Enhancing Data Management and Value Creation through a Knowledge-centric Data Stack

Enrico Franconi[1], Théo Abgrall[2]

KRDB Research Centre for Knowledge-based Artificial Intelligence, Free University of Bozen-Bolzano, Italy - https://krdb.eu

Abstract

Nowadays we observe an evolving landscape of data management and analytics, emphasising the significance of meticulous data management practices, semantic modelling, and bridging business-technical divides, to optimise data utilisation and enhance value from datasets in modern data environments. In this paper we introduce and explain the basic formalisation of the Semantic SQL Transducer, a well-founded but practical tool providing the materialised lossless conceptual view of an arbitrary relational source data, contributing to a knowledge-centric data stack.

1. Introduction

The landscape of data management and analytics is undergoing continuous evolution, aiming to optimise data utilisation, ensure governance, and enhance the value derived from the datasets. Several pivotal concepts shape an enhanced modern data environment, emphasising the significance of robust data preparation, semantic modelling, and bridging the gap between technical and business perspectives. By looking to the current trends in data management, key aspects include the increasing significance of metadata management for data governance, the necessity of comprehensive semantic enrichment in data contracts and data preparation, the importance of bridging the divide between business problem models and data domains through the integration of semantic mediation, the adoption of a semantic-based declarative transformation process, and the facilitation of seamless data integration and improved interoperability through shared semantic understanding.

Auditability and Data Governance. Auditability stands as a critical factor in aiding data analysts to comprehend and model schemas effectively. The emphasis on properly managing metadata supports effective data governance by encompassing the meticulous organisation, quality control, and management of key properties like completeness, consistency, fairness, privacy, provenance, and other data qualities. The focus on enhancing data analyst comprehension and schema modelling is pivotal, highlighting the paramount importance of serious metadata management [1].

Data Preparation. Data preparation serves as the foundational pillar upon which successful analytics stands. Tasks encompassed within data prep, such as cleaning, parsing, integrity checks, and data set unification, are crucial. Much like the significance of a solid foundation

SEBD 2025: 33rd Symposium on Advanced Database Systems, June 16-19, 2025, Ischia, Italy

franconi@inf.unibz.it (E. Franconi); theo.abgrall@student.unibz.it (T. Abgrall)

for monumental structures, a meticulous data preparation phase is indispensable for accurate visualisation, reporting, and forecasting, and it is the step where raw data transforms into actionable insights. Unluckily, the tools that vendors propose for data preparations do not include a serious role of semantics in the pipeline, but only some shallow semantic enrichment such as semantic data types [2].

Shifting Perspectives on Data Utilisation. Historically, the predominant focus in the data domain has been on downstream tasks like visualisation and reporting. However, contemporary trends reveal a substantial shift, where data preparation and meticulous data management consume a significant portion of analysts' time. Modern tools provide a bridge to streamline these preparation-intensive tasks, minimising manual labor and enhancing efficiency [3].

Semantic Modelling and Data-Centric Approaches. The traditional approach to data warehousing often resulted in tangled data sets, complicating queries and hindering data trustworthiness. Embracing semantic modelling involves steps like knowledge graphs creation, engineering ownership, data contracts, and subsequent event implementation. Such methodologies ensure that data scientists and engineers spend their time defining and utilising high-quality data, contributing to a more robust and efficient data warehouse. In this context, semantic enhancement plays a critical role in elevating the value and usability of data within industrial contexts. Semantic enhancement isn't merely about structuring data but imbuing it with rich context, relationships, and meaning [4].

Data Catalogs. Classical data catalogs and data contracts often fall short in capturing the meaning of the business domain of the companies they are meant to represent. To effectively serve users, catalogs must act as bridges between the user's problem model and the underlying data domain by incorporating semantic understanding. This involves creating ontologies, structured representations of knowledge that enhance the comprehension of data relationships. When business users attempt to interpret data, they often lack the semantic context necessary for a full understanding. They communicate in their familiar business language, while data appears as a new language tied to the source structures. Here's where the semantic layer comes into play. This critical component adds a layer of business context to data stored in catalogs, essentially translating it into a language that the business can easily comprehend. The semantic layer also ensures consistency across diverse and heterogeneous data by implementing a common business logic. This conformity empowers businesses to derive meaning from the data coming out of the data stack pipeline [5].

Challenges and Solutions in Modern ETL. Bill Inmon highlights the evolution of ETL (Extract, Transform, Load) and ELT (Extract, Load, Transform) processes, noting a tendency among vendors and consultants to shift from ETL to ELT. As a consequence, ELT often places the burden of transformation on others, delaying the essential data transformation step. "The upshot is – if you wanted to just move some data around, then ELT is your thing. But if you want believable data, then you have to do ETL. So the choice is yours. Do you want data quickly and easily, that may be essentially unreliable? Or do you want data that forms a firm foundation for – AI, analytics, data mesh, ML, et al?". While modern ETL practices have expedited data processing, its procedural nature and lack of a systematic analysis of the data transformation process also brought challenges like incomplete or irrelevant data, tight coupling between

services and analytics, and a surge in technical debt. There is the need of a solution involving a return to upfront semantic data definition and design driving a declarative transformation process.

