma \)
- Provenance Management: traceability of a result back to the relevant original data
- CHASE+BACKCHASE

Schema:
\[ S_1 \rightarrow S_2 \rightarrow S_3 \]

Instance:
\[ \begin{array}{c}
\text{PROV} \\
\text{CHASE} \\
\text{BACKCHASE}
\end{array} \]

Tanja Auge, Andreas Heuer

INSTITUTE OF COMPUTER SCIENCE | DATABASE RESEARCH GROUP | UNIVERSITY OF ROSTOCK, ALBERT-EINSTEIN-Straße 22 | 18059 ROSTOCK, GERMANY