ProSA —
Using the CHASE for Provenance Management

Tanja Auge and Andreas Heuer

University of Rostock, 18051 Rostock, Germany
tanja.auge@uni-rostock.de
andreas.heuer@uni-rostock.de
https://dbis.informatik.uni-rostock.de

Abstract. Collecting, storing, tracking, and archiving scientific data is the main task of research data management, being the basis for scientific evaluations. In addition to the evaluation (a complex query in the case of structured databases) and the result itself, the important part of the original database used has also to be archived. To ensure reproducible and replicable research, the evaluation queries can be processed again at a later point in time in order to reproduce the result. Being able to calculate the origin of an evaluation is the main problem in provenance management, particularly in ***why*** and ***how*** data provenance. We are developing a tool called ProSA which combines data provenance and schema / data evolution using the CHASE for the different database transformations needed. Besides describing the main ideas of ProSA, another focus of this paper is the concrete use of our CHASE tool ChaTEAU for invertible query evaluation.

Keywords: theoretical foundations of databases · data curation · annotation · provenance · temporal databases · CHASE

1 Introduction

Collecting, evaluating, analyzing, archiving, and publishing research data are the main tasks of research data management. Research institutes all over the world are producing research data in huge amounts. The processing, analysis, and storage of a huge amount of data, which is usually generated in research, can be considerably supported by the use of *data provenance*. With the combination of the CHASE – a universal tool for transforming databases or queries in database systems – and data provenance, a minimal sub-database of an original research dataset can be computed which is one of the main problems in minimizing research data [6]. So questions like (1) Where does the data come from? (2) Why this result? and (3) How is the result being computed? can be answered in research data management [13, 28]. As a use case, we apply our results in research data management and data provenance within a joint project of the University of Rostock and the Leibniz Institute for Baltic Sea Research Warnemünde (IOW) [11, 12].

In this paper, we will introduce our data provenance tool called ProSA, a tool for combining data provenance, schema and data evolution by means of the CHASE in the case of invertible query evaluation (Section 5). The CHASE as a basic tool is implemented in ChaTEAU (Section 4). Besides the presentation of the general techniques behind ChaTEAU, we will introduce different applications of the CHASE, all being research topics within our Database Research Group at the University of Rostock (DBIS). These applications are *invertible query evaluation* (Section 3), *semantic query optimization* and *answering queries using operators* (Section 6). First, however, basic Notions and the State of the Art in the research areas research data management, CHASE and data provenance are introduced in the following Section 2.

2 Basic Notions and State of the Art

In our data provenance tool ProSA, we are using the CHASE technique to represent all of the database transformations needed for research data management, specifically evaluation queries against the original research data, schema and data evolution operations, as well as data exchange processes (between different research repositories). Therefore, we first introduce the CHASE as a general tool for different database problems. We close this section by introducing the main concepts of data provenance.

2.1 The CHASE algorithm

Semantic query optimization, answering queries using views (AQuV), data exchange and integration, cleaning as well as invertible query evaluation are basic problems in database theory that can be solved using the CHASE. For this reason, different versions of the algorithm itself as well as various tools that implement or execute the CHASE have been developed over time. Each tool such as Llunatic, PDQ or ProvCB is specifically designed for one area of application like data cleaning, semantic optimization and AQuV (see Table 1).

Hence, the idea of this universal tool can be summarized as follows: For an object $\bigcirc$ (e.g. a database, or a query) and a set of dependencies $\star$ (e.g. a set of FDs and JDs) the CHASE incorporates $\star$ into $\bigcirc$, so that $\star$ is implicitly contained in $\bigcirc$. We represent this by:

$$\text{chase}_{\star}(\bigcirc) = \circledast.$$

Incorporating parameters $\star$ into an object $\bigcirc$ can mean different things. We therefore distinguish between two cases:

1. When a CHASE incorporates dependencies into a query, all interpretations of variables of a query (e.g., in a predicate calculus notation, replacing variables by attribute values) will always satisfy the functional dependencies given;
2. When a CHASE incorporates dependencies into a database with marked null values, the CHASEd database will satisfy all of these dependencies, sometimes replacing null values by attribute values.

	Parameter $\star$	**Object** $\bigcirc$	**Result** $\circledast$	**Goal**
0.	dependency	database schema	database schema with integrity constraint	optimized database design
I.	dependency	query	query	semantic optimization
II.	view	query	query using views	AQuV
II'.	operator	query	query using given operators	AQuO
III.	s-t tgds, egds	source database	target database	data exchange, data integration
IV.	tgds, egds	database	modified database	cleaning
V.	tgds, egds	incomplete database	query result	certain answers
VI.	s-t tgds, egds, tgds	database	query result	invertible query evaluation

Table 1. Overview of CHASE variants

Database structures or Queries as a CHASE object The original idea of the CHASE was to include dependencies like *functional dependencies* (FD) $X \rightarrow Y$ or *join dependencies* (JD) $\bowtie [R_1, ..., R_n]$ into a given database schema [1, 33]. The result, a database schema with integrity constraints, led to an *optimized database design* (Table 1, case 0) by guaranteeing some desirable database design properties such as the lossless join property. *Semantic query optimization*, on the other hand, requires the incorporation of dependencies into a given query. One of the already existing CHASE tools PDQ [9] has been developed for this purpose. For optimization, the CHASEd query has to be transformed to an equivalent, optimal query by using an additional BACKCHASE phase [15].

