Data Vault Mappings to Dimensional Model Using Schema Matching

Mikko Puonti[1,2(✉)] and Timo Raitalaakso[1,2]

[1] Solita Ltd., Åkerlundinkatu 11, 33100 Tampere, Finland
{puonti, timo.raitalaakso}@iki.fi
https://www.solita.fi/en/
[2] Tampere University, Kalevantie 4, 33100 Tampere, Finland

Abstract. In data warehousing, business driven development defines data requirements to fulfill reporting needs. A data warehouse stores current and historical data in one single place. Data warehouse architecture consists of several layers and each has its own purpose. A staging layer is a data storage area to assists data loadings, a data vault modelled layer is the persistent storage that integrates data and stores the history, whereas publish layer presents data using a vocabulary that is familiar to the information users. By following the process which is driven by business requirements and starts with publish layer structure, this creates a situation where manual work requires a specialist, who knows the data vault model. Our goal is to reduce the number of entities that can be selected in a transformation so that the individual developer does not need to know the whole solution, but can focus on a subset of entities (partial schema). In this paper, we present two different schema matchers, one based on attribute names, and another based on data flow mapping information. Schema matching based on data flow mappings is a novel addition to current schema matching literature. Through the example of Northwind, we show how these two different matchers affect the formation of a partial schema for transformation source entities. Based on our experiment with Northwind we conclude that combining schema matching algorithms produces correct entities in the partial schema.

Keywords: Schema matching · Data flow · Data warehouse · Data vault · Dimensional model

1 Introduction

In a data warehouse, whereas several data sources are integrated as one data set, mapping information is crucial. Most commonly used in data warehouse implementation is Extract-Transform and Load (ETL) process [6].

Business driven development defines data requirements to fulfill reporting needs. These reporting needs are typically modelled with a dimensional modeling [7] technique. To enable parallel work with data transformation (ETL) creation and reporting tools we have created a dimensional model as a prerequisite for actual implementation [11]. Reporting tools need a dimensional model populated with a sample data set. Populating a sample data set to a dimensional model creates data flow mapping

© IFIP International Federation for Information Processing 2019
Published by Springer Nature Switzerland AG 2019
P. Doucek et al. (Eds.): CONFENIS 2019, LNBIP 375, pp. 55–64, 2019.
https://doi.org/10.1007/978-3-030-37632-1_5

information at attribute level, whereas one or many attributes are mapped to one target attribute. As a new interface is introduced to a data warehouse, it is created first to staging layer. A sample data set is mapped from staging layer to the publish layer (Fig. 1B). Mapping may be implemented as a database view or a ETL-transformation that populates tables. In this paper, we are referring to these views and tables as entities.

A data warehouse stores current and historical data in one single place. A data vault model [8] is used for storing history. When the data vault model exists, the transformations implementation between staging layer and data vault is automated by using the process presented in our earlier research [12].

By following the process which is driven by business requirements and start with designing a dimensional model with data, continuing with transformation implementation from a staging layer to a data vault model. The transformation graph forms a data flow. This creates a situation where we have in a place publish layer structure, data vault model and transformation mapping between a staging layer and a publish layer together with a data vault (Fig. 1C). Transformation mapping between a data vault and a publish layer is missing. When producing a durable implementation, a target is to create transformation to the publish layer based on the data vault model. Currently, this is manual work and it requires a specialist who knows the data vault model. A large data warehouse may consist of several hundred entities. Our research is focused on how to help this transformation mapping creation. How may we use the data flow mapping information, which is generated in earlier phases? Is there any particular schema mapping technique useful to solve this manual work and at least partially automate tasks in the current situation? Our goal is easing up the developers work. This is accomplished with finding candidates to be used in schema mappings. We are also finding ways to prune parts of a big data model to be used.

2 Related Work

Ontology matching is a wider research area than our schema matching. We are using ontology matching a name based technique where strings are identical [4]. Our ontologies are database schemas, even when the actual implementation not include a database schemas there are structures where tables (relation) contain attributes.

We are using existing data flow mapping information to generate new replacing transformation mappings together with more commonly researched schema matching methods. The data flow forms an directed acyclic graph. It may be considered form a computer program. It describes the dependencies between all entities in a system. Frank Tip is writing about using program slicing in program integration [14, Chap. 5.2]. We are using such slicing to generate subgraphs assisting transformation generation.

Villányi describes schema matching techniques in service-oriented enterprise application integration in his dissertation [15]. In a hybrid matcher Villányi combine a vocabulary matcher and a structural matcher, where structural matcher uses a neighborhood level structural similarity.