The Importance of Business Literacy. Bridging the gap between business and technical perspectives emerges as a crucial necessity. Envisioning and developing semantic warehouses that facilitates enriched communication between business leaders and data practitioners through context-rich data representations serve as the cornerstone for unified and comprehensible data landscapes. As Juan Sequeda says, data modelling without working with subject matter experts/end users is simply a recipe for disaster.

Unifying Ontologies for Data Integration. The integration of diverse metadata within enterprises requires a systematic approach. Semantic modelling supports the harmonisation of disparate data elements within an organisation. A singular, adaptable, and shareable ontology serves as a linchpin, unifying complex data landscapes, fostering cohesive data management strategies, enabling seamless data integration and enhancing interoperability [6].

A Pragmatic Approach. Dave McComb calls for a pragmatic approach to semantic enhancement, balancing the theoretical underpinnings of ontology development with practical implementations tailored to industrial settings. He advocates for methodologies that facilitate the creation of ontologies that are adaptable, shareable, and capable of evolving alongside dynamic data landscapes.

The Boring Data Stack. Joe Reis introduces the concept of the "Boring Data Stack", signalling a shift in focus from managing underlying technologies to addressing critical yet often neglected aspects like data governance and semantic modelling [7]. He emphasises the importance of these "boring" practices, attributing their significance to the era of AI and ML advancements. He points out the need for organisations to address data quality issues, especially in the context of AI and ML applications, where messy datasets pose significant challenges. Reis envisions a future where data-centric approaches evolve into knowledge-centric ones, emphasising the necessity of robust data governance, management, and semantic modelling practices to achieve this transition effectively. He stresses the criticality of conceptual and logical data modelling in aligning data with the realities of business operations, cautioning against the prevalent tendency to solely focus on physical data models disconnected from business needs.

Given this context, we believe a contribution to support proper semantic modelling within a data preparation pipeline is badly needed. In this paper we introduce and explain the basic formalisation of the *Semantic SQL Transducer*, a well-founded but practical tool providing the materialised *lossless* and possibly conceptual view of an arbitrary relational data, contributing to a knowledge-centric data stack. The Semantic SQL Transducer can be seen as a seamless semantic wrapper around arbitrary relational data at any stage of the data stack, independently on its architecture. The advantage of this technology is that it can be seen as a replacement of the data it models, providing a restructuring of the data according to its restructured (and possibly conceptual) model as a standard SQL database, which can be therefore queried, updated, transformed. It can be used also to replace procedural data transformation tasks with semantic-based declarative executable specification of the transformation task, guaranteeing the losslessness of the transformation itself. By restructuring the relational data directly in its

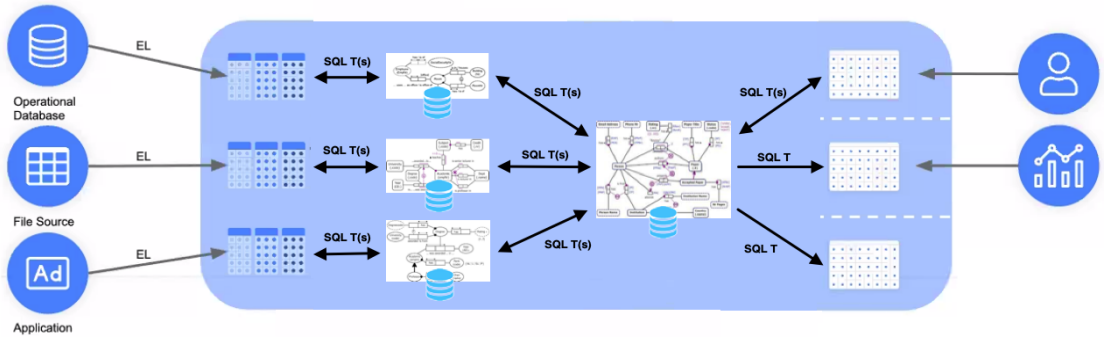

Figure 1: A Semantic Data Stack

conceptual model, the transducer provides a conceptual access to the data, providing to business users and data analysts the right understanding of the available data.

2. The role of a Semantic SQL Transducer

Our contribution to a knowledge-centric data stack consists in a SQL-based tool (supported by a design methodology) providing the materialised *lossless* conceptual view of an arbitrary relational source data. Such materialised conceptual view can be queried and updated using the knowledge vocabulary, with virtually no overhead with respect to the original source data. Updates to the materialised conceptual view are replicated instantaneously to the source data (e.g., to push a semantic-conscious data cleaning update to the source), and updates to the source data are replicated instantaneously to the materialised conceptual view providing always a fresh view of the source. The materialised view is the conceptual lossless mirror of the source data, and it acts as a mediator by providing the conceptual API for a complete access and change to the source data.

Our *Semantic SQL Transducer*is based completely on standard SQL technology, it can be deployed on any SQL platform, and it does not require any additional tool or code to work. The transducer can exactly translate legacy SQL queries and updates over the source schema to SQL queries and updates over the conceptual schema, and it can exactly translate analytical SQL queries and updates over the conceptual schema to SQL queries and updates over the source schema. By using semantic SQL transducers within a data prep process as semantic wrappers around the data sources (e.g., see Figure 1), the "transform" part of the ETL process can operate over semantically well defined entities and relationships. The transducer supports transactions, to systematically guarantee semantic integrity and consistency of both the source and its conceptual model.