Answering Queries using Views (AQuV) and its generalization *answering queries using capabilities of database engines* that we call *Answering Queries using Operators* (AQuO, see Section 6) are also possible goals that can be realized with the CHASE. Here a set of views (or operators) will be incorporated as the CHASE parameters $\star$ into a given query, the CHASE object $\bigcirc$ (case II. and II.'). Finding an efficient method of answering a query using a set of previously defined materialized views over the database, rather than accessing the database relations is discussed in [27, 14]. Here a given query Q is enriched by its views (CHASE phase) and then reduced to the minimum equivalent view: The CHASE will therefore be extended by a BACKCHASE phase, again. One of the tools for efficiently[1] calculating CHASE&BACKCHASE applied to AQuV is ProvCB [29].

Database instances with null values as a CHASE object Instead of queries as CHASE objects $\bigcirc$ we can also use databases with null values as $\bigcirc$. Processing *source-to-target tuple generating dependencies* (s-t tgd's,[16]) using

[1] By the use of Provenance information.

the CHASE is implemented by LLUNATIC [19] and ChaseFUN [10]. S-t tgd's are formulas

$$\forall \mathbf{x} : (\phi(\mathbf{x}) \rightarrow \exists \mathbf{y} : \psi(\mathbf{x}, \mathbf{y}))$$

where ϕ is a function over source predicates (e.g., relation schemes of the source database), ψ is a function over target predicates (e.g., relation schemes of the target database), $\mathbf{x}$ being domain variables over attributes in the source, and $\mathbf{y}$ being domain variables over (new, additional) attributes in the target. While the CHASE creates a new target database using s-t tgd's as some kind of inter-database dependencies, *tuple generating dependency* (tgd), a s-t tgds on the same database, and *equality generating dependencies* (egd)

$$\forall \mathbf{x} : (\phi(\mathbf{x}) \rightarrow (x_1 = x_2))$$

can be seen as intra-database dependencies representing integrity constraints within a database [7, 16]. Egd's replace marked null values by other marked null values or attribute values (also called constants) [16, 18].

LLUNATIC and ChaseFUN support data exchange and data integration scenarios as well as database cleaning tasks (case III. and IV.). For the goal of invertible query evaluations, no known tool yet exists (case VI.). In this case, the CHASE result $\circledast$ is not a (new, integrated or exchanged) database but the result of a database query. The BACKCHASE phase is afterwards used to calculate an inverse of the query or to generate a provenance polynomial [21, 22] to be able to determine a (minimal) sub-database that guarantees the reproducibility of the query result.

The CHASE in general The CHASE applications we are using in our research projects at the University of Rostock are shown as red rows in Table 1. The main application *Invertible Query Evaluation* of the project *ProSA* will be described in Section 3. The other CHASE applications of the Data Research Group (DBIS) of the University of Rostock are summarized in Section 6. A generalization of CHASE and the development of a general chase tool is therefore a common task for several, different research projects.

A first idea of a universal CHASE tool is given in [8]. Here the authors tested and compared different CHASE tools like PDQ, ChaseFUN and LLUNATIC. They tested for different CHASE strategies like *oblivious CHASE*, *standard/restricted CHASE* and *core CHASE* [20]. The concept of the standard CHASE is shown in Algorithm 1. Eliminating the first If-statement (h is an active trigger) transforms this algorithm to the oblivious CHASE, the core chase is neglected here.

Basically, for a given database instance I, Algorithm 1 provides a modified database I' by incorporating any possible dependency $\sigma \in \Sigma$ into I. While tgd's create new tuples under active trigger, egd's clean the database by replacing null values where an *active trigger* h is a homomorphism from the left-hand side of σ to I which cannot be extended to a homomorphism from the right-hand side to I. In other words, a trigger is active, if a new tuple can (and has to) be generated or null values be replaced.

Algorithm 1 standard CHASE (Σ, I)

Require: Set of dependencies Σ, Database instance I
Ensure: Modified database I'

```
for all trigger h for a dependency σ ∈ Σ do
  if h is an active trigger then
    if σ is a tgd then
      Adding new tuples to the database instance I
    else if σ is an egd then
      if values compared are different constants then
        CHASE fails
      else
        Substituting null values by other null values or constants
```

The problem of this basic algorithm is that the CHASE parameter $\star$ is fixed (a set of dependencies Σ) and the CHASE object $\bigcirc$ is also fixed (a database instance I). In our approach ChaTEAU, we will extend the CHASE to become applicable to other CHASE parameters (such as views, query operators, or privacy constraints, see [24, 25]) and other CHASE objects (such as queries and general database transformations).

In Section 3, we will focus on the use of the CHASE for calculating inverses of an evaluation query in research data management, to determine the origin of the evaluation results. This is one of the main tasks in data provenance. We therefore will introduce some basic notions of data provenance in the next subsection.

