Information Integration using the Typed Graph Model

Fritz Laux
Fakultät Informatik
Reutlingen University
D-72762 Reutlingen, Germany
email: fritz.laux@fh-reutlingen.de

Malcolm Crowe
School of Computing
University of the West of Scotland
Paisley PA1 2BE, UK
email: malcolm.crowe@uws.ac.uk

***Abstract*—Schema and data integration have been a challenge for more than 40 years. While data warehouse technologies are quite a success story, there is still a lack of information integration methods, especially if the data sources are based on different data models or do not have a schema. Enterprise Information Integration has to deal with heterogeneous data sources and requires up-to-date high-quality information to provide a reliable basis for analysis and decision making. The paper proposes virtual integration using the Typed Graph Model to support schema mediation. The integration process first converts the structure of each source into a typed graph schema, which is then matched to the mediated schema. Mapping rules define transformations between the schemata to reconcile semantics. The mapping can be visually validated by experts. It provides indicators and rules to achieve a consistent schema mapping, which leads to high data integrity and quality.**

Keywords–data integration process; typed graph model; schema mapping; mapping rules; data quality.

I. INTRODUCTION

Information integration, integrating data from different sources, enables us to gain new knowledge and insights that help to make predictive analysis, coordinate complex processes and control systems. Depending on the application the mediated data can be materialized as in a Data Warehouse (DW) or each query can access the data sources to get an up-to-date result, which is called *virtual integration*. Many situations need up-to-date or near real-time information, which can only be achieved by virtual integration. This is the case not only in emergency situations like fighting epidemic, earthquake, and other disaster aid but also in industry production.

In Enterprise Information Integration (EII), the integration of heterogeneous data sources is usually achieved by a manually supervised process to ensure a high-quality mediated global schema. In this paper, we concentrate on *supervised* schema integration, i.e., the semantic data integration problem. Much research has been conducted to automatically match and map data sources [1][2], but the quality of the results are usually not sufficient for EII systems [3]–[6]. Thus, additional manual changes using context and other semantic information are necessary to build a high-quality global/mediated schema, i.e., an *engineered schema* [7].

The manual improvement and quality checking of the mediated schema should be supported by software that visualizes the semantics of the data integration and the impact and interplay of any changes.

Let us illustrate our approach with an example scenario presented in Figure 1. It is borrowed from Crowe et al. [8]. The

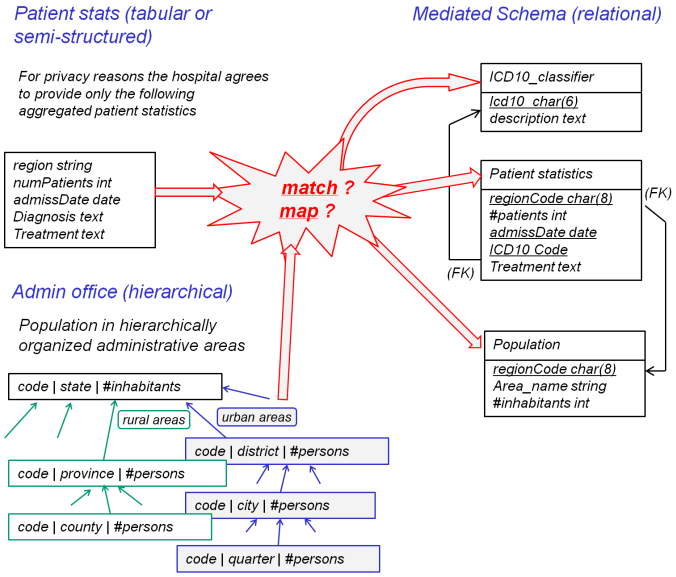

Figure 1. Data integration scenario (Running example)

World Health Organization reports on emergencies/epidemics, using data provided in many different formats by national or regional autonomous actors like national health authorities or hospitals. For simplicity, we only show one hospital providing aggregated *tabular data* on patients grouped by regions, admission date and diagnosis. For each group element the number of patients and their treatment are recorded. The statistics admin offices deliver demographic data in a *hierarchical structure* reflecting the administrative areas. We want to create an integrated/mediated schema that combines these sources using the International Classifier for Diseases (ICD). Given a *relational* Integration Schema, the question is "how do we find matches for the data items and transform them".

We can identify five possible problems in this example.

From Figure 1, we need to identify the matching between *region string* of the hospital patient stats and the *regionCode* in the mediated schema despite the different coding (Problem 1). There might be different spelling or naming of *region* strings in different hospitals. These conflicts need to be resolved in the mediated schema (Problem 2). Preserving the semantics (Problem 3) requires the *ICD10_classifier* for the mediated schema to be obtained from the *diagnosis* text. The population data from administrative areas need to be mapped into relations preserving the hierarchy (Problem 4). And finally (Problem 5), we need consistency when multiple mapping paths exist

 ISBN: 978-1-61208-857-0

like from *region* (hospital stats) to *regionCode* (Population), one via *code* (admin office) and the other either direct or via *regionCode* (Patient statistics).