Atzeni et al. introduce meta-mappings as a formalism that describes transformations between generic data structures [2]. This enables mapping reuse, when similar

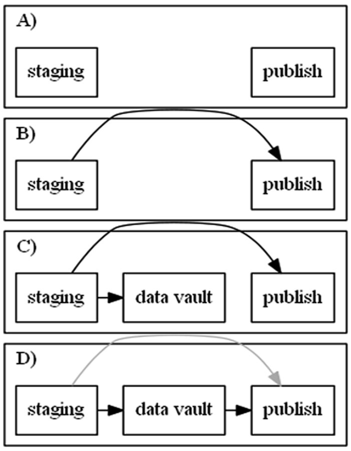

Fig. 1. Transformation stack evolution

information is located in several schemas, whereas our reuse is use data flow mapping for creating new transformation between schemas where transformation is not yet defined.

Golfarelli et al. [5] introduce a starry vault approach to generate a dimensional model automatically from a data vault model. In their paper there are formal definitions of data vault and multi dimensional schemas introduced. Their paper is aiming to find a multi dimensional model from data vault structures. Our approach is to match and generate data flow between two predefined models.

Human effort is needed in schema matching scenarios as existing matching algorithm results are not perfect. Nguyen et al. concentrate on minimizing human effort in reconciling match networks [10]. They stress that after matching there is still a need for a post-matching phase, which is manual correction. Their reconciliation process is an iterative process, whereas our solution is to offer a partition schema for transformation as a selection. Many authors agree that mapping can not totally automate, there is a need for manual corrections [1, 2, 5, 10, 15].

3 Schema Matching and Experiments

In a data warehouse data is in relational form [3], even when NoSQL techniques are used in implementation. A relational database consists of tables and attributes.

A set of tables is grouped together with a schema. Schema is used as an implementation of data warehouse layers, each layer in Fig. 1 is a separate schema. Now we can refine our research question “how to help transformation mapping generation” to form a match schema between a data vault and a publish layer schema.

3.1 Matching Workflow

In a matching workflow, we are using phases introduced Rahm [13]. First phase is preprocessing, where metadata information is extracted from a relational database (Fig. 2a). Relational schema offer metadata: a table name, an attribute name, the attribute data type, additional information for data type and a description field for attribute.

Matching is an execution of a matching algorithm, whereas it can contain several matching steps. These matching can be sequential, parallel or mixing both of those principals.

A combination of matcher results is combining different matcher algorithm values and possibly calculation aggregated value of those values. In a sequential matching a matcher can use earlier matcher values in an algorithm.

A selection of correspondences is in our case a human work phase, which we aim to help with offering matching results in use. If there is a tool built based on our article, it could suggest good matches and human work would be only accept suggestions and creating more complex mappings.

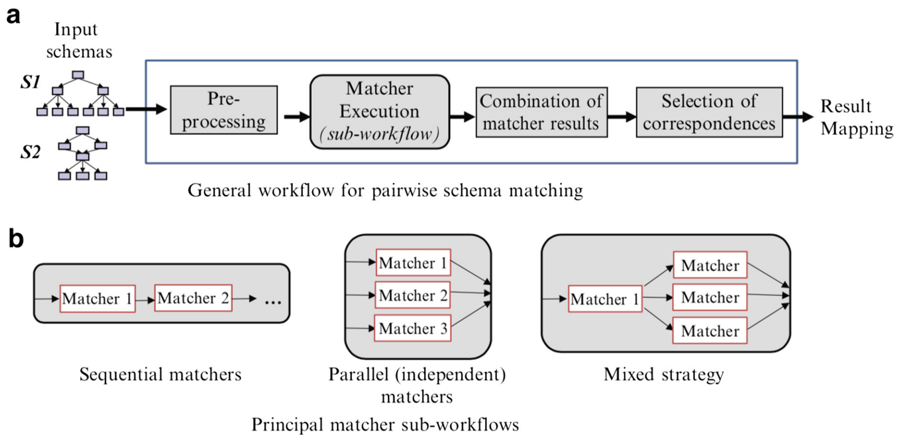
a
Input schemas
S1
S2
Pre-processing
Matcher Execution (sub-workflow)
Combination of matcher results
Selection of correspondences
Result Mapping
General workflow for pairwise schema matching
b
Matcher 1
Matcher 2
...
Matcher 1
Matcher 2
Matcher 3
Matcher 1
Matcher
Matcher
Matcher
Sequential matchers
Parallel (independent) matchers
Mixed strategy
Principal matcher sub-workflows

Fig. 2. General match workflow (copied from [13])

3.2 Schema Matching Based on Attribute Names

A matcher compares every attribute name of a data vault with every attribute in a publish layer target attribute. This cross join operation can be easily quite large and this is reason why we suggest to do this few publish layer entities at a time. The preprocess phase in Fig. 3 target subset is chosen. In matcher first pruning is to feed only a partial entity set from the publish layer entities, this may be interpreted as a partition of a second schema [13].