2.2 Data Provenance

Given a database instance I and a query Q, the central task in data management is to compute the result of the query $K = Q(I)$. However, in many cases the result of the query itself is not sufficient. We are interested, for example, in:

1. Where do the result tuples come from?
2. Why – by means of which parts of the original database I – was a certain result achieved?
3. How – by means of which sequence of operations – was a result actually achieved?
4. Why is an expected value missing in the result?

Thus, in *Data Provenance* we typically distinguish between four Provenance queries (***where***, ***why***, ***how***, and ***why not***) and four Provenance answers (extensional, intensional, query-based, modification-based). In the remainder of this paper, we focus on ***how***- and ***why***- provenance, and on extensional answers to these queries. An extensional answer to a ***why***-query is the sub-database of the original research database I, the result tuples are derived from. The tuples of this sub-database of I are called *witnesses* for the evaluation result K.

The ***why***- and ***where***-provenance can be derived from the result of the ***how***-provenance (see Figure 1). For this we can define a reduction based on the

information content

$$\textbf{\textit{where}} \preceq \textbf{\textit{why}} \preceq \textbf{\textit{how}}.$$

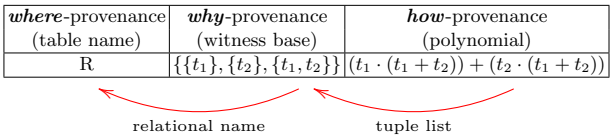


Fig. 1. Reduction of ***why*** and ***where*** from ***how***

In our research project ProSA, we mainly focus on ***why***- and ***how***-provenance with extensional answers to the provenance queries, which are given by *provenance polynomials* [21, 3] and *(minimal) witness bases* [13] to guarantee the reproducibility of the query result. The polynomials are defined by a commutative semi-ring $(\mathbb{N}[X], +, \cdot, 0, 1)$ with $+$ for union and projection as well as $\cdot$ for natural join. The selection does not change a tuple, hence it is not represented in the semi-ring. Projection and union both eliminate duplicate tuples, they are therefore represented by an operation in the semi-ring. The second operation $\cdot$ symbolizes the natural join, representing the combination of tuples. The polynomials apply to the positive relational algebra (i.e., without difference or negation), but approaches for the extension by aggregate functions and negation already exist [3, 2].

We are interested in specifying a concrete calculation rule (***how***), or at least in specifying all necessary source tuples (***why***). For the given example in Figure 1 the provenance polynomial is calculated as $(t_1 \cdot (t_1 + t_2)) + (t_2 \cdot (t_1 + t_2))$, where t_1, t_2 and t_3 are tupel identifiers of the source instance I. Reducing each partial polynom t_1^2, $2t_1t_2$, t_2^2 to the set of its identifiers results the witness base $\{\{t_1\},\{t_2\},\{t_1,t_2\}\}$. A minimal witness base would be $\{\{t_1\}\}$ or $\{\{t_2\}\}$.

3 Invertible Query Evaluation

In our project ProSA (**Pro**venance management using **S**chema mappings with **A**nnotations) [4], which summarizes the DBIS research interests in the area of research data management, we develop techniques to invert evaluation queries against a research database (case VI. in Table 1) to be able to answer ***why***-provenance queries. We use the *witnesses* (results of the ***why***-query, see [13]) and the *provenance polynomials* (results of ***how***-query, see [21]) to determine the minimal sub-database of the original research database that has to be archived for reproducibility and replicability of research results.

3.1 Research Data Management

In *research data management* research data has to be managed, analyzed, stored, and archived for a long period of time. One of our research interests in this topic

is the question of how we can minimize the amount of data to be archived, especially in the case of constantly changing databases or database schemes and permanently performing new evaluations on these data (see [6])? Therefore we have to answer two concrete research questions:

1. Calculation of the minimal part of the original research database (we call it *minimal sub-database*) that has to be stored permanently to achieve replicable research.
2. Unification of the theories behind data provenance and schema (as well as data) evolution.

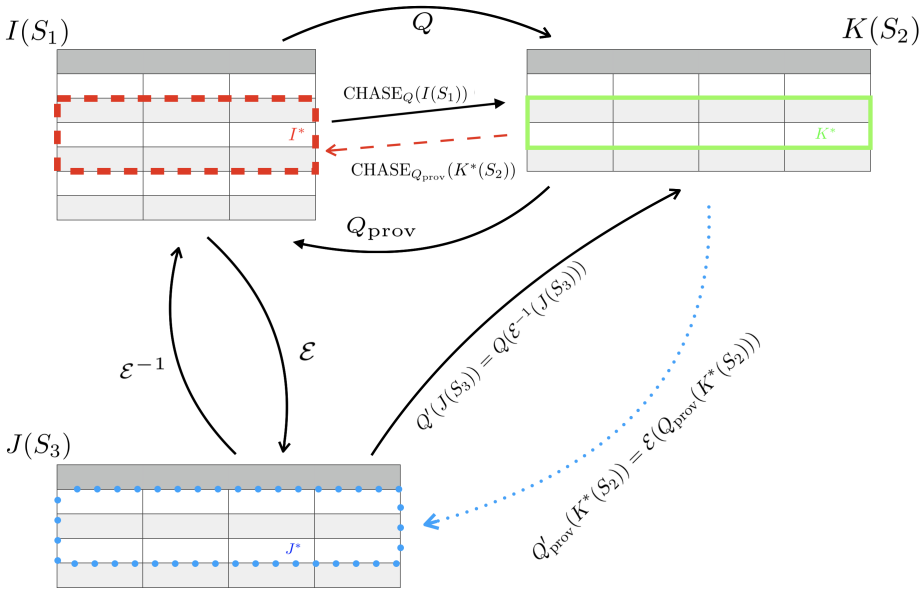


Fig. 2. Unification of Provenance and Evolution [4]