In order to represent the exact semantics of the source data, the Semantic SQL Transducer supports several popular conceptual data models: ERD, ORM, UML Class Diagrams, Property Graphs Schemas, Knowledge Graphs. It is based on several years of theoretical research by us on the formalisation of the connection between conceptual data models and relational databases, and on the formalisation of core SQL [8, 9, 10, 11, 12]. A rigorous methodology to properly design

a semantic SQL transducer given the data sources has been studied and experimented [13, 14]; we are developing several tools to support it, which are not yet publicly available.

We believe that our pragmatic approach to semantic enrichment provides a useful knowledge-based core element which can be embedded within many different data architectures, improving on the issues emphasised above in the direction of making the boring data stack more exciting: auditing and data governance now are based on a clear semantic view of the data, supporting a more transparent environment to exploit business literacy; during data preparation, semantic integrity checks can be defined over the conceptual structures, and data cleaning can be enforced at the level of the conceptual view; data integration and entity recognition are now semantic-based, and the presence of a unified conceptual model exactly capturing the diverse data sources supports the harmonisation of the disparate data elements within the enterprise; the gap between data management and data analysis is reduced, being mediated by the transducers, reducing therefore the risk of a technical debt; data scientists operate over high-quality data, contributing to a more robust and efficient data warehouse.

Note that there are many semantic-based approaches that introduce an intermediate layer between the data layer and the business layer, which should be compared the proposal, such as enterprise data fabrics, data meshes, data lakes, etc. The proposed Semantic SQL Transducer is orthogonal to any of such architectural choices. The suggested semantic data stack in Figure 1 above had the only purpose how the Semantic SQL Transducer could support semantic enrichment in a modern data stack. The unicity of our proposal lies in the fact that it can losslessly "present" the data in a restructured way, possibly according to its conceptual schema, always using SQL as the foundational formalism.

3. Inside the Semantic SQL Transducer

The abstract internal architecture of the Semantic SQL Transducer is shown in Figure 2. It is a generic architecture, implementing the lossless bidirectional interoperation between two databases, called `S` (source) and `T` (target). The SQL code guarantees that the two databases are always automatically synchronised after any update (wrapped within a transaction) to any of the two databases. The two databases maintain their original constraints and indexes, maximising therefore the efficiency of querying. The updates to a database are recasted directly as actual updates to the other database, maximising therefore the efficiency of updates. Some attention has to be paid to avoid infinite looping of the triggers.

The real complexity comes in defining both the lossless mappings from `S` to `T` and from `T` to `S` (appearing in the SQL code as `create table X as select ...`) and the constraints of the two databases `S` and `T`. Those mappings and constraints are provided by our theory of *lossless transformations* [9, 14, 11, 13, 12], based on the original works on *information capacity* [15, 16, 17, 18, 19, 20].

When the transducer is used to provide the semantic enrichment as a conceptual view of a data source within an ETL pipeline, the database `S` is indeed the source database, and the database `T` is the dynamic restructuring of `S` according to its conceptual schema. The involved mappings and constraints are a generalisation of classical mappings studied in the reverse engineering literature [21, 22, 23, 24, 25, 26], started by the seminal papers by Hainaut [27, 28, 29].

Source S:

```
create table S1 ...      (with source constraints)
...
create table Sm ...      (with source constraints)
```

Target T:

```
create table T1 ...      (with target constraints)
...
create table Tn ...      (with target constraints)
```

Insert S ⇒ T:

```
create table S1_INSERT ...      (same schema as S1)
...
create table Sm_INSERT ...      (same schema as Sm)

create trigger S1_INSERT after insert on S1 ...   (updates S1_INSERT)
...
create trigger Sm_INSERT after insert on Sm ...   (updates Sm_INSERT)

create table T1_INSERT as select ... from S1_INSERT,...,Sm_INSERT ...      (same schema as T1)
...
create table Tn_INSERT as select ... from S1_INSERT,...,Sm_INSERT ...      (same schema as Tn)

create trigger T1_INSERT after insert on T1_INSERT ...   (updates T1)
...
create trigger Tn_INSERT after insert on Tn_INSERT ...   (updates Tn)
```

Insert T ⇒ S:

```
create table T1_INSERT ...      (same schema as T1)
...
create table Tn_INSERT ...      (same schema as Tn)

create trigger T1_INSERT after insert on T1 ...   (updates T1_INSERT)
...
create trigger Tn_INSERT after insert on Tn ...   (updates Tn_INSERT)

create table S1_INSERT as select ... from T1_INSERT,...,Tn_INSERT ...      (same schema as S1)
...
create table Sm_INSERT as select ... from T1_INSERT,...,Tn_INSERT ...      (same schema as Sm)

create trigger S1_INSERT after insert on S1_INSERT ...   (updates S1)
...
create trigger Sm_INSERT after insert on Sm_INSERT ...   (updates Sm)
```

Similarly: **Delete S ⇒ T**, **Delete T ⇒ S**.

