Data migration in column family database evolution using MDE

Pablo Suárez-Otero[1], Michael J. Mior[2], María José Suárez-Cabal[1] and Javier Tuya[1]

[1]University of Oviedo, Campus de Viesques, Gijón, Spain
[2]Rochester Institute of Technology, Golisano College of Computing and Information Sciences, Rochester, USA

Abstract

When software requirements change, databases used by an application may evolve, including the database models (conceptual model and schema), which may require data migrations to maintain data integrity. In some databases such as NoSQL column family databases, the tables of the database often store repeated data, as schema denormalization is encouraged to achieve the best performance. This makes data integrity maintenance more complex, as a single conceptual model change may trigger several changes in the schema. In this work, we propose using a model-driven engineering approach named MoDEvo for data migration in NoSQL column family databases to maintain data integrity after the schema evolves. Using a motivating example as a case study from an open source project that requires data migration, we describe MoDEvo and use it to illustrate how MoDEvo determined these data migrations.

Keywords

Model-Driven Engineering, Database evolution, Data Migration, NoSQL, Denormalization

1. Introduction

The requirements of a project determine how the database schema is designed, usually using a conceptual model during the design phase. This is especially important for some database types such as NoSQL column family databases (e.g. Apache Cassandra) where the schema is usually denormalized [1, 2]. A change in the requirements can require modifying both the conceptual model and the schema (tables in the database), which can create data integrity problems due to losing the synchronization between the conceptual model and the schema [3]. This problem is more difficult to solve in databases where data is duplicated among the tables and where the primary keys of the tables might not match the primary keys of the conceptual model [1]. This happens in the aforementioned NoSQL column family databases, where each table is designed so that a specific query can be executed against it, implying a denormalization of the schema as the same datum can be queried more than once and is therefore stored in several tables [4].

In NoSQL column family databases, the schema is more flexible than in a relational database and does not have integrity constraints, which makes mistakes during schema evolution more likely. A single change in one table must be replicated to every table storing the same data. This is relevant when schema evolution is required, as this evolution may jeopardize data integrity in the database [4], requiring additional data migrations to maintain this integrity.

In this work in progress, we address the maintenance of data integrity in a NoSQL column family database when a change of requirements causes an evolution of the schema. We propose using a MDE (model-driven engineering) approach named MoDEvo (Model (Driven) Data Evolution) to determine the data migrations required to maintain this integrity through models. In this work, we use a motivating example as a proof of concept to determine the feasibility of our proposal. The contributions of this work are 1) the determination of the data migrations required to maintain the data integrity in a use case and 2) the definition of MoDEvo to automate this determination.

The remainder of this paper is structured as follows. Section 2 contains a motivating example of a real open-source project where data migration was required. In Section 3 we describe of MoDEvo and its use for the motivating example. The related work is presented in Section 4. The paper ends in Section 5 with the conclusions and future work.

2. Motivating example

We use as a motivating example a schema change from the open-source project *Wireapp*[1] in which the column 'team' was added to the primary key of the table "issuer_idp". As it is not possible to directly alter the pri-

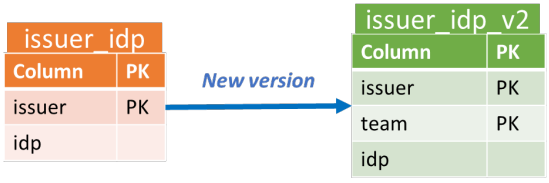
[1]https://github.com/wireapp/wire-server

Published in the Workshop Proceedings of the EDBT/ICDT 2023 Joint Conference (March 28-March 31, 2023, Ioannina, Greece)

suarezgpablo@uniovi.es (P. Suárez-Otero); mjmvcs@rit.edu (M. J. Mior); cabal@uniovi.es (M. J. Suárez-Cabal); tuya@uniovi.es (J. Tuya)
0000-0003-3282-5456 (P. Suárez-Otero); 0000-0002-4057-8726 (M. J. Mior); 0000-0001-8262-2871 (M. J. Suárez-Cabal); 0000-0002-1091-934X (J. Tuya)

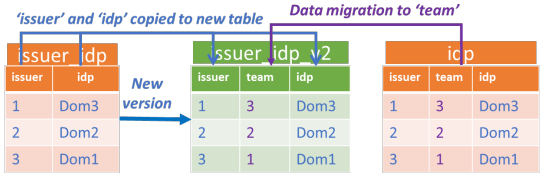
mary key of a table, a new table: "issuer_idp_v2" was created, which is illustrated in Figure 1.