The idea behind these goals is sketched in Figure 2: The calculation of an inverse query Q_{prov}, which is used to determine the required minimal sub-database I^* (red dashed box), depends on the original query Q and the result instance (green box) that should be reconstructed:

$$Q_{\text{prov}}(K^*(S_2)) = Q^{-1}(K^*(S_2)).$$

Unfortunately, the specification of a concrete inverse is not always possible, but it is not even necessary in many cases. Sometimes, e.g. due to privacy requirements, we only want to reconstruct some kind of sub-database of the original research database by a quasi-inverted mapping [17]. If there is no direct inverse function, the only possibility to define such an inverse is adding provenance annotations

such as provenance polynomials. The aim of our research is to calculate the inverses or quasi-inverses of the evaluation query, if possible, and calculating additional provenance polynomials as annotations in cases, where direct inverses are not possible. The complete inverse mappings for the basic relational algebra operations are introduced in [6].

Nevertheless, the situation in research data management is even more complicated since the research database will evolve over time. There are frequent data evolution operations (updates) and occasional schema changes. Under the schema evolution $\mathcal{E} : S_1 \to S_3$, the query Q' can be directly calculated as a composition of the original query Q and the inverse evolution $\mathcal{E}^{-1}$. So the new provenance query Q'_{prov} of the new minimal sub-database J^* (blue dotted box) results as

$$Q'_{\text{prov}}(K^*(S_2)) = (Q_{\text{prov}} \circ \mathcal{E})(K^*(S_2)).$$

To combine data provenance (particularly ***why***-and ***how***-provenance to calculate a minimal sub-database of our research database), schema and data evolution in a common framework, we will use

- the CHASE for the evaluation query, the schema evolution and update operations on our research database,
- while using a second CHASE step (called *BACKCHASE*) applied to the result of the first CHASE step to be able to formally describe the ***why***- and ***how***-provenance by inverse mappings and provenance polynomials.

3.2 CHASE&BACKCHASE

For a given database (*CHASE object* $\bigcirc$) and a set of dependencies (*CHASE parameter* $\star$) the CHASE represents an evaluation query generating the query result. This evaluation can be particularly difficult in the case of operations aggregating data. In many cases, there is no suitable inverse. However, if the existence of an exact inverse is not important, in many cases a so-called *quasi-inverse* can be specified.

Usually three different types of (quasi-)inverse schema mappings created by the CHASE are distinguished: *exact*, *classic* and *relaxed CHASE inverses* [17, 18]. Two more inverse types, the *tuple preserving-relaxed* (*tp-relaxed*) and *result equivalent CHASE inverse* are introduced in [6], which also includes an overview of the inverse types of all relational algebra operations. As one example, the inverse type of the query $\pi_{AD}(\sigma_{C=c}(r_1) \bowtie \pi_{AB}(r_2))$ can be identified as relaxed. With additional provenance polynomials the projection itself can be identified as tp-relaxed. Generally, most basic operations can be optimized to become invertible by adding additional annotations like provenance polynomials [21, 3] or provenance games [31]. The inverse type of a composition of operations always corresponds to the weakest type of all partial mappings.

In this context, the CHASE&BACKCHASE procedure [14] can be used to calculate a suitable inverse function. A query Q formulated as an s-t tgd can now be processed using the CHASE and then inverted using the BACKCHASE to

calculate a minimal part of the original database. This calculated *minimal sub-database* should be able to reconstruct the query results under various constraints [5, 4]. So the questions are: How to automatically compute this minimal sub-database of the primary data? Given an evaluation query and the result of the evaluation query, is it possible to derive the minimal sub-database simply by an inverse mapping without any additional annotations of the data and without any need to freeze the original database? These questions address some exciting problems of research data management, such as traceability, reproducibility and replicability of research results [11, 12, 28].

3.3 Provenance using CHASE

Over the years, the CHASE algorithm has been used in many different areas of application. In our ProSA project, the CHASE is used for query evaluation (i.e., data transformation), schema and data evolution, and data exchange. The BACKCHASE is used to invert the query evaluation Q, i.e. to calculate the extensional answer of the ***why***-Provenance as the witness base of the query Q.

Given a schema mapping $\mathcal{M}$ defined by an s-t tgd and a source instance I, Fagin [18] computes a target instance via $\text{CHASE}_{\mathcal{M}}(I)$. He also discussed a target and a source schema evolution and specified some types of CHASE inverse functions (exact, classic, relaxed), which are essential for schema evolution. In ProSA, we use Fagin's schema mappings as queries Q, schema evolution, and data exchange steps.