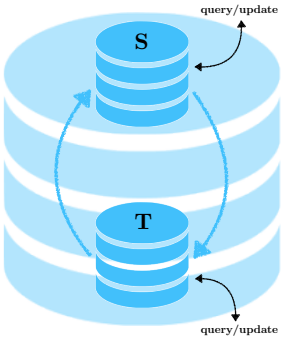

Figure 2: The Semantic SQL Transducer abstract architecture

We can show that any source database has associated a unique *canonical abstract relational model* [30], which is the lossless materialisation of the database in its conceptual schema in 6^{th} normal form. The canonical abstract relational model has a direct correspondence with the most popular conceptual modelling languages such as ERD, ORM, UML Class Diagrams, Property Graphs Schemas, RDF-based models. We provide a rigorous methodology to properly design the conceptual schema in the form of a canonical abstract relational model, given the data sources [11, 12]; we are developing several tools to support it. In the next Section we will define the notion of losslessness in database restructuring, and we will explain all the basic transformation steps of the design methodology through a completely worked out example.

In conclusion, the Semantic SQL Transducer provides a seamless access to databases through their conceptual schemas, and it contributes to a knowledge-centric data stack adding a declarative semantic layer to databases.

4. Designing the Semantic SQL Transducer

In order to formalise the SQL transducer, we introduce first the notion of a *first-order database schema*. A first-order database schema *DB* is a pair $\langle \mathbb{A}_{DB}, \mathbb{C}_{DB} \rangle$ where $\mathbb{A}_{DB}$ is a set of database predicates with their attributes $R(a_1, a_2, \cdots, a_n)$ – for simplicity we do not consider here the domains of the attributes – and $\mathbb{C}_{DB}$ is a set of first order constraints (aka dependencies) over the predicates. In order to capture exactly SQL, we restrict constraints to be in the domain-independent fragment of first-order logic: all interesting kinds of constraints can be represented, ranging from functional and multivalued dependencies (including keys), to inclusion dependencies (including foreign keys), to constraints on domain values. We will use a standard notation for classical database dependencies, most notably, $a_1, \cdots, a_n \rightarrow b_1, \cdots, b_m$ for functional dependencies, $a_1, \cdots, a_n \twoheadrightarrow b_1, \cdots, b_m$ for multivalued dependencies, $a_1, \cdots, a_n \subseteq b_1, \cdots, b_n$ for inclusion dependencies, and $a \subseteq \{\text{“}k_1\text{”}, \cdots, \text{“}k_n\text{”}\}$ for domain constraints. We will write general dependencies using relational algebra.

Lossless transformations.

Suppose that $I(S)$ and $I(T)$ are the sets of legal database instances (or models) for schemas S and T respectively: following [16, 17] a *schema transformation* from S to T is a total mapping function $f_{S \rightarrow T} : I(S) \rightarrow I(T)$. In order to define a *lossless* transformation, we need to introduce first the notion of schema dominance [16, 17].

Given two schemas S and T, T *dominates* S if there is a total and injective mapping function $f_{S \rightarrow T} : I(S) \rightarrow I(T)$ which maps legal database instances in S to legal database instances in T. Equivalently, we can say that T *dominates* S if there are two mapping functions $f_{S \rightarrow T} : I(S) \rightarrow I(T)$ and $f_{T \rightarrow S} : I(T) \rightarrow I(S)$ exist, such that their composition, $f_{T \rightarrow S} \circ f_{S \rightarrow T}$ (the result of applying $f_{T \rightarrow S}$ after applying $f_{S \rightarrow T}$), is the identity on $I(S)$.

Two schemas S and T are equivalent, written $S \equiv T$, if and only if T dominates S and S dominates T. When two schemas S and T are equivalent, the mappings $f_{S \rightarrow T}$ and $f_{T \rightarrow S}$ are bijective, and we say that both schemas have the same *information capacity* and that the transformation is *lossless*.

In our setting, we consider mappings as first-order *views* establishing the relation between two database schemas. More precisely, given two schemas S and T, a first-order mapping from S to T, written $M_{S \rightarrow T}$, is a set of first-order views $R_T = e_S^{R_T}$ for each predicate R_T of arity n in $\mathbb{A}_T$, with $e_S^{R_T}$ a relational algebra expression of arity n over the alphabet $\mathbb{A}_S$.

In the first-order setting, it can be proved that the definition of equivalence (or lossless transformation) between S and T corresponds to the following condition over the schemas and mappings: $(\mathbb{C}_S \cup M_{S \rightarrow T}) \equiv (\mathbb{C}_T \cup M_{T \rightarrow S})$, which really means $I(\langle \mathbb{A}_S \cup \mathbb{A}_T, \mathbb{C}_S \cup M_{S \rightarrow T} \rangle) = I(\langle \mathbb{A}_S \cup \mathbb{A}_T, \mathbb{C}_T \cup M_{T \rightarrow S} \rangle)$; see [13, 14].

Transformation patterns.

Transformation patterns are crafted templates describing a specific structure of schema transformation with the constraints necessary to ensure its losslessness [12]. The two basic lossless

transformation patterns are *vertical decomposition* and *horizontal decomposition*. We introduce them via two basic examples.

Given two schemas S and T as follows:

$S = \langle \{p(a, b, c)\}, (p.b \to p.c)\} \rangle$
$T = \langle \{q(a, b), r(b, c)\}, (r.b \to r.c), (q.b = r.b)\} \rangle$

The schemas S and T have the same information capacity since there is a lossless transformation through the following mappings – characterising the vertical decomposition in the classical database literature:

$M_{S \to T} = \{(q = \pi_{ab}\ p), (r = \pi_{bc}\ p)\}$
$M_{T \to S} = \{(p = q \bowtie r)\}$

The vertical decomposition transformation pattern maps a schema with a join dependency (i.e., a key dependency, or a functional dependency, or a multivalued dependency) to its vertical decomposition, with all the appropriate dependencies in both schemas to guarantee losslessness.

