Towards Automated Schema Optimization*

André Conrad[1], Sebastian Gärtner[2], and Uta Störl[1][0000−0003−2771−142X]

[1] University of Hagen, Germany
andre.conrad|uta.stoerl@fernuni-hagen.de
[2] Accso Accelerated Solutions GmbH, Darmstadt
sebastian.gaertner@accso.de

Abstract. Non-relational systems are essential to manage large amounts of semi- and/or unstructured data. To use the optimal data storage at a given time, it may be necessary to change the data model during the lifetime of an application. This paper offers a visionary approach providing an automated schema migration and optimization between different NoSQL data stores. By means of data and query analyses, optimizations of all existing cardinalities can be achieved with respect to good query performances with minimal redundancy. First performance measurements prove the increase in performance.

Keywords: Schema Migration · Schema Optimization · Data Migration

1 Introduction

Due to the steady development and appearance of new database technologies as well as the further development of existing applications and the consequential changed requirements to a database system used at a given time, it can be necessary to migrate data between different, heterogeneous database systems. If the data of the source system are migrated without optimizing the schema with respect to the target system, performance may be affected.

For example, missing or restricted join operations in NoSQL systems lead to performance losses. Otherwise changes in the database schema (e.g. embedding) can help to improve the performance.

This paper introduces the concept of a flexible migration architecture between different database systems with various data models. Our approach includes an automated optimization process based on data metrics and query analysis that allows the identification of the best way to embed or reference the data.

At first a migration from relational to document stores is presented. This approach uses an automatic optimization process where the transformation rules can be modified and extended. The contributions of the paper are:

* This work has been funded by Deutsche Forschungsgemeinschaft (German Research Foundation) - 385808805

- A comparison of existing approaches to migrate schema and data between different database models.
- The proof of concept of a flexible migration architecture with an automated optimization of the schema transformation.
- First performance analyses to investigate the increase in performance through optimization.

This paper is organized as follows: Section 2 presents related work on migration and optimization processes between different database systems. Section 3 describes first approaches to a concept on the way to an automated and flexible migration process. Section 4 shows performance improvements based on first measurements regarding a migration from the relational database system MariaDB to the document-oriented system of MongoDB. Finally, Section 5 provides the conclusions as well as an outlook on our further work.

2 Related Work

This chapter presents the state of the art on migration and optimization processes between different database systems. After a short introduction of the different approaches a summarizing comparison is given at the end of the chapter (cf. Table 1).

[10] describes an algorithm that outputs the order in which related tables can be aggregated into one large "NoSQL-table" with regard to the optimization of read queries in a migration from relational to document stores. [1] presents a process that takes conceptual UML class diagrams and transforms them into physical models of different NoSQL stores using transformation rules and a platform independent meta model. [2] describes a semi-autonomous rule-based process by which a conceptional UML class diagram can be transformed into various platform specific models. The class diagram can be edited manually in order to divide it into multiple regions which describe several physically separated models. [6] shows a concept of data and schema migration from relational to document and column-oriented NoSQL stores that has flexible optimization possibilities. The core is a platform independent meta model that additionally describes data and query characteristics of the source system. Different strategies are provided to optimize the model with respect to different target systems. [3] describes the migration of data from relational systems to the document-oriented model of MongoDB in detail. This includes a semi-autonomous optimization process against the target model. Note that many-to-many relationships are always referenced. In [8] an optimization process is described where several target models based on transformation heuristics are generated.

Finally, Table 1 summarizes the works considered and offers a comparison between different properties regarding automated data and schema migration including optimization processes.

[3] is one of the few works that offers a detailed description of the migration process including schema optimizations. However, they do not provide optimizations for many-to-many relationships. Therefore, our approach provides flexible

Table 1. Comparison of different approaches to schema and data migration between heterogeneous database systems (✓*: Not described in detail, (✓): not fully automated).

	[10]	[1]	[2]	[6]	[3]	[8]
Schema optimization	(✓)	(✓)	-	(✓)	(✓)	(✓)
Data/query analysis	-	-	-	✓*	✓	-
Meta model	-	✓	-	✓*	-	✓
SOURCE MODELS:						
Relational	✓	-	-	✓	✓	-
UML	-	✓	✓	-	-	✓
TARGET MODELS:						
Relational	-	-	✓	-	-	✓
Document	✓	✓	✓	✓	✓	✓
Graph	-	✓	✓	-	-	✓
Column	-	✓	-	✓	-	✓
Multi storage	-	-	✓	-	-	-

and extensible transformation rules regarding different target models. We perform an in-depth analysis of performance critical queries by using multiple quantitative metrics of existing data. *One-into-many* and *many-into-many* embeddings are also considered.