In Figure 1, we used different graphical representations for the schemata to reflect the various data models and structures. But a consistent graphical representation would be better to solve the matching/mapping task. The Typed Graph Model (TGM) [9] represents a flexible model with enough expressive power to cover most data structures comprehensively and support several abstraction levels. This is why we have chosen the TGM for the schema mediation and data mapping.

Data integration has been intensively investigated from a theoretical point of view [10]–[12] leading to mainly two concepts, *Global As View* (GAV) and *Local As View* (LAV). As the terms indicate the global (resp. local) data are expressed as view of local (resp. global) data. The ACM digital library alone retrieves 163 matches for the key words "GAV or LAV". These papers are of great value to understand and specify the correspondences between two or more databases. But using description logic as formal specification gives little help to identify synonyms and homonyms or discover their semantics. Where sources of low quality overlap, it is important to remove redundancy while seeking to benefit from any extra information. The mapping of heterogeneous data sources to a specific target does not allow a fully automatic procedure if high data quality is required [5][3][13][14].

A. Contribution

Our idea is to use the TGM to support, formalize, and visualize data integration. The TGM helps to create a mediated target schema, in contrast to the less formalized Extract-Transform-Load process used in Data Warehousing. We propose a process, which divides the integration task into two phases: first, model all sources and the target using TGM, then, second, match and map the source models into the target model. The proposed process is well defined and combines for the first time *supervised* semi-automatic matching with mapping and merging of data. It provides rules for mapping and conflict resolution and defines criteria for quality control. We illustrate all integration steps using our running example.

B. Structure of the Paper

Section II briefly presents theoretical data integration work and practical experiences with emphasis on high-quality integration. In Section III the integration framework is presented. It consists of four activity steps:

1) Modeling the source data structures with TGM (Subsection III-C)
2) Defining the target schema using TGM (Subsection III-D)
3) Match/Map source with target data, resolve conflicts, and define necessary transformation (Subsection III-E)
4) Check and improve quality (Section IV).

Quality criteria and measures are developed in Section IV to help check and improve the integration quality. The paper ends with a summary of our findings and gives an outlook on ideas for future work.

II. Related Work

Since the middle of 1980s many papers on data integration have been published. In the following review, we restrict the focus to high-quality integration targeting EII.

A. Data Integration Overview

The papers of Sheth and Larson [15] give an introduction to federated database systems. Later textbooks of Öszu and Valduriez [16] and also Leser and Naumann [17] describe different integration methods and present general approaches for schema matching and mapping. Because our focus is on high-quality integration the experiences and considerations of Bernstein/Haas [3] and Halevy et al. [13] with real live projects are of high value to us. Both papers emphasize the need to integrate heterogeneous data sources including email, order documents, warranties, and other un- or semi-structured data.

Laura Haas states in her paper [6] that "despite the weighty body of literature, the information integration challenge is far from solved, especially in the enterprise context". As "Big I" challenges she identifies *entity resolution* and the lack of *theoretical and practical guidance* to make schema integration choices, noting the lack of a "broader framework" with quality control, which "considers the entire end-to-end integration process". This statement is confirmed by many papers that address the integration of Web data [18]–[20], XML data [20], or RDF data [21][22] in a declarative way but with little help on how to proceed in practice.

B. Data Integration using Graph Models

Some authors use a Graph Data Model (GDM) for data analysis and transformation. GRADOOP [23] and Pregel [24] are example prototypes for this approach. Most of the integration of graph databases is instance level based and uses graph transformations, see [25][2], but the question of how well the integration matches the original semantics and schemata of the sources is not addressed.

None of the papers really addresses how to match and map differently structured data elements by preserving the semantics of the sources. Practical guidance is available from de Sousa and del Val Cura [26] only for the mapping from the Extended Binary Entity Relationship (EB-ER) model to a Property Graph Model (PGM).

The only publication we are aware of that tries to improve data integration quality is Gelman [27]. In his paper he develops a theory that helps to produce accurate data integration output from multiple, overlapping, and inaccurate sources. He assumes that errors are not random and that "complementary" information helps to select the most accurate data. This approach could also help with the entity resolution problem.

Another problem with virtual data integration is that the global database may not be consistent with respect to integrity constraints. Bertossi and Bravo [28] recommend querying only the consistent part of the global database. They define a consistent answer to a query when the result is the answer to every "repair" of the global database, i.e., a maximal subset of the global database that satisfies the integrity constraints. The solution to the problem is quite general and conceptually

clear, however an implementation is still missing. The authors demand that these issues must be "addressed in order to use those solutions in real database applications".

III. The Integration Framework

The data integration framework uses the TGM as intermediate data model. This is why the TGM is shortly presented here. A detailed and formal introduction can be found in [9]. The data integration process consists of transforming the sources and target into Typed Graph Schemata (TGS) and then make the matching and mapping of schema elements.

A. The Typed Graph Model

The TGM combines schema support, complex data structures and abstraction using sub-graphs. Because of its rich semantics, we use it to capture the source and target semantics as accurately as possible. The source and target schemata act as a means of quality control.