As an example of a lossless horizontal decomposition transformation, consider the schema U:

$$U = \langle \{q(a, b), r_1(b, c), r_2(b, c)\}, \{(r_1.b \to r_1.c), (r_2.b \to r_2.c), \\ (r_1.c = \{\text{“}k\text{”}\}), (r_2.c \not\subseteq \{\text{“}k\text{”}\}), \\ (\pi_b\ q = \pi_b\ r_1 \cup \pi_b\ r_2), (\pi_b\ r_1 \cap \pi_b\ r_2 = \emptyset)\} \rangle$$

The schemas T and U have the same information capacity since there is a lossless transformation through the following mappings – characterising the horizontal decomposition via the condition $\sigma_{c=\text{“}k\text{”}}\ r$:

$M_{T \to U} = \{(r_1 = \sigma_{c=\text{“}k\text{”}}\ r), (r_2 = \sigma_{c \neq \text{“}k\text{”}}\ r)\}$
$M_{U \to T} = \{(r = r_1 \cup r_2)\}$

The horizontal decomposition transformation pattern maps a schema to a horizontally decomposed one via a selection condition, with all the appropriate dependencies in both schemas to guarantee losslessness.

We can also observe that also S and U have the same information capacity, since they are related by a sequence of lossless transformations.

A special case of horizontal decomposition is the lossless transformation leading to a SQL NULL-free schema. According to the logic theory of SQL NULL values [31, 32], a schema has the same information capacity as an horizontally decomposed one via a NULLABLE condition over some attribute. Whenever there is a NULLABLE constraint over an attribute, a table can be losslessy decomposed into two tables, one having all the attributes but not the NULLABLE one, and the other having all the attributes but with a NOT NULL constraint replacing the NULLABLE constraint.

We have identified several lossless transformation patterns [14, 12], which can be used to design a Semantic SQL Transducer allowing for arbitrary data restructuring processes, whenever we want to guarantee that no information is lost during the restructuring process. The transformation patterns identify the lossless mappings from S to T and from T to S and the constraints of the two databases S and T, needed to design a correctly working Semantic SQL Transducer, as described in Section 3.

Reverse Engineering as Semantic Layer

A very special data restructuring process is the *reverse engineering* process, which looks for the lossless transformation from a source database schema to the schema corresponding to its conceptual schema – see [27] for a survey. This is the scenario we have presented in Section 2: we want to expose the source data with a vocabulary that corresponds to its conceptual schema, useful for the business perspective. If the transformation is lossless, we have the guarantee that no information is lost, and that high-level users can query and update freely the transformed database, in this case the database organised in a meaningful structure. More specifically, we want to losslessly transform a source schema into a schema in 6^{th} normal form with explicit *Object Identifiers* (OIDs) to identify "entities", namely instances of entity types. Object identifiers can be implemented by surrogate keys, URIs, or UUIDs. This form is called an Abstract Relational Model (ARM) [33]. Given an arbitrary database schema, there exists a unique *Canonical Abstract Relational Model* (CARM) for that schema, based on the 5^{th} or 6^{th} normal forms, which plays the role of the *Core Conceptual Schema* of the original database expressible in conceptual modelling languages such as ORM, EER, UML class diagrams, or in RDF-based modelling languages. In order to show losslessness, we consider the set of legal database instances of the CARM schema by projecting away the OID attributes of entities; as a consequence, queries or updates over the CARM schema can never explicitly ask or update OID values.

In order to understand how to losslessly transform a database schema into an equivalent one (the CARM) which includes OIDs, let's consider the following basic example. Assume we have a schema in 5^{th} normal form, so that the only constraints within a table are key constraints, and the constraints across tables are foreign keys or inverse foreign keys, for example:

Employee(ssn,name), works-in(ssn,depname), Department(depname,address)
Employee.ssn $\rightarrow$ Employee.name
works-in.ssn $\rightarrow$ Employee.ssn
works-in.depname $\rightarrow$ Department.depname
Department.depname $\rightarrow$ Department.address

A domain expert should recognise that Employee and Department are *entity types*, while works-in is a *relationship type*. As a rule of thumb, we can recognise entity types since they should be the target of at least a foreign key with a relationship type as source, while a relationship type should have at least one attribute as the source of a foreign key with an entity type as target. A new attribute with domain OID (disjoint from STRING and INTEGER) is added as a surrogate key to each entity type table, and coherently a new OID attribute replaces the attributes involved in a foreign key path from the entity type. The foreign key and inverse foreign key constraints holding across tables are duplicated to hold between the added OID attributes. Following our example, the lossless transformation of the above schema with added OIDs is:

Employee(eoid,ssn,name), works-in(eoid,doid), Department(doid,depname,address)
Employee.eoid $\rightleftarrows$ Employee.ssn
Employee.ssn $\rightarrow$ Employee.name
works-in.eoid $\rightarrow$ Employee.eoid

Figure 3: The Semantic SQL Transducer of the example

works-in.doid → Department.doid
Department.doid ⇄ Department.depname
Department.depname → Department.address

The schema above is the logical representation corresponding to the Entity-Relationship Diagram conceptual schema of Figure 3.