The result of the evaluation query Q described by extended s-t tgd's and egd's can be calculated using the CHASE algorithm. The calculation of the minimal sub-database I' is computed by inverting Q. This inverse Q_{prov} does not necessarily have to correspond to an inverse in the classical sense $Q \circ Q_{\text{prov}} = \text{Id}$, since a CHASE inverse cannot always be specified [4]. The provenance answer I' can then be calculated using the BACKCHASE.

4 Many application areas – one tool: ChaTEAU

Since different research projects in our group use the CHASE as a basic technique (see Section 6), we decided to either use or develop one general CHASE tool that is applicable to all of these applications. A nice overview of already existing CHASE tools is given in [8]. To our knowledge, and after evaluating the CHASE tools being publicly available, none of the existing tools can be applied to all of the scenarios mentioned above (Table 1), or at least to some scenarios of each of both groups (case I. and II.: queries as CHASE objects; case III. to VI:: databases as CHASE objects).

Here, the basic idea of our universal tool ChaTEAU (**Cha**se for **T**ransforming, **E**volving, and **A**dapting databases and queries, **U**niversal approach) as well as some preliminary implementation work will be presented. Finally, we can construct a tool for each of the three application scenarios based on ChaTEAU (see Figure 4(b) in Section 5). In the case of invertible query evaluation, for example, extended data preparation and a BACKCHASE phase are necessary.

4.1 Theoretical Foundation of ChaTEAU

The main idea of making the various use cases of the CHASE applicable in a single, universal tool is based on the fact that the CHASE objects ◯ and parameters ⋆ are essentially interchangeable without significantly changing the way the CHASE is executed. With this in mind, we can abstract Algorithm 1 by replacing the input (a set of dependencies Σ and a database instance I) by a general parameter ⋆ and a general object ◯.

CHASE parameter ⋆ This parameter is always represented as a intra-database or inter-database dependency: In semantic optimization, it corresponds to a set of egd's and tgd's as intra-database constraints, in the case of AQuV, to a set of views (interpreted as dependencies between base and view relations), or to a set of s-t tgd's, tgd's and egd's as in data exchange and data integration.

CHASE object ◯ The CHASE object ◯ is either a query Q or a database instance I. While in queries (distinguished or non-distinguished) variables v_i [32] can be replaced by other variables v_j or constants c_i, in database instances null values η_i are replaced by other null values η_j or constants c_i. The variable substitution depends on certain conditions which are shown in Figure 3. These conditions guarantee a homomorphic mapping of the CHASE object.

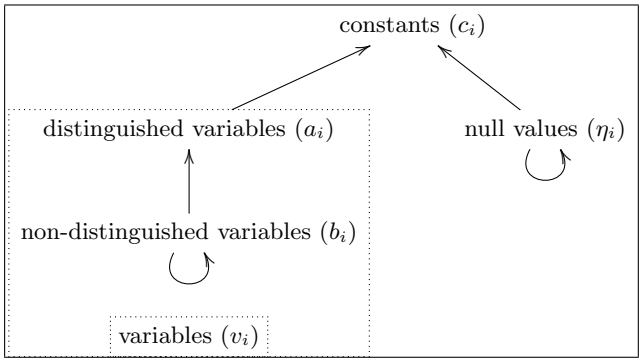


Fig. 3. Possible homomorphisms for the object ◯

CHASE rules The replacement rules incorporating the CHASE parameter ⋆ into the CHASE object ◯ are independent of the selection of ⋆ or ◯ itself. Currently these are egd-, tgd- and s-t tgd-rules [32, 18]. Currently, we are working on an extension of these rules to handle aggregates in evaluation queries, too: these aggregates have to be integrated similar to Skolem functions in the target part of an s-t tgd. There are also existing rules for views like in case II [14].

4.2 ChaTEAU

The core of ChaTEAU is based on Algorithm 1. A first implementation was given in [30]. The input of the algorithm can be any possible CHASE parameter $\star$ and object $\bigcirc$ as described above. The output will then be a transformed database schema, a query (satisfying certain constraints) or a modified database instance.

ChaTEAU is based on five classes of objects: terms, homomorphisms, integrity constraints, atoms and instances. Terms form an elementary data structure in ChaTEAU which are represented by the class TERM. As in theory, they can be a variable, a null value or a constant. Constant values are represented by simple data types. However, since they can be strings, numbers, or other values, and the data type is not known at runtime, the approach used here is dynamic typing. The currently supported data types are `string`, `double` and `integer`. For each data type, there is a field variable of the class that stores the corresponding value. A term can only contain one value of one type, which is why only one of these field variables is ever used. Variables and null values are also part of the TERM class. The distinction between these three types is made by an enumeration, which determines what kind of term it is when a term is created. So TERM is elementary for the other classes used in ChaTEAU.