A result of the attribute name matcher is a set of data vault entities, which has common attribute naming compared between data vault and target entities.

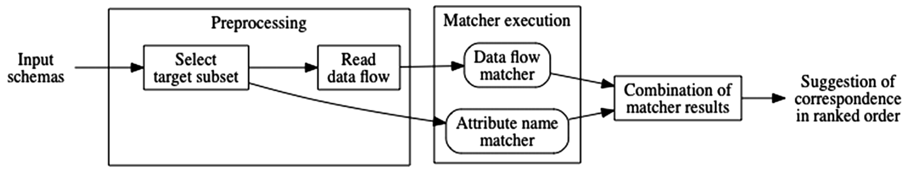

Fig. 3. Match workflow

3.3 Schema Matching Based on Data Flow Mapping

Publish layer entities have data flow mapping for a sample data set. This mapping can be implemented from the data vault or the staging area, nevertheless there is data flow mapping information as presented in Fig. 1B.

Data flow mappings are expressed at attribute level between source and target schemas. Depending on technology used it may be a challenging process to extract that attribute level mapping information, or even worse manually create this mapping information. We suggest expressing data flow information between source and target schema entities, this information is useful and it is easier to extract from ETL-tool or database view metadata information.

The target publish layer entities are chosen as an end point of the subgraph slice. A result of data flow mapping is data vault (source schema) entities, which have corresponding data flow to a publish layer (target schema) entities.

Inside data vault there might exist layers. A raw data vault that is populated straight from staging area. Business data vault [9] is a layer that enriches the data vault model and uses other data vault entities as a source. This piles up the transformation stack and makes the data flow graph deeper. We are using this depth as an indicator of enriched information. Giving a better ranking for business vault entities to be used in the suggested mappings.

3.4 Schema Matching Combination of Attribute Names and Data Flow Mapping

Last phase of our algorithm is a combination of earlier matchers results. Noteworth, these matchers have result sets at different level of granularity. This is presented in Fig. 3 combination of matcher results.

As we are aiming to match schemas between the data vault and the publish layer, this algorithm is for helping creating transformations between these schemas. For human decision, we are presenting potential entities for transformation creation. The earlier matchers enable us to use the following strategy:

- Present all potential entities in order where first is the most prominent candidate
- Present only potential entities, which are common in both matcher result set.

As the data flow information is not always available, our suggested strategy is to present all potential entities in relevant order.

The algorithm to create this ordering for candidate entities.

1. Count at entity level how many attributes is in an attribute name matcher result set.
2. Add a depth value for each entity, which is in a data flow matcher result set.
3. Summarise these result sets.
4. Order the result set according to the value of each entity.

3.5 Northwind Example

As a demonstration of our approach, we use Northwind[1] source data model. We modeled a publish layer schema of order fact and dimensions. The ORDER_F references to CUSTOMER_D, EMPLOYEE_D, ORDER_D and SHIPPER_D. After data vault modeling and transformation population at the phase (Fig. 1C) we get suggestions to new (Fig. 1D) phase transformations as described in (Tables 1 and 2).

CUSTOMER D gets side different false positives from both suggestions but CUSTOMER_H and CUSTOMER_S are found in both sets to be considered as source for mappings. The number of common columns in naming suggestion is higher for these and gets prioritized based on the naming match. Adding the second suggestion set from data flows the results become more convincing. SHIPPER_D gets assurance that CUSTOMER_H and CUSTOMER_S should not be considered as source for mapping. ORDER_F gets suggestions from either ORDER_L or ORDER_BV_L. Data vault model is layered. ORDER_L represent a raw data vault layer and ORDER_BV_L is an entity of business data vault layer. ORDER_BV_L uses another data vault entities as a source for it data. This dependency graph depth is visible in (Table 2) ORDER_BV_L - ORDER_F suggestion row. It is used together with higher score from naming suggestion to choose correct mappings to be implemented.

3.6 Observations from Northwind Example

Both our matchers return side hits - false positives. Attribute naming return overlapping from irrelevant similarity matches. The data flow matcher raises other potential mapping candidates. As the data flow subgraph from used staging entities contains transformations to other data vault entities that are not needed in the desired resulting mapping for a specific target publish layer entity.