5. A complete example

Suppose we have a source database schema as described at the top of Figure 4. The schema is composed by a single table Source and a set of constraints. As humans, we can tell that the schema is about people, identified by their social security number, having one or more phone numbers, and possibly working in some department, identified by its name, having an address, which also uniquely identifies the department at that address.

Clearly, a lot of the information we just described about this database is hidden in the schema,

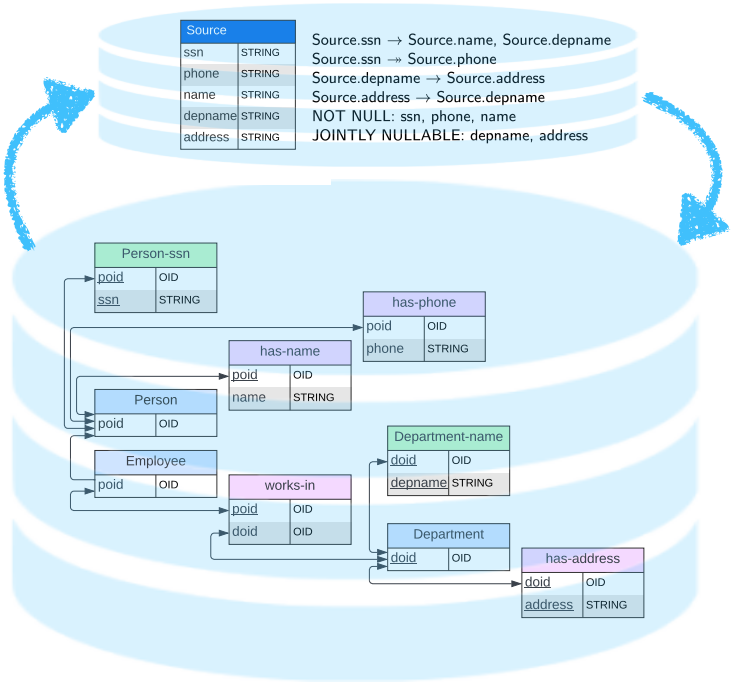

Figure 4: The Semantic SQL Transducer of the example

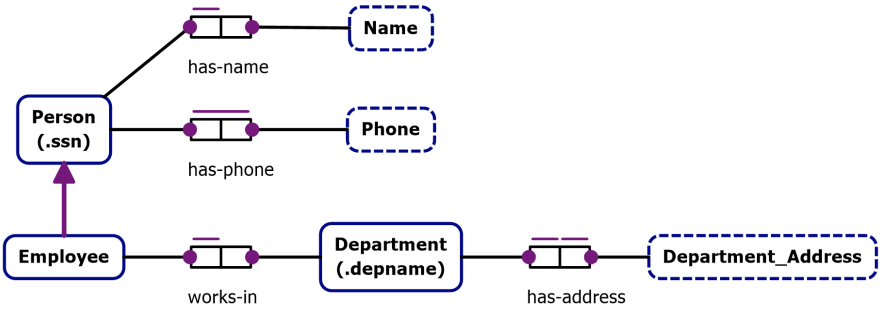

Figure 5: The conceptual schema of the example (in the ORM notation)

and we may wonder which could be its explicit conceptual schema to have a more direct understanding of the data. By jumping a little ahead, let's have a glance at the conceptual schema, expressed in the ORM notation, in Figure 5. That schema describes precisely, non-ambiguously, and formally the data as we were saying above. But how can we get this conceptual schema from the original source schema? First we notice that this conceptual schema has a direct representation as a logical schema in the relational setting: this is described at the bottom of Figure 4. The relational version of the conceptual schema denotes exactly the same legal databases as the ORM schema of Figure 5, but now in a pure relational setting. Note that the constraints of this schema are only key constraints and unary inclusion dependencies among OID datatypes – this property holds for all core conceptual schemas derivable from arbitrary source schemas.

The core conceptual schema is obtained form the source database schema by the application of a sequence of lossless transformation patterns, of the type we have briefly introduced in the previous Section. Since any lossless transformation step is accompanied by the mappings in the two directions, we get from the transformation process also the views from the source data to the conceptual data *and* the views from the conceptual data to the source data. In our example, these mappings are shown in Figure 6. We have devised a methodology driving the design of the correct sequence of lossless transformation patterns leading to a core conceptual schema from a source schema.

So now we have all the ingredients (the two sets of constraints and the two mappings) to finalise the implementation of a Semantic SQL Transducer for the source schema as explained in Section 4. Many things can be done once we have the transducer in place. We can understand what the source data is about. We can query the source database with SQL queries using only the core conceptual schema table names, bridging therefore the gap between business models and available data sources. Along these lines, business users can directly update the source data using just the conceptual vocabulary. In the other direction, legacy queries over the source database can be explained by looking at their expansion in terms of the conceptual schema. The Semantic SQL Transducer provides a well-founded semantic layer for data catalogues. Source data structured according its core conceptual schema makes data analysis much more effective, since the correlations, the classifications, and the similarities of the data elements are much more meaningful when done in business terms. Indeed, we observe that the conceptual schema (Figure 5) is materialised in the relational database (lower part of Figure 4) using exactly the