The TGM informally constitutes a directed property hypergraph that conforms to a schema. Formally, the TGM is defined as quadruple $TGM = (N, E, TGS, \phi)$ where:

- N is the set of named (labeled) nodes n with data types from N_S of schema TGS.
- E is the set of named (labeled) edges e with properties of types from E_S of schema TGS.
- TGS is a typed graph schema defined as tuple $TGS = (N_S, E_S, \rho, T, \tau, C)$, where N_S, E_S are vertices and edges of the graph schema, ρ defines the hyper-edges with its cardinalities τ. T is a set of data types used for vertices and edges, and finally C is a set of integrity constraints, which the graph database must obey.
- ϕ is a homomorphism that maps each node n and edge e of TGM to the corresponding type element of TGS,

As graphical representation for the TGS, we adopt the Unified Modeling Language (UML) to visualize the data model, which makes it useful for visualization tools to support the human controlled information matching and mapping. The visualization allows the modeling expert to validate the matching and mapping. The possible abstraction via sub-graphs facilitates the overview and management of complex and large models.

A data model can act as a kind of **supermodel** if it is general enough to capture all popular models. Hull [29] and Atzeni et al. [30][31] describe such a supermodel that subsumes all popular data models including the Relational Model (RM), ERM, XML, Object Oriented Model (OOM), Object Relational (OR), and XSD. It consists of the following meta-constructs: *lexical, abstract, aggregation, generalization,* and *function*. The TGM can represent these meta-constructs in an information preserving way [32] by matching one-to-one lexical elements with properties, abstract constructs with nodes, aggregation and generalization constructs with the corresponding edge types, and functions with directed edges having multiplicity 1. This implies that the TGM is able to map the above-mentioned data models without loss of any information.

B. The Data Integration Process

The benefit of information integration is maximized when source data are integrated with their full semantics. We believe a key success factor is to model the sources and target information as accurately as possible. The expressive power and flexibility of the TGM allows to describe the meta-data of the sources and target precisely and in the same model, which simplifies the matching and mapping of the sources to the target.

The data integration process consists of five tasks:

1) model sources as TGS $S_i \quad (i = 1, 2, ..., n)$
2) model target schema T as TGS G
3) match and map sources S_i with TGS G
4) check and improve quality
5) convert TGS G back to T again

The full process is depicted in Figure 2 as a workflow modeled in BPMN. The 4^{th} task of the workflow is crucial for EII and other data integration projects, which demand highly accurate information quality. It might turn out that the resulting quality is not sufficient. As consequence the process might have to be iterated with different mappings in order to improve the quality. The last task is only necessary if the target schema is not a graph schema.

The first and second tasks consist of two alternatives depending on the pre-existence of source schemata, resp. target schema. If a schema already exists it is only necessary to transform it into a TGS. If a source has no schema, it is then necessary to collect structure and type information from a data expert or from additional information. This information is necessary before an appropriate TGS can be created. At least a minimal schema is required for every data source. This extra effort has the advantage to ensure a better data quality.

C. Model the Data Sources as TGS (Task 1)

The relevant data must first be identified together with its meta-data if available. This includes coding and names for the data items. The measure units and other meta-data provided by the data owner are used to adjust all measures to the same scale.

If the source is a database or other rigid structured data, the modeling of a TGS is rather simple and there are publications [33][34] that propose automatic schema conversion. The paper of Laux [9] gives some examples how to transform relational, object oriented, and XML-schemata into a TGS. If a schema already exists for a data source it may seem to be extra work to model it again as TGS. The benefit comes later when we look for matches because the schemata are all based on the same model. In addition, the quality of the matching can be checked formally with graph analysis.

If the source is unstructured or semi-structured, e.g., documents or XML/HTML data, concepts and mechanisms from Information Retrieval (IR) and statistical analysis may help to identify some implicit structure or identify outliers and other susceptible data. If the data are self-describing (JSON, key-value pairs, or XML) linguistic matching can be applied with additional help from a thesaurus or ontology. Nevertheless, it