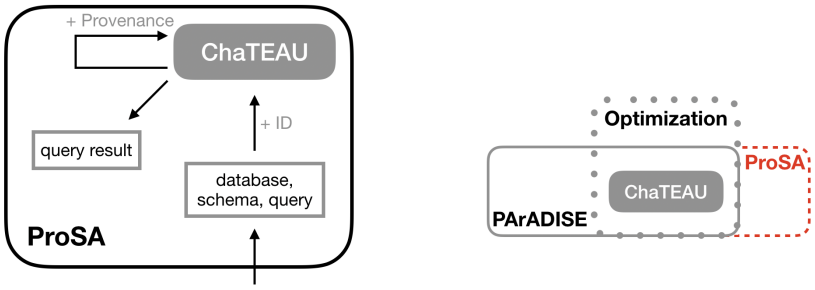
The tool as well as the theory differentiates between relational and equality atoms. Both are the basic building block for instances, queries and integrity constraints. Relational atoms themselves consist of a term and an identifier. Equality atoms contain two terms, expressing an equality relation between them. Both are combined in the class ATOM.

Instances are implemented in INSTANCE and constraints like tgd's as well as egd's in INTEGRITY CONSTRAINTS. An integrity constraint always consists of a head and a body, which again consist of atoms. Apart from the head of an egd, which is formed from an equality atom, relational atoms are used for the heads and bodies. Also instances are based on relational atoms.

If there is a homomorphism between a source and a target term, a term mapping is generated which maps one term to another term like $v_i \to v_j$, $\eta_1 \to$ Max and $\eta_2 \to \eta_1$. Homomorphisms are also fundamental for the use in CHASE rules and defining active triggers (see Algorithm 1).

5 ProSA using ChaTEAU

The reconstruction of a minimal sub-database I^* like in Figure 2 is the main task of our project ProSA [4]. To achieve this, a given database instance I as well as the associated schema S_1 are extended by a unique identifier. This ID is a coding usually consisting of the table name and a consecutive number. This modified database, the schema and the evaluation query Q are now input for the CHASE algorithm implemented in ChaTEAU and extended by provenance aspects which are realized in a separate BACKCHASE phase (see Figure 4(a)). The result of the evaluation query and the result of the provenance query are finally the output of ProSA.



(a) Overview of ProSA (b) Tools for CHASE application

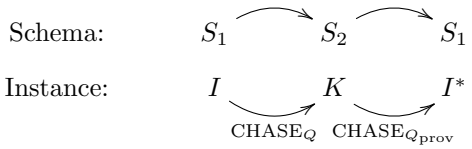
Fig. 4. Applications of the CHASE at DBIS

ChaTEAU is not only the core of ProSA (red dashed box, Figure 4(b)), but also the basis of two other tools for Answering Queries using Operators (solid box) as well as for semantic query optimization (dotted box). Both applications will be briefly motivated below in Section 6.

CHASE*&*BACKCHASE The combination of CHASE and provenance in one application is explained in detail in [6]. There we describe the idea that the *recovered instance*

$$I^* = \text{CHASE}_{Q_{\text{prov}}}(K) = \text{CHASE}_{Q_{\text{prov}}}(\text{CHASE}_Q(I))$$

is thus the result of a query Q_{prov} on the *result instance* K. I^* contains whole tuples from I or tuples restricted to certain attributes of the source schema (and filled with (marked) null values).



The CHASE&BACKCHASE method for determining a CHASE inverse schema mapping / provenance mapping $Q_{\text{prov}} = (S_2, S_1, \Sigma_2)$ to $Q = (S_1, S_2, \Sigma_1)$ can therefore be described in two phases:

- CHASE phase: Calculate the CHASE of I as the CHASE object $\bigcirc$ with Q as a CHASE parameter $\star$ represented as a sequence of s-t tgd and egd rules which generate new tuples (s-t tgd) and replace (marked) null values with constants or null values with smaller index (egd).
- BACKCHASE phase: Calculate the CHASE of K (or an interesting subset of the result, called K^*, for which the provenance should be checked) as the CHASE object $\bigcirc$ with Q_{prov} as a CHASE parameter $\star$, again represented as a sequence of s-t tgd and egd rules.

An evaluation query Q is processed by the CHASE. So a given database instance $I(S_1)$ delivers a result instance $K(S_2)$ which is shown in Figure 5 (green). Such a mapping is easy to find for simple queries with SPJU (select, projection, join, union), but difficult for queries with aggregation. For this purpose, the CHASE must be extended by functions which code aggregation functions like MAX, MIN, COUNT, SUM and AVG.

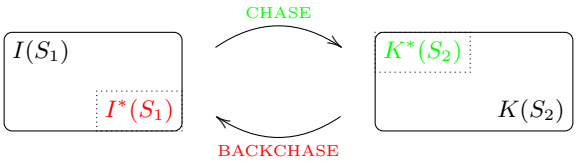


Fig. 5. CHASE&BACKCHASE in ProSA

Additionally, the provenance query Q_{prov} is processed by the BACKCHASE resulting in an (exact, classic, relaxed, result-equivalent, ...) CHASE inverse adding some annotations like provenance polynomials to guarantee the reconstruction of the recovered instance I^* as a minimal sub-database of the research database I that has been the basis for the evaluation query Q.

6 Other Applications of the CHASE

Within the Database Research Group of the University of Rostock, we work in different areas such as *Semantic Query Optimization*, *Answering Queries using Operators* (AQuO) and *Invertible Query Evaluation*. However, the different areas of application have one thing in common. All three can be processed by means of the CHASE implemented in ChaTEAU (see Fig. 4(b)).