$$
\begin{aligned}
\text{Person} &= \varrho_{[\text{ssn/poid}]}(\pi_{\text{ssn}}(\text{Source})) \\
\text{Person-ssn} &= \varrho_{[\text{ssn}_1/\text{poid},\text{ssn}_2/\text{ssn}]}(\sigma_{\text{ssn}_1=\text{ssn}_2}(\pi_{\text{ssn}_1,\text{ssn}_2}(\text{Source} \times \text{Source}))) \\
\text{has-name} &= \varrho_{[\text{ssn/poid}]}(\pi_{\text{ssn,name}}(\text{Source})) \\
\text{has-phone} &= \varrho_{[\text{ssn/poid}]}(\pi_{\text{ssn,phone}}(\text{Source})) \\
\text{Employee} &= \varrho_{[\text{ssn/poid}]}(\pi_{\text{ssn}}(\sigma_{\text{depname NOT NULL}}\text{Source})) \\
\text{works-in} &= \varrho_{[\text{ssn/poid}]}(\pi_{\text{ssn,depname}}(\sigma_{\text{depname NOT NULL}}\text{Source})) \\
\text{Department} &= \varrho_{[\text{depname/doid}]}(\pi_{\text{depname}}(\sigma_{\text{depname NOT NULL}}\text{Source})) \\
\text{Department-name} &= \varrho_{[\text{depname}_1/\text{doid},\text{depname}_2/\text{depname}]} \\
&\quad (\sigma_{\text{depname}_1=\text{depname}_2}(\pi_{\text{depname}_1,\text{depname}_2} \\
&\quad\quad ((\sigma_{\text{depname NOT NULL}}\text{Source}) \times (\sigma_{\text{depname NOT NULL}}\text{Source})))) \\
\text{has-address} &= \varrho_{[\text{depname/doid}]}(\pi_{\text{depname,address}}(\sigma_{\text{depname NOT NULL}}\text{Source})) \\
\\
\text{Source} &= \pi_{\text{ssn,phone,name,depname,address}} \\
&\quad \bowtie (\text{Person, Employee, has-phone, has-name,} \\
&\quad\quad \text{works-in, has-address, Person-ssn, Department-name})
\end{aligned}
$$

Figure 6: The lossless mappings from source to CARM and viceversa

vocabulary appearing in the ER diagram (Figure 5), and that both the source and the conceptual databases (Figure 4) are connected via two sets of mappings (Figure 6), which implement the lossless translations of the queries and of the updates at both ends.

6. Conclusions

We have introduced in this paper a tool to losslessly restructure relational data, allowing for seamless views of the data, which can be queried and updated at both ends maintaining consistency and integrity. A special kind of transformation is when a database is restructured according to its conceptual schema, providing therefore a materialised copy of the data, always in sync with it, using the actual vocabulary understood by the business. The goal of this tools is to support Knowledge-based Artificial Intelligence for Data Science, enabling advanced and interpretable data analytics: by integrating semantic layers into data analysis pipelines, data scientists can develop more robust models and interpret predictions more effectively. This ensures data quality, accessibility, integrity, meaning, and interoperability.

We have briefly experimented this framework over the data collected from construction process monitoring within the *Confucious* project of the Free University of Bozen-Bolzano. We are planning a more serious evaluation in the context of a data integration project in a large enterprise.

This paper is a revised and extended version of [34]. This long-standing work has been realised through collaborations and discussions with Nicola Pedot, Nonyelum Ndefo, Francesco Sportelli, Sergio Tessaris, Volha Kerhet, Nhung Ngo, Paolo Guagliardo, David Toman, Grant Weddell, Alex Borgida, Terry Halpin, Jan Hidders, Sebastian Link.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] P. Ciaccia, D. Martinenghi, R. Torlone, Conceptual constraints for data quality in data lakes, in: Proceedings of the 1st Italian Conference on Big Data and Data Science (itaDATA 2022), Milan, Italy, September 20-21, 2022, 2022, pp. 111–122.

[2] M. Hameed, F. Naumann, Data preparation: A survey of commercial tools, SIGMOD Rec. 49 (2020) 18–29.

[3] M. Console, M. Lenzerini, Editorial: Special issue on quality aspects of data preparation, ACM J. Data Inf. Qual. 15 (2023) 40:1–40:2.

[4] P. Cudré-Mauroux, Leveraging knowledge graphs for big data integration: the XI pipeline, Semantic Web 11 (2020) 13–17.

[5] H. Dibowski, S. Schmid, Y. Svetashova, C. Henson, T. Tran, Using semantic technologies to manage a data lake: Data catalog, provenance and access control, in: 12th International Workshop on Scalable Semantic Web Knowledge Base Systems, 2020, pp. 65–80.

[6] D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, R. Rosati, Ontology-based data access and integration, in: L. Liu, M. T. Özsu (Eds.), Encyclopedia of Database Systems, Second Edition, Springer, 2018.

[7] J. Reis, M. Housley, Fundamentals of Data Engineering, O'Reilly Media, 2022. URL: https://books.google.it/books?id=3qd2EAAAQBAJ.

[8] E. Franconi, U. Sattler, A data warehouse conceptual data model for multidimensional aggregation, in: Intl. Workshop on Design and Management of Data Warehouses, DMDW'99, 1999.

[9] L. Lubyte, S. Tessaris, Automatic extraction of ontologies wrapping relational data sources, in: International Conference on Database and Expert Systems Applications, Springer, 2009, pp. 128–142.