3 Concept

Our concept is based on the use of a platform independent meta model. We offer a flexible and extensible process for an automated schema and data migration between all kinds of data stores. A further important goal is the automated optimization of the schema with respect to the target system using different strategy rules as presented in [6] as well as the previously collected data metrics and analysis of the queries in the form of a so-called *dependency graph*. Fig. 1 shows an overview of the design described.

At first an automatic migration and schema optimization, from relational to document-oriented stores is considered. The optimization of the schema is achieved through de-normalization, i.e. embedding certain relationships. In order to determine the best direction of embedding among many-to-many relationships as well as the rule-based decision process whether to embed or reference, further data metrics must be considered, compared to [3]. Based on these metrics as well as the cardinalities of concerning relationships, the *dependency graph* is generated from the queries. This has the advantage that all previously analyzed queries can be involved in a later optimization step concerning target system specific rules.

To provide a high degree of flexibility and extensibility, the strategy rules can be described on two levels following the concept from [6]. On the one hand, those that are applicable to certain data models, such as the document model and – on the other hand rules – for specific systems like MongoDB.

Schema extraction and meta model: The step of extracting the physical data structure from the source database to a platform independent meta model is an important base for the migration architecture [4, 5, 8]. The description or

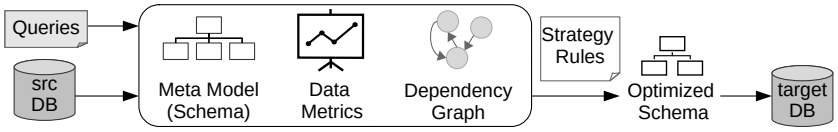

Fig. 1. Simplified overview of the migration and optimization process.

development of a suitable meta model would, however, go beyond the scope of this paper.

Data metrics and query analysis: An important point of the schema optimization are the metrics that are collected in the form of a data analysis, referred to the source database. The following metrics are used here:

- `avgEntitySize`: Average size of entities within an entity type.
- `entityCount`: Number of entities within an entity type.
- `rCount{Min,Max,Avg}`: Concerning the relationships between two entity types, the minimal, maximal and average number of "connections".

The goal of the analysis of queries is to generate the *dependency graph* to show the dependencies between the entity types involved in the queries. Algorithm 1 illustrates the process for generating this graph. Note that currently no queries can be optimized that have joins over more than two entity types while at the same time more than one entity type is affected by filter operations.

Algorithm 1: Generation of the *dependency graph*.

Input : Set of queries Q; Data metrics M; Schema S
Output: Set of digraphs G; dependency graph DG

for q_i **in** Q**:**
  $g_{query} \leftarrow convertToUndirectedGraph(q_i)$ `// entity types to vertices and joins to edges`
  $removeUnconnectedVertices(g_{query})$
  $v \leftarrow None$ `// start node`
  if *only one vertex in* g_{query}**:**
    $v \leftarrow$ vertex
  elif *only one vertex affected by filter operations*:
    $v \leftarrow$ affected_vertex
  elif g_{query} *has exactly two vertices*:
    if *one-to-many*:
      $v \leftarrow$ one-side_vertex
    elif *many-to-many*:
      $v \leftarrow$ higher_redundance_vertex `// calculated by GET_REDUNDANCE_SIZE()`
  if $v \neq None$:
    align all edges in g_{query} to v
    add q_{query} to G
merge all $g \in G$ into DG

def `GET_REDUNDANCE_SIZE`$(v1, v2)$**:**
  return $((M.entityCount(v_1) * M.rCountAvg(v_2)) - M.entityCount(v_1)) * M.avgEntitySize(v_1)$

Strategy rules: The structure of the physically target model is finally defined by rules related to specific target systems. These make use of the data metrics, user defined thresholds of those metrics, the cardinalities of the relationships, and the *dependency graph* to describe the transformations of the schema for an optimal performance. In the following the early approaches of transformation rules for a migration to the document-oriented model of MongoDB are shown:

- `R1 (doc):` $\forall$ relationships $r \in S : r \notin DG \rightarrow$ REF
- `R2 (doc):` $\forall$ relationships $r \in S : r \in DG \rightarrow$ EMBED
- `R3 (mongo):` $\forall$ edges $\in DG$: direction is *one-into-x* $\rightarrow$ EMBED
- `R4 (mongo):` $\forall$ edges $\in DG$: direction is *many-into-x* $\rightarrow$
  if $mCond$ = true EMBED else $\rightarrow$ REF
  with $mCond$: $(\text{rCountMax}(\text{e}_\text{j}) < \text{rCountMax}_\text{thold}) \wedge$
  $(\text{rCountMax}(\text{e}_\text{j}) * \text{avgEntitySize}(\text{e}_\text{j}) < \text{rCountMax}_\text{thold} * \text{avgEntitySize}_\text{thold})$

Note that the database system specific rules `R3` and `R4` are preferred to the general rule `R2`.