With combing results from both approaches we get more precise suggestion for the new data vault publish layer transformations. With our experiments, the false positive groups are some what differing. So exclusion of false positive mapping candidates becomes more convincing. This minimizes the needed human effort while creating the end results.

[1] https://github.com/dshifflet/NorthwindOracle_DDL.

Table 1. Transformation suggestions based on naming

MAINENTITY	SOURCEENTITY	TARGETENTITY	C
CUSTOMER_H	CUSTOMER_H	CUSTOMER_D	2
CUSTOMER_H	CUSTOMER_S	CUSTOMER_D	2
SHIPPER_H	SHIPPER_H	CUSTOMER_D	1
SHIPPER_H	SHIPPER_S	CUSTOMER_D	1
EMPLOYEE_H	EMPLOYEE_H	EMPLOYEE_D	2
EMPLOYEE_H	EMPLOYEE_S	EMPLOYEE_D	2
ORDER_H	ORDER_H	EMPLOYEE_D	1
ORDER_H	ORDER_S	EMPLOYEE_D	1
ORDER_H	ORDER_H	ORDER_D	2
ORDER_H	ORDER_S	ORDER_D	2
CUSTOMER_ID_CUSTOMER_L	CUSTOMER_H	ORDER_F	1
CUSTOMER_ID_CUSTOMER_L	CUSTOMER_ID_CUSTOMER_L	ORDER_F	1
CUSTOMER_ID_CUSTOMER_L	CUSTOMER_ID_H	ORDER_F	1
ORDER_BV_L	CUSTOMER_H	ORDER_F	4
ORDER_BV_L	EMPLOYEE_H	ORDER_F	4
ORDER_BV_L	ORDER_BV_L	ORDER_F	4
ORDER_BV_L	ORDER_H	ORDER_F	4
ORDER_BV_L	SHIPPER_H	ORDER_F	4
ORDER_L	CUSTOMER_ID_H	ORDER_F	3
ORDER_L	EMPLOYEE_H	ORDER_F	3
ORDER_L	ORDER_H	ORDER_F	3
ORDER_L	ORDER_L	ORDER_F	3
ORDER_L	SHIPPER_H	ORDER_F	3
CUSTOMER_H	CUSTOMER_H	SHIPPER_D	1
CUSTOMER_H	CUSTOMER_S	SHIPPER_D	1
SHIPPER_H	SHIPPER_H	SHIPPER_D	2
SHIPPER_H	SHIPPER_S	SHIPPER_D	2

4 Discussion and Future Work

In our experimentation, we created a target schema as a database views from the staging layer. The data flow based matcher used this information at an entity level. There are possibilities to extract this data flow mapping information at an attribute level, one option is to use a tool like Queryscope[2]. This would open possibility to create more fine tuned result of the combined matcher described in this paper.

In this paper, we introduce two schema matcher. By adding more schema matchers, it is possible to improve the suggestions. A structural matcher might be beneficial. Link and fact granularities might be used. Link granularity, calculated based on the number of hubs it references, and fact cardinality, how many dimensions it references, could be compared.

[2] https://app.sqldep.com/demo/.

Table 2. Transformation suggestions based on data flows

SOURCEENTITY	TARGETENTITY	SCORE
CUSTOMER_H	CUSTOMER_D	1
CUSTOMER_ID_CUSTOMER_L	CUSTOMER_D	1
CUSTOMER_ID_H	CUSTOMER_D	1
CUSTOMER_S	CUSTOMER_D	1
ORDER_BV_L	CUSTOMER_D	2
EMPLOYEE_H	EMPLOYEE_D	1
EMPLOYEE_S	EMPLOYEE_D	1
CUSTOMER_ID_H	ORDER_D	1
EMPLOYEE_H	ORDER_D	1
ORDER_BV_L	ORDER_D	2
ORDER_H	ORDER_D	1
ORDER_L	ORDER_D	1
ORDER_S	ORDER_D	1
SHIPPER_H	ORDER_D	1
CUSTOMER_ID_H	ORDER_F	1
EMPLOYEE_H	ORDER_F	1
ORDER_BV_L	ORDER_F	2
ORDER_H	ORDER_F	1
ORDER_L	ORDER_F	1
ORDER_S	ORDER_F	1
SHIPPER_H	ORDER_F	1
SHIPPER_H	SHIPPER_D	1
SHIPPER_S	SHIPPER_D	1

Future work would be to suggest transformations between source schema (data vault) and target schema (publish layer) entities. At least the result set from the attribute name matcher is re-usable for creating transformation where is one-to-one mapping between source and target attribute. Our target is to reducing manual work by offering a subset of source schema entities for creating transformations, not actually create that data flow.