[10] D. Calvanese, E. Franconi, First-order ontology mediated database querying via query reformulation, in: A Comprehensive Guide Through the Italian Database Research Over the Last 25 Years, volume 31, Springer, 2018, pp. 169–185.

[11] T. Abgrall, Formalization of data integration transformations, in: New Trends in Database and Information Systems - ADBIS 2022, volume 1652, Springer, 2022, pp. 615–622.

[12] T. Abgrall, Schema decomposition via transformation patterns, in: 32nd Symposium of Advanced Database Systems (SEBD-24), volume 3741, CEUR-WS.org, 2024, pp. 551–564.

[13] N. Ndefo, E. Franconi, On preserving information in schema transformations: A constructive perspective, in: 2nd IEEE International Conference on Artificial Intelligence and Knowledge Engineering, AIKE 2019, 2019, pp. 57–64.

[14] N. Ndefo, E. Franconi, A study on information-preserving schema transformations, International Journal of Semantic Computing 14 (2020) 27–53.

[15] I. Kobayashi, Losslessness and semantic correctness of database schema transformation: another look of schema equivalence, Information Systems 11 (1986) 41–59.

[16] R. Hull, Relative information capacity of simple relational database schemata, SIAM Journal on Computing 15 (1986) 856–886.
[17] R. J. Miller, Y. E. Ioannidis, R. Ramakrishnan, The use of information capacity in schema integration and translation, in: 19th International Conference on Very Large Data Bases (VLDB-1993), 1993, pp. 120–133.
[18] X. Qian, Correct schema transformations, in: EDBT'96, 5th International Conference on Extending Database Technology, Springer, 1996, pp. 114–128.
[19] A. Poulovassilis, P. McBrien, A general formal framework for schema transformation, Data & Knowledge Engineering 28 (1998) 47–71.
[20] P. McBrien, A. Poulovassilis, Data integration by bi-directional schema transformation rules, in: Proceedings 19th International Conference on Data Engineering (ICDE-03), 2003, pp. 227–238.
[21] N. A. Mian, S. A. Khan, N. A. Zafar, Database reverse engineering methods: What is missing?, Res. J. Recent Sci. 2 (2013) 49–58.
[22] N. Lammari, I. Comyn-Wattiau, J. Akoka, Extracting generalization hierarchies from relational databases: A reverse engineering approach, Data & Knowledge Engineering 63 (2007) 568–589.
[23] I. Astrova, Reverse engineering of relational databases to ontologies, in: European Semantic Web Symposium, Springer, 2004, pp. 327–341.
[24] C. Soutou, Relational database reverse engineering: algorithms to extract cardinality constraints, Data & Knowledge Engineering 28 (1998) 161–207.
[25] M. Andersson, Extracting an entity relationship schema from a relational database through reverse engineering, in: International Conference on Conceptual Modeling, Springer, 1994, pp. 403–419.
[26] R. H. Chiang, T. M. Barron, V. C. Storey, Reverse engineering of relational databases: Extraction of an EER model from a relational database, Data & knowledge engineering 12 (1994) 107–142.
[27] J.-L. Hainaut, Introduction to database reverse engineering, LIBD Lecture Notes (2002).
[28] J.-L. Hainaut, M. Chandelon, C. Tonneau, M. Joris, Contribution to a theory of database reverse engineering, in: [1993] Proceedings Working Conference on Reverse Engineering, IEEE, 1993, pp. 161–170.
[29] J.-L. Hainaut, C. Tonneau, M. Joris, M. Chandelon, Schema Transformation Techniques for Database Reverse Engineering, Springer Verlag, 1993, pp. 353–372.
[30] W. Ma, C. M. Keet, W. Oldford, D. Toman, G. E. Weddell, The utility of the abstract relational model and attribute paths in SQL, in: Knowledge Engineering and Knowledge Management - 21st International Conference, EKAW, Springer, 2018, pp. 195–211.
[31] E. Franconi, S. Tessaris, Relational algebra and calculus with SQL null values, CoRR abs/2202.10898 (2022). arXiv:2202.10898.
[32] E. Franconi, S. Tessaris, On the logic of SQL nulls, in: 6th Alberto Mendelzon International Workshop on Foundations of Data Management, CEUR Workshop Proceedings, 2012, pp. 114–128.
[33] A. Borgida, D. Toman, G. E. Weddell, On referring expressions in information systems derived from conceptual modelling, in: Conceptual Modeling - 35th International Conference, ER 2016, 2016, pp. 183–197.

[34] T. Abgrall, E. Franconi, Understanding the semantic sql transducer, in: J. P. A. Almeida, C. Di Ciccio, C. Kalloniatis (Eds.), Advanced Information Systems Engineering Workshops, Springer Nature Switzerland, Cham, 2024, pp. 135–146.
[35] J. Albert, Y. E. Ioannidis, R. Ramakrishnan, Equivalence of keyed relational schemas by conjunctive queries, J. Comput. Syst. Sci. 58 (1999) 512–534.
[36] A. Khatiwada, R. Shraga, W. Gatterbauer, R. J. Miller, Integrating data lake tables, Proc. VLDB Endow. 16 (2022) 932–945.
[37] P. Papotti, R. Torlone, Schema exchange: Generic mappings for transforming data and metadata, Data & Knowledge Engineering 68 (2009) 665–682.