4 First Evaluations

First measurements were performed to prove an increase in performance with the example of a migration from MariaDB to MongoDB. Therefore, a part of the data model of the multi-model benchmark UniBench [9] and the three provided data sets[1] with different scale factors (*SF1, SF10* and *SF30*) were used. As a first example, optimizations have been made for following SQL query:

```
SELECT c.firstName, c.lastName, f.feedback FROM customer c, feedback f, product p
WHERE c.id = f.customerId AND f.asin = p.asin AND p.title = 'Katadyn TRK Drip Ceradyn...';
```

Fig. 2 shows the data model used as well as the "relationship structure" after the automatic optimization process using Algorithm 1 and the strategy rules regarding MongoDB (see Section 3). Fig. 3 shows that first approaches of a simple optimization process have already led to performance improvements compared to a non-optimized migration.

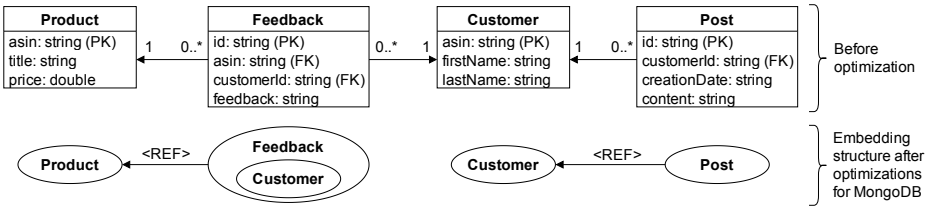

Fig. 2. The data model used and the optimized MongoDB "relationship structure" after the optimization process.

[1] `https://github.com/HY-UDBMS/UniBench/releases`

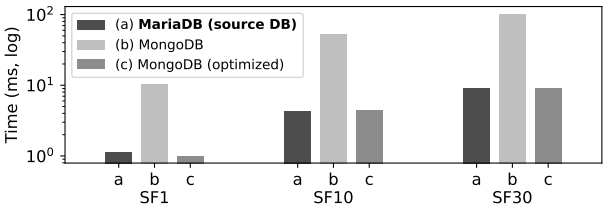

Fig. 3. Query performance with and without optimizations on MongoDB compared to the source database (Indices on all primary keys, foreign keys and *Product titles*).

5 Conclusion and Outlook

This paper presents a visionary approach to a flexible and extensible concept towards schema and data migration between different heterogeneous database systems, including automated optimization processes. We first describe strategy rules using an example for a migration from relational to document-oriented stores. In contrast to previous work like [3], the optimization process presented in this paper considers *one-into-many* and *many-into-many* embeddings.

The first measurements taken show a performance improvement in comparison to a non-optimized migration.

Our future work aims at the extension and in-depth evaluation of our automated schema optimization approach to all NoSQL data models, including multi-model systems [7].

References

1. Abdelhedi, F., et al.: UMLtoNoSQL: Automatic transformation of conceptual schema to NoSQL databases. In: AICCSA'17. IEEE (2017)
2. Daniel, G., Gómez, A., Cabot, J.: UMLto[no] SQL: Mapping conceptual schemas to heterogeneous datastores. In: RCIS'19. IEEE (2019)
3. Jia, T., Zhao, X., Wang, Z., Gong, D., Ding, G.: Model Transformation and Data Migration from Relational Database to MongoDB. In: Big Data'16. IEEE (2016)
4. Klettke, M., Störl, U., Scherzinger, S.: Schema Extraction and Structural Outlier Detection for JSON-based NoSQL Data Stores. In: BTW'15. GI (2015)
5. Klettke, M., Störl, U., Shenavai, M., Scherzinger, S.: NoSQL schema evolution and big data migration at scale. In: SCDM'16. IEEE (2016)
6. Liang, D., Lin, Y., Ding, G.: Mid-model Design Used in Model Transition and Data Migration between Relational Databases and NoSQL Databases. In: SmartCity'15. IEEE (2015)
7. Lu, J., Holubová, I.: Multi-model Databases: A New Journey to Handle the Variety of Data. ACM Comput. Surv. (2019)
8. Mali, J., Atigui, F., Azough, A., Travers, N.: ModelDrivenGuide: An Approach for Implementing NoSQL Schemas. In: DEXA'20. Springer (2020)
9. Zhang, C., Lu, J.: Holistic evaluation in multi-model databases benchmarking. Distributed and Parallel Databases (2019)
10. Zhao, G., Lin, Q., Li, L., Li, Z.: Schema Conversion Model of SQL Database to NoSQL. In: P2P, Parallel, Grid, Cloud and Internet Computing'14. IEEE (2014)