Semantic Query Optimization One of the first and classical applications of the CHASE is injecting data dependencies into queries to be able to optimize queries under given dependencies. In one of our research projects, we looked for different kinds of tgd's (such as inclusion dependencies) as inter-object relationships in multimedia databases. Instead of extracting multimedia features for content-based retrieval, we use the inter-object dependencies to optimize queries consisting of longer join paths. The dependencies are CHASEd into the query, and a BACKCHASE phase will generate an optimized query plan minimizing the number of joins used.

Anwering Queries using Operators (AQuO) In our project PArADISE (**P**rivacy-**A**ware **a**ssistive **D**istributed **I**nformation **S**ystem **E**nvironment, see [23, 25]) we use query rewriting and query containment techniques to achieve an efficient and privacy-aware processing of queries. To achieve this, the whole

network structure, from data-producing sensors up to cloud computers, is utilized to create a database machine consisting of billions of devices, the Internet of Things. Based on previous research in the field of query rewriting, we developed a concept to split a query into fragment and remainder queries. Fragment queries can operate on resource limited devices to filter and preaggregate data. Remainder queries take these preprocessed (filtered and aggregated) data and (only) execute the last, complex part of the original queries on more powerful devices like cloud servers. As a result, less data is processed and forwarded to a cloud server and the privacy principle of data minimization is accomplished [24–26].

In one of our rewriting techniques, we used an extension of the AQuV technique to answer queries using only restricted capabilities of query processors. The AQuV problem has to be generalized to the AQuO problem (Anwering Queries using Operators), describing restricted query operators available on the devices in the Internet of Things. The idea for ChaTEAU is to CHASE these operators into the query to be able to BACKCHASE the enhanced query to fragment and remainder queries automatically.

7 Conclusion and Future Work

The aim of our project ProSA is the development of techniques to invert evaluation queries against a research database to be able to answer ***how***- and ***why***-provenance. We use the CHASE tool ChaTEAU to represent the queries, data and schema evolution, and calculate he provenance answers by the BACKCHASE process. Some open questions in the areas ChaTEAU and ProSA are:

ChaTEAU We have to complete our implementation to get a tool that can handle a generalized CHASE object $\bigcirc$ as well as a generalized CHASE parameter $\star$. We are working on extensions of s-t tgd, tgd and egd rules to handle more complex dependencies and query operators such as aggregations. On the other hand, we need to control the CHASE process by evaluating complexity and runtime of the CHASE procedure, and we need to ensure the termination of the CHASE procedure by adapting criteria such as the weak acyclicity introduced for tgd's as a CHASE parameter [16].

ProSA For a concrete implementation of the BACKCHASE phase, we first need a definition of special BACKCHASE rules. These rules correspond to the CHASE rules with additional annotations like provenance polynomials. Again, the processing of aggregates is a difficulty here, but similar to the processing using the CHASE.

Additionally, we have to decide about the annotations to be stored if an exact or classic inverse does not exist for our evaluation query. While the use of provenance polynomials as an annotation is quite elegant, it is not efficient to process all the polynomials on a large database. This is particularly the case if the research database is very large. Besides this, due to privacy reasons, it is

not always appropriate to calculate elements of the original research database exactly. In this case, an intensional answer of the provenance query (only describing the content of the original database in a descriptive, anonymized manner) seems to be more appropriate than the extensional answer we calculated up to now.

Lastly, the ProSA technique is prepared to integrate schema evolution steps and database updates. We have to describe these evolutions by s-t tgd's to incorporate evolving databases into the ProSA tool.

Acknowledgements

We thank our students Fabian Renn and Frank Röger for their comparison of different CHASE tools like LLUNATIC and PDQ as well as Martin Jurklies for the basic implementation of our CHASE tool ChaTEAU.

References

1. Aho, A.V., Beeri, C., Ullman, J.D.: The Theory of Joins in Relational Databases. ACM Trans. Database Syst. **4**(3), 297–314 (1979)
2. Amarilli, A., Bourhis, P., Senellart, P.: Provenance circuits for trees and treelike instances (extended version). CoRR **abs/1511.08723** (2015)
3. Amsterdamer, Y., Deutch, D., Tannen, V.: Provenance for aggregate queries. In: PODS. pp. 153–164. ACM (2011)
4. Auge, T., Heuer, A.: Combining Provenance Management and Schema Evolution. In: IPAW. Lecture Notes in Computer Science, vol. 11017, pp. 222–225. Springer (2018)
5. Auge, T., Heuer, A.: Inverses in Research Data Management: Combining Provenance Management, Schema and Data Evolution (Inverse im Forschungsdatenmanagement). In: Grundlagen von Datenbanken. CEUR Workshop Proceedings, vol. 2126, pp. 108–113. CEUR-WS.org (2018)
6. Auge, T., Heuer, A.: The Theory behind Minimizing Research Data: Result equivalent CHASE-inverse Mappings. In: LWDA. CEUR Workshop Proceedings, vol. 2191, pp. 1–12. CEUR-WS.org (2018)
7. Benczúr, A., Kiss, A., Márkus, T.: On a general class of data dependencies in the relational model and its implication problems. Computers & Mathematics with Applications **21**(1), 1–11 (1991)
8. Benedikt, M., Konstantinidis, G., Mecca, G., Motik, B., Papotti, P., Santoro, D., Tsamoura, E.: Benchmarking the Chase. In: PODS. pp. 37–52. ACM (2017)
9. Benedikt, M., Leblay, J., Tsamoura, E.: PDQ: Proof-driven Query Answering over Web-based Data. PVLDB **7**(13), 1553–1556 (2014)
10. Bonifati, A., Ileana, I., Linardi, M.: ChaseFUN: a Data Exchange Engine for Functional Dependencies at Scale. In: EDBT. pp. 534–537. OpenProceedings.org (2017)
11. Bruder, I., Heuer, A., Schick, S., Spors, S.: Konzepte für das Forschungsdatenmanagement an der Universität Rostock(Concepts for the Management of Research Data at the University of Rostock). In: LWDA. CEUR Workshop Proceedings, vol. 1917, p. 165. CEUR-WS.org (2017)