 ISBN: 978-1-61208-857-0

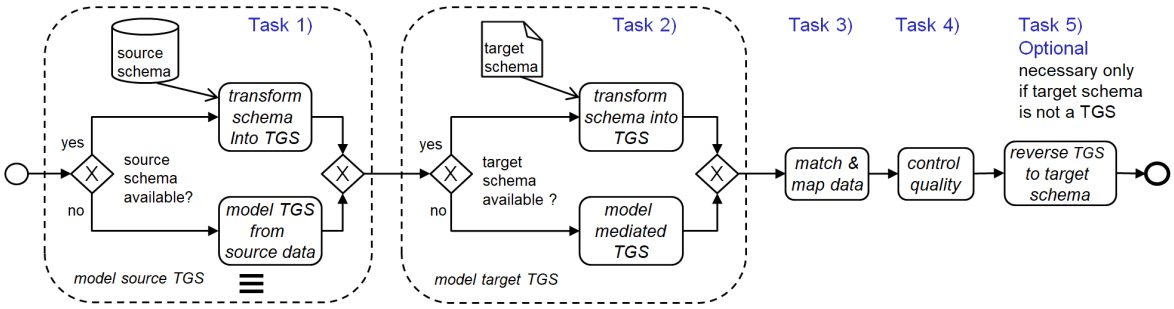

Figure 2. Workflow of information integration using the TGM

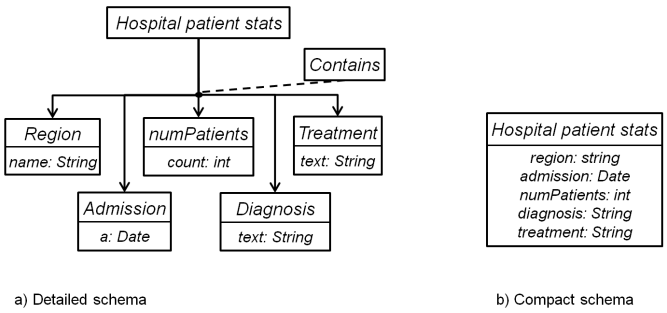

Figure 3. TGS in UML notation for the patient statistics

is advisable to validate the matching with instance data or an information expert.

As example for semi-structured source data, we take the hospital patient statistics from our running example. The data for the statistics are entered in a form. We present two possible TGS in UML notation in Figure 3. Part a) shows a detailed schema where each data element is modeled as a node with its corresponding property and (simple) data type. In part b) the hospital patient stats are modeled as a complex data type. This little example demonstrates already the flexibility of the model in terms of detail and abstraction.

D. Model Mediated TGS (Task 2) and Target Schema (Task 5)

In general, many integration schemata are possible. In most cases the mediated schema tends to cover the union of the source schemata. Petermann et al. [35] present an automatic graph instance integration with the help of a Unified Metadata Graph (UMG). This method can be applied to generate a mediated graph TGS if the UMG is replaced by the data source schema graphs. Another approach for unstructured data is proposed by Buneman et al. [36] by union of the source graphs.

If a target schema T is given, but not already as TGS, it needs to be transformed to a TGS. In most cases a 1–1 transformation is possible. This can be done automatically using the same techniques as mentioned in III-C. This is the only case where Task 5) has to be executed. The TGS has to be reversed in this case to the target schema T again by applying the inverse transformation.

Let us return to our running example and model the two

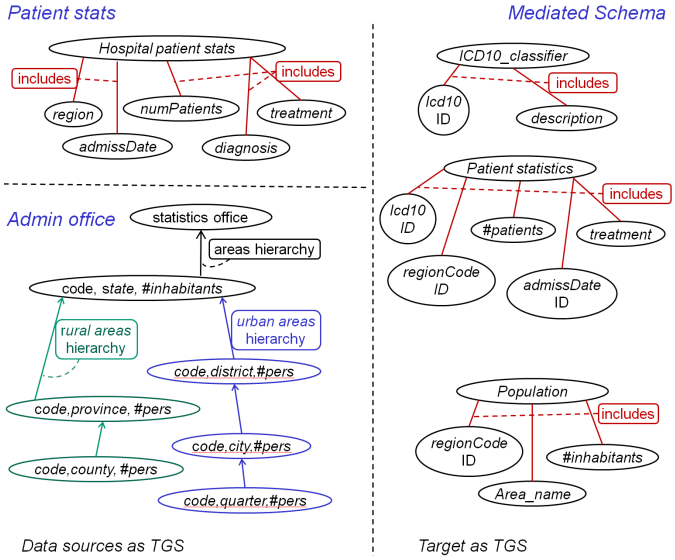

Figure 4. Running example as TGS (data types not shown for simplicity)

data sources and the mediated target as TGS. The result is shown in Figure 4. The nodes of the schemata are visualized with property names inside the ovals. The data types of the properties are suppressed, but the edge types (labels) are color coded and connected by dashed lines with the corresponding edges. In the next task, we have to decide on the matching and mapping of vertices.

E. Matching & Mapping Source TGS to Target TGS (Task 3)

Task 3 addresses problem 1 from the Introduction I. The matching step only identifies nodes and edges that are related, not necessarily one-to-one. In our example hospital statistics, apart from national language differences, the English names: cases, positive cases, reported cases, hospitalized, etc. could mean all the same or could mean different things. The TGM can help to identify, visualize, and match nodes and edges correctly using type and edge information (edge type, structure analysis, description). The matching can be supported by linguistic methods (name similarity, synonym and homonym list, thesaurus or ontology) and value analysis, e.g., using duplicates [37]. There are some publications that automate this process, see [1][2]. The use of TGM is flexible enough to support various data structures and visualizes the integration

process, which helps to identify and resolve mapping problems manually.