This paper is talking about a process of creating new or extending an existing data warehouse. A similar approach may be used when replacing an existing direct star schema publish layer data flows by adding data vault modeled enterprise data warehouse layer between a staging and a publish layer. Replacing old ETL tool implementation. Benefits of data vault methodology such as history in satellites and better agile development enablement. The process described in (Fig. 1) fits as is also on such replacement process. Phase (A) Use existing staging and star schema model. (B) Reverse engineer data flow transformation dependencies from old ETL implementation. (C) and (D) phases as described in this paper.

It is inevitable that the data vault model does not have a perfect match for source mapping at some point in data warehouse evolution. These pruned partial matcher results could be used to be a base for new business vault entities. The knowledge of a developer and an access path suggestor[3] could be used to generate links and bridges that satisfy publish layer source information needs.

5 Conclusion

Our research is focused on helping transformation creation between a data vault and a publish layer. For each publish layer entities we create a set of source entity candidates from a data vault schema entities.

As we are following the process which is driven by business requirements, there is a publish layer entity populated with a sample data. This sample data population construct a data flow to the target schema entity.

We examine whether schema matching is based on attribute names enough to suggest correct entities from a data vault schema. Schema matching based on attribute names finds correct entities from a source schema, but still there is room for improvement.

By adding a schema matcher based on data flow mapping we get a result set to enrich an attribute name based matching. Combining the results from these two matchers we may present potential transformation source entity candidates in order where the most prominent one is at first.

This combined algorithm is suitable for building a tool to help transformation mapping creation between a data vault and a publish layer. The result set from algorithm offer correct source entities for transformation mapping sources.

After choosing source entities for transformation mapping, we could additionally suggest mappings from schema matching based on attribute names as a base for that transformation where a specialist could continue with additional mappings and with the complex mappings.

References

1. Alexe, B., Ten Cate, B., Kolaitis, P.G., Tan, W.C.: Designing and refining schema mappings via data examples. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, pp. 133–144. ACM (2011)
2. Atzeni, P., Bellomarini, L., Papotti, P., Torlone, R.: Meta-mappings for schema mapping reuse. Proc. VLDB Endow. **12**(5), 557–569 (2019)
3. Codd, E.F.: A relational model of data for large shared data banks. Commun. ACM **13**(6), 377–387 (1970)
4. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38721-0
5. Golfarelli, M., Graziani, S., Rizzi, S.: Starry vault: automating multidimensional modeling from data vaults. In: Pokorný, J., Ivanović, M., Thalheim, B., Šaloun, P. (eds.) ADBIS 2016. LNCS, vol. 9809, pp. 137–151. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44039-2_10

[3] http://rafudb.blogspot.com/2019/01/access-path-suggestor.html.

6. Kimball, R., Caserta, J.: The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming and Delivering Data. Wiley, Hoboken (2004)
7. Kimball, R., Ross, M.: The Data Warehouse Toolkit: the Complete Guide to Dimensional Modeling. Wiley, Hoboken (2011)
8. Linstedt, D.: Data Vault Series 1–Data Vault Overview. The Data Administration Newsletter, Baltimore (2002)
9. Linstedt, D., Graziano, K., Hultgren, H.: The new business supermodel, the business of data vault modeling. Lulu.com (2008)
10. Quoc Viet Nguyen, H., et al.: Minimizing human effort in reconciling match networks. In: Ng, W., Storey, V.C., Trujillo, J.C. (eds.) ER 2013. LNCS, vol. 8217, pp. 212–226. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41924-9_19
11. Puonti, M., Lehtonen, T., Luoto, A., Aaltonen, T., Aho, T.: Towards agile enterprise data warehousing. In: ICSEA 2016, p. 241 (2016)
12. Puonti, M., Raitalaakso, T., Aho, T., Mikkonen, T.: Automating transformations in data vault data warehouse loads. In: EJC, pp. 215–230 (2016)
13. Rahm, E.: Towards large-scale schema and ontology matching. In: Bellahsene, Z., Bonifati, A., Rahm, E. (eds.) Schema Matching and Mapping, pp. 3–27. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-16518-4_1
14. Tip, F.: A survey of program slicing techniques. J. Program. Lang. **3**, 121–189 (1995)
15. Villányi, B.J.: Schema matching techniques in service-oriented enterprise application integration. Informatikai Tudományok Doktori Iskola (2016)