12. Bruder, I., Klettke, M., Möller, M.L., Meyer, F., Heuer, A.., Jürgensmann, S., Feistel, S.: Daten wie Sand am Meer - Datenerhebung, -strukturierung, -management und Data Provenance für die Ostseeforschung. Datenbank-Spektrum **17**(2), 183–196 (2017), https://doi.org/10.1007/s13222-017-0259-4
13. Buneman, P., Khanna, S., Tan, W.C.: Why and Where: A Characterization of Data Provenance. In: ICDT. Lecture Notes in Computer Science, vol. 1973, pp. 316–330. Springer (2001)
14. Deutsch, A., Hull, R.: Provenance-Directed Chase&Backchase. In: In Search of Elegance in the Theory and Practice of Computation. Lecture Notes in Computer Science, vol. 8000, pp. 227–236. Springer (2013)
15. Deutsch, A., Popa, L., Tannen, V.: Query reformulation with constraints. SIGMOD Record **35**(1), 65–73 (2006)
16. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: semantics and query answering. Theor. Comput. Sci. **336**(1), 89–124 (2005)
17. Fagin, R., Kolaitis, P.G., Popa, L., Tan, W.C.: Quasi-inverses of schema mappings. ACM Trans. Database Syst. **33**(2), 11:1–11:52 (2008)
18. Fagin, R., Kolaitis, P.G., Popa, L., Tan, W.C.: Schema Mapping Evolution Through Composition and Inversion. In: Schema Matching and Mapping, pp. 191–222. Data-Centric Systems and Applications, Springer (2011)
19. Geerts, F., Mecca, G., Papotti, P., Santoro, D.: That's All Folks! LLUNATIC Goes Open Source. PVLDB **7**(13), 1565–1568 (2014)
20. Greco, S., Molinaro, C., Spezzano, F.: Incomplete Data and Data Dependencies in Relational Databases. Synthesis Lectures on Data Management, Morgan & Claypool Publishers (2012)
21. Green, T.J., Karvounarakis, G., Tannen, V.: Provenance semirings. In: PODS. pp. 31–40. ACM (2007)
22. Green, T.J., Tannen, V.: The Semiring Framework for Database Provenance. In: PODS. pp. 93–99. ACM (2017)
23. Grunert, H., Heuer, A.: Datenschutz im PArADISE. Datenbank-Spektrum **16**(2), 107–117 (2016)
24. Grunert, H., Heuer, A.: Privacy Protection through Query Rewriting in Smart Environments. In: EDBT. pp. 708–709. OpenProceedings.org (2016)
25. Grunert, H., Heuer, A.: Rewriting Complex Queries from Cloud to Fog under Capability Constraints to Protect the Users' Privacy. OJIOT **3**(1), 31–45 (2017)
26. Grunert, H., Heuer, A.: Query rewriting by contract under privacy constraints. OJIOT **4**(1), 54–69 (2018)
27. Halevy, A.Y.: Answering queries using views: A survey. VLDB J. **10**(4), 270–294 (2001)
28. Herschel, M., Diestelkämper, R., Ben Lahmar, H.: A survey on provenance: What for? What form? What from? VLDB J. **26**(6), 881–906 (2017)
29. Ileana, I., Cautis, B., Deutsch, A., Katsis, Y.: Complete yet practical search for minimal query reformulations under constraints. In: SIGMOD Conference. pp. 1015–1026. ACM (2014)
30. Jurklies, M.: CHASE und BACKCHASE: Entwicklung eines Universal-Werkzeugs für eine Basistechnik der Datenbankforschung. Master's thesis, Universität Rostock (2018)
31. Köhler, S., Ludäscher, B., Zinn, D.: First-Order Provenance Games. CoRR **abs/1309.2655** (2013), http://arxiv.org/abs/1309.2655
32. Maier, D.: The Theory of Relational Databases. Computer Science Press (1983)
33. Maier, D., Mendelzon, A.O., Sagiv, Y.: Testing Implications of Data Dependencies. ACM Trans. Database Syst. **4**(4), 455–469 (1979)