In the next step, we define mappings between the source and the target nodes. The mapping is mainly a manual task and the integration schema designer has the responsibility to choose the best quality (freshness, reliability, precision) data, resolve conflicts if redundant data are available and to correct apparently incorrect names. These include merge operations with conflict resolution (Problem 2), e.g., entity deduplication, overlapping conflicting data, identification of global data, and (data type) transformation. It is important to preserve the semantics as far as possible (Problem 3). This requires the knowledge of meta-data (data type, structure analysis, description), which is supported by the TGM. The mapping may include data grouping, e.g., grouping patients according to cost factors (ABC analysis). The workflow sequence and the mapping function suggest following the more natural Global-As-View (GAV) approach when implementing the mappings. Even if the integration of new sources more complicated compared to the Local-As-View (LAV) approach, it reflects the reality of overlapping data with different coding or semantics, which has to be resolved in both approaches.

To complete our running example, we present in Figure 5 the result of the mapping of our running example. As in Figure 4 on the left side are two TGS representing the data sources and on the right side is the *Mediated Schema* which can be converted back to the relational target schema given in Figure 1. This corresponds to the final step 5 of the workflow. For illustration purpose the Foreign Keys (FK) are indicated by dashed blue lines. The two data source graphs are semantically connected (*related with*) via the region information. The matching and mappings (red arrows) between source schemata and target TGS exemplify only some of necessary matching and mappings. Between *Hospital patient stats* (source Patient stats) and *regionCode* (mediated schema) exist two commutative mappings: (1) via *Patient statistics* and (2) via *region*. Both include an isomorphism (iso) and a projection (π).

F. Example Patterns

In order to illustrate the modeling power and flexibility of the TGM, we present a series of typical mappings that arise during schema integration and mediation. Such mapping patterns reoccur often and have a standardized solution.

1) Merge Pattern: The Merge pattern solves problem 2) of how to merge two or more data nodes of similar semantics. The different data sources can provide additional and overlapping data. Multiple sources may produce conflicting values or duplicates, differ in scale and coding, and have different resolution. Duplicates must be removed. Rules need to be established for conflicting values, e.g., prioritize the most reliable value or calculate a mean value if all sources are of similar quality. In case of different coding use translation tables. For scale and unit mismatches define value transformations. The merge pattern is useful to reconcile data from different sources and to improve data quality if the data is redundant.

Figure 6 shows a typical merge situation of two sources (Hospital and Clinic) with different region coding. The mapping M_{12} between Hregion and Cregion is required and allows to merge the overlapping data. In case of a value conflict the Cregion gets precedence.

2) Homomorphism Pattern: A Homomorphism is a structure preserving mapping that helps to transform the source schemata into a target schema and thereby solves Problem 4) from the Introduction. It preserves edges but allows multiple nodes to map to the same target node. This can be used for data aggregation. If the mapping is injective (one-to-one), we have an Isomorphism. It transforms the source schemata into an equivalent target schema.

In Figure 7 the homomorphism f is a mapping that transforms all nodes and edges from schema S onto nodes and edges of target G. In this example the patient nodes are mapped to the numPatients node of the target by incrementing the numPatients value. The edges between patient$x$ ($x = 1, 2$) and Hospital are mapped (green arrows) to the same edge (PatientStats–NumPatients).

3) Commutative Mapping Pattern: When defining the mappings between two schemata special care has to be taken if a target node can be reached via different paths. This happens for example when in the source schema two data items are related and both items are mapped to the same target node. In this case, we have to use the Merge pattern to resolve the conflict. But, if we preserve edges like in our example in Figure 8, we should have mappings that commute. A function chain is called commutative if and only if the order of the functions does not matter, i.e., $f_2 \circ f_1 = f_1 \circ f_2$. If special mappings are used like projection g and isomorphism iso then chances are good that the mapping chain is commutative. Communicative mappings are an essential criterion for a consistent mapping, and it helps to solve problem 5) from the Introduction. Commutativity is important because it guarantees that we have the choice between alternative mapping paths and still end up with the same target data. In Figure 8 it is irrelevant if the projection g to region is done first or the isomorphic mapping iso_1 to patientCode. We always end up with the same regionCode.

IV. Quality Criteria (Task 4)

A bipartite graph is a graph where the nodes are separated in two disjoint subsets with no edges within the subsets. This is the case with graph matching if we remove (in our mind) the connections inside the local schemata. If we consider such a bipartite graph, we can define the following quality criteria: A matching between the source and target graphs is called *maximum matching* if the number of matched vertices is maximized. If all nodes are matched, we call this a *perfect matching*. This guarantees that all nodes (data elements) from the sources are matched with target nodes and this gives us a criterion of how well the matching covers the integration task.

The theorem of Hall [38] states that there exists a perfect matching if all possible subsets of the source nodes have at least as many links to target nodes as the cardinality of this target subset. More formally, let $V = (S \cup G, E)$ be a bipartite graph of two disjoint sets S, G, then there exists a perfect matching if $\forall R \subseteq S$ the inequality $d(R) \geq |R|$ holds where $d(R) := |\{g \in G \mid r \in R \wedge (r, g) \in E\}|$ is the number of nodes in G linked to R. The perfect matching is a general criterion for the coverage or completeness of a data integration. If no

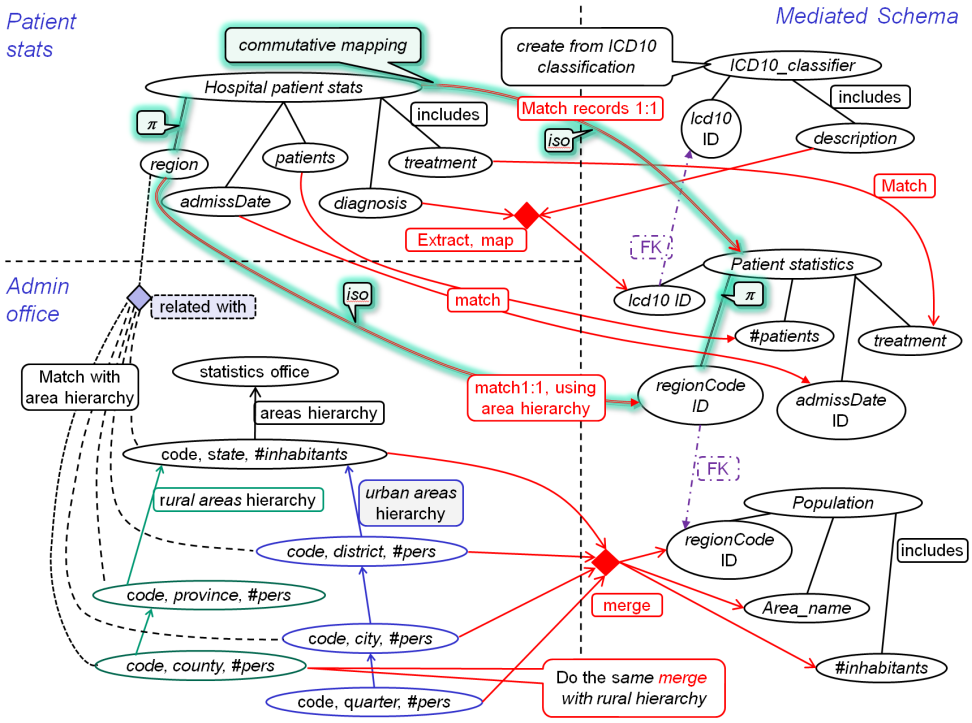

Figure 5. Matching and mapping result of the complete running example. To not overload the figure, only some of the matches and mappings (red arrows) are shown. The commutative mapping between *Hospital patient stats* and *regionCode* is highlighted (green glow).

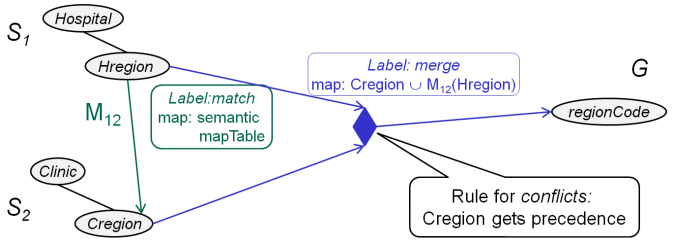

Figure 6. Merge example of two Patient stats sources (Hospital and Clinic) with map table and conflict resolution

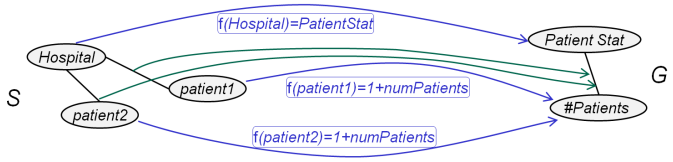

Figure 7. Example instance graph Homomorphism that sums patients

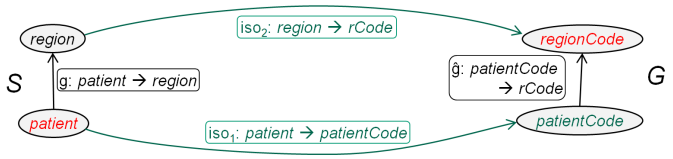

Figure 8. Commutative mapping from patient to regionCode

perfect match exists a merge conflict can arise and conflict resolution is necessary.

The theorem of Hall is only a formal quality criterion. In order to improve the semantic mapping quality, we may distinguish three cases on the instance level:

(1–1) Data are mapped 1–1 to compatible data types or via an enumeration list. This may include disjoint merge operations.

(n–1) Data are mapped to an aggregated value or a merge with redundant data.

(1–n) Data are distributed to multiple data elements. This can occur for address data that is split into separate fields or split up values that include tax.

A semantic mapping quality measure can be established by assigning 3 points to every 1–1 mapping, 2 points to every n–1 mapping. The n–1 mapping loses information compared to the 1–1 mapping; therefore, it receives a lower score. The 1–n mappings receive 1 to 3 points depending on the reliability of the split operation. Integration mappings with the highest sum represent the best match. There exist many other schema quality measures for completeness, correctness, minimality, etc. A nice overview on these measures is given in [39]. These measures are of great value to quantify the quality of a schema but give no direct help how to decide between mapping options.

Let us take Figure 8 as example and calculate the quality measure for the pictured mappings. The 1–1 mappings id_1, id_2, iso_1, and iso_2 have 3 points each and the projections g and $\hat{g}$ get 2 points each because of the n–1 mapping. The mapping chains $g \circ iso_2$ and $iso_1 \circ \hat{g}$ are commutative and yield

the same score 5 (= 3 + 2), which confirms the equivalence of both mapping paths.

The matching and mapping in Figure 5 also show a pair of commutative mappings from *Hospital patient stats* to *regionCode* with equal quality scores (see arrows with green glow, *iso* = 3 points, projection π = 2 points). If the scores differ, it is recommended to only use the mapping with the higher score because it represents a higher mapping quality.

V. Conclusion and Future Work

This paper presented an information integration process using the TGM with emphasis on semantically high quality as required for enterprise or government applications. Due to the TGS with predefined and user-defined data types, the TGM improves the formal data quality compared to other data integration approaches. Integration patterns and quality criteria give guidelines for practical use of the matching and mapping task. The whole process was illustrated by a running example.

We conclude that high-quality integration with an engineered target schema is feasible. Some integration steps may be supported by automatic methods but still require human support with meta-data and context knowledge to achieve high-quality results. The development of software to support schema generation as well as the matching and mapping remains as future work. In particular, the current expert-based mappings in Task 3 could be enhanced with semi-automated suggestions to the user with possible mapping options. One approach for such a program could use some elements from the academic prototypes GRADOOP [23], IncMap [40] or Pregel [24] to reduce the development effort.

References

[1] E. Rahm and P. Bernstein, "A survey of approaches to automatic schema matching", The VLDB Journal 10, pp. 334-350, 2001, DOI: 10.1007/s007780100057

[2] S. Melnik, H. Garcia-Molina, and E. Rahm, "Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching", ICDE, pp. 117–128, 2002, DOI: 10.1109/ICDE.2002.994702

[3] P. Bernstein and L. Haas, "Information integration in the enterprise", Communications of the ACM 51(9), pp. 72–79, 2008

[4] B. Golshan, A. Halevy, G. Mihaila, and W.-Ch. Tan, "Data Integration: After the Teenage Years", ACM PODS 2017, pp. 101–106

[5] D. Strong and O. Volkoff, "Data Quality Issues in Integrated Enterprise Systems", Proceedings of the MIT ICIQ Conference, no page numbers, 2005, [online] URL: http://mitiq.mit.edu/ICIQ/Documents/IQ Conference 2005/Papers/DQIssuesinIntegratedEnterpriseSystems.pdf [retrieved: 2021-04-24]

[6] L. Haas, "Beauty and the Beast: The Theory and Practice of Information Integration", In T. Schwentick and D. Suciu (Eds.): ICDT 2007, pp. 28–43, Springer-Verlag, Berlin Heidelberg, 2007, DOI: 10.1007/11965893_3

[7] P. Bernstein and S. Melnik, "Model Management 2.0: Manipulating Richer Mappings", Proceedings of the ACM SIGMOD International Conference on Management, pp. 1–12, 2007, DOI: 10.1145/1247480.1247482

[8] M. Crowe, C. Begg, F. Laux, and M. Laiho, "Data Validation for Big Live Data", DBKDA 2017, pp. 30–36, ISBN: 978-1-61208-558-6.

[9] F. Laux, "The Typed Graph Model", DBKDA 2020, pp. 13–19, ISBN: 978-1-61208-790-0.

[10] A. Doan, A. Halevy, and Z. Ives, Principles of Data Integration, Morgan Kaufmann, Elsevier, 2012, ISBN: 978-0-12-416044-6.

[11] M. Lenzerini, "Data Integration: A Theoretical Perspective", Proceedings of the 21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS) , pp. 233-246, 2002

[12] R. Fagin and Ph. Kolaitis, "Local transformations and conjunctive-query equivalence", PODS 2012, pp. 179–190, DOI: 10.1145/2213556.2213583

[13] A. Halevy, N. Ashish, D. Bitton, M. Carey, D. Draper, J. Pollock, A. Rosenthal, and V. Sikka, "Enterprise information integration: successes, challenges and controversies" SIGMOD Conference, pp. 778–787, 2005

[14] D. Maluf and P. Tran, "Netmark: A schema-less extension for relational databases for managing semi-structured data dynamically", ISMIS 2003, pp. 231–241, 2003

[15] A. Sheth and J. Larson, "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases", ACM Computing Survey 22(3), pp. 183–236, 1990

[16] M. T. Özsu and P. Valduriez, *Principles of Distributed Database Systems*, Fourth Edition, Springer Nature Switzerland AG, 2020, ISBN: 978-3-030-26252-5

[17] U. Leser and F. Nauman, *Informationsintegration - Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen* (Eng. Information integration - Architectures and methods for integration of distributed and heterogeneous data sources), dpunkt.verlag, 2006

[18] M. Friedman, A. Levy, and T. Millstein, "Navigational Plans for Data Integration", AAAI/IAAI, pp. 67–73, AAAI Press, 1999

[19] D. Roman et al., "The Linked Data AppStore - A Software-as-a-Service Platform Prototype for Data Integration on the Web", Mining Intelligence and Knowledge Exploration (MIKE); pp. 382–396; 2014

[20] L. Popa, Y. Velegrakis, R. Miller, M. Hernández, and R. Fagin, "Translating Web Data", pp. 598–609, VLDB, 2002

[21] A. Langegger, "Virtual Data Integration on the Web: Novel Methods for Accessing Heterogeneous and Distributed Data with Rich Semantics", iiWAS 2008, pp. 559–562, DOI: 10.1145/1497308.1497410,

[22] A. Adamou and M. d'Aquin, "Relaxing Global-As-View in mediated data integration from Linked Data", In: *Proceedings of The International Workshop on Semantic Big Data* (SBD 2020), Article No.: 4, pp. 1-6, DOI: 10.1145/3391274.3393635

[23] M. Junghanns, A. Petermann, K. Gómez, and E. Rahm, "GRADOOP: Scalable Graph Data Management and Analytics with Hadoop", Computing Research Repository (CoRR), vol. abs/1506.00548, no page numbers, 2015, [online] URL: https://arxiv.org/abs/1506.00548 [retrieved: 2021-04-24]

[24] G. Malewicz et al., "Pregel: a system for large-scale graph processing", SIGMOD Conference 2010, pp. 135–146

[25] M. Kricke, E. Peukert, and E. Rahm, "Graph Data Transformations in Gradoop". In T. Grust et al. (eds.) BTW 2019, pp.193–202, DOI: 10.18420/btw2019-12.

[26] V. de Sousa and L. del Val Cura, "Logical Design of Graph Databases from an Entity-Relationship Conceptual Model" iiWAS 2018, pp. 183–189

[27] I. A. Gelman, "A Theory of Complementarity for Extracting Accurate Data from Inaccurate Sources through Integration", no page numbers, ICIQ 2005

[28] L. Bertossi and L. Bravo, "Consistent Query Answers in Virtual Data Integration Systems", Inconsistency Tolerance 2005, pp. 42–83

[29] R. Hull, "Managing Semantic Heterogeneity in Databases: A Theoretical Perspective", PODS '97, pp. 51–61, ACM Press, 1997, DOI: 10.1145/263661.263668

[30] P. Atzeni and R. Torlone, "Management of Multiple Models in an Extensible Database Design Tool", EDBT 1996, pp. 79–95

[31] P. Atzeni, P. Cappellari, R. Torlone, Ph. Bernstein, and G. Gianforme, "Model-independent schema translation", VLDB Journal 17(6), pp. 1347–1370, 2008

[32] F. Laux, "The Typed Graph Model – a Meta-Language for Model Management and Data Integration", International Journal On Advances in Software, vol. 14 no 1&2, 2021, ISSN: 1942-2628, unpublished

[33] R. De Virgilio, A. Maccioni, and R. Torlone, "Converting Relational to Graph Databases", First International Workshop on Graph Data Management Experiences and Systems (GRADES), pp. 1–6, CWI/ACM, 2013, DOI: 10.1145/2484425.2484426

 ISBN: 978-1-61208-857-0

[34] R. Stoica, G. Fletcher, and J. Sequeda, “On Directly Mapping Relational Databases to Property Graphs”, Proceedings of the 13th AMW 2019, no page numbers, [online] URL: http://ceur-ws.org/Vol-2369/short06.pdf [retrieved: 2021-04-30]

[35] A. Petermann, M. Junghanns, R. Mller, and E. Rahm, “Graph-based Data Integration and Business Intelligence with BIIIG”, Proc. of the VLDB Endow, Vol. 7 no 13, pp. 1577–1580, 2014, DOI: 10.14778/2733004.2733034

[36] P. Buneman, S. Davidson, M. Fernandez, and D. Suciu, “Adding Structure to Unstructured Data”, 6th International Conference on Database Theory - ICDT, pp. 336–350, 1997

[37] A. Bilke and F. Naumann, “Schema Matching using Duplicates”, Proceedings of the 21st International Conference on Data Engineering (ICDE), pp. 69–80, 2005

[38] P. Hall, “On representation of subsets”, Journal of the London Math. Society, Vol. s1-10, Issue 1, pp. 26–30, 1935

[39] L. Ehrlinger and W. Wöß, “Automated Schema Quality Measurement in Large-Scale Information Systems”, in: H. Hacid, Q. Sheng, T. Yoshida, A. Sarkheyli, and R. Zhou (eds) Data Quality and Trust in Big Data (QUAT 2018), Lecture Notes in Computer Science, vol 11235, pp. 16–31, Springer, 2018, [online] URL: https://doi.org/10.1007/978-3-030-19143-6_2 [retrieved: 2021-05-02]

[40] Ch. Pinkel et al., “IncMap: A Journey towards Ontology-based Data Integration”, in Mitschang et al. (eds.) BTW 2017, pp. 145-164, [online] URL: https://dl.gi.de/20.500.12116/625 paper10.pdf [retrieved: 2021-04-24].

 ISBN: 978-1-61208-857-0