Figure 1: Previous and new version of issuer_idp



To maintain data integrity, each row of "issuer_idp" must be migrated to "issuer_idp_v2", adding the appropriate value for column 'team' to complete each row. Values of column 'team' must be obtained from table "idp" as it is displayed in Figure 2.

Figure 2: Migration of data from column team from 'idp' to column 'team' from 'issuer_idp_v2'



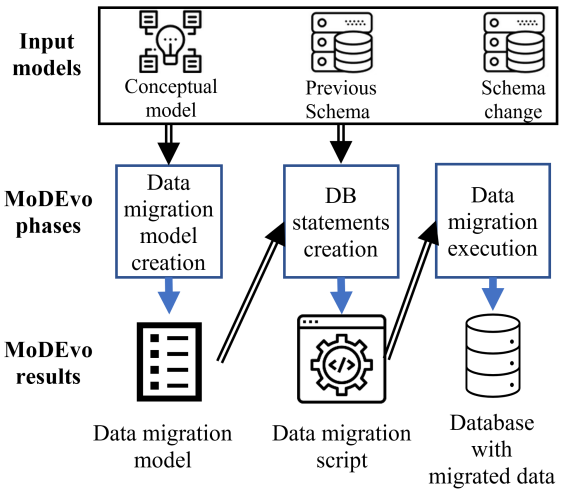
It is important to note, that this migration is still not being executed by the developers of the project. They have left the previous table to be queried for the data that was previously stored and a TODO task to migrate the data in the future. Maintaining the old table "issuer_idp" jeopardizes data integrity, as both tables are actually storing different data, although they are intended to store the same relationship between entities "issuer" and "idp". In the next section, we propose MoDEvo to avoid scenarios like this one.

3. MoDEvo description

In this section we propose MoDEvo, an MDE approach that determines the required migrations of data to maintain data integrity when the schema evolves. Figure 3 displays the MoDEvo process divided in three phases (middle row) alongside its inputs (top row) and results of each phase (bottom row), which are also inputs in the following phase.

In the first two phases, MoDEvo receives the information of the evolution of the schema through 3 input models: 1) the conceptual model, 2) the schema before the change and, 3) the schema change. In the first phase

Figure 3: MoDEvo process



(**Data migration model creation**), MoDEvo determines through model transformations implemented in ATL [5] the "Data migration model" (conforms to the metamodel illustrated in Figure 4), which contains the data migrations. In the second phase (**DB statement creation**), the "Data migration model" is transformed, through M2T (model to text) transformations, into a script that contains the database statements required to perform the data migrations. In the third phase (**Data migration execution**), MoDEvo sends this script to a migration engine (e.g. Apache Spark) to execute the script. We focus on the two first phases.

A Data Migration model (Figure 4) contains a MigrationTable element for each target table that requires data migrations in some of its columns. For each of these target columns, a MigrationCol element is defined, which determines this target column that receives data and the source column that provides the data.

The target and source columns are specified through references to Target and Source elements, which likewise contain the references 'Data' and 'Key' to schema column elements:

- 'Data': Indicates the **source** column from where data will be obtained and the **target** column where data will be inserted.
- 'Key': Indicates a functional dependency that the 'Data' column has with other columns, which makes it possible to determine the specific data to insert in each row of the 'Data' **target** column. MoDEvo determines a functional dependency if the following condition is met: Being a column Datac mapped to a non-key attribute $Data_a$ and a set of columns Key_c, mapped to a set of key

Figure 4: Data Migration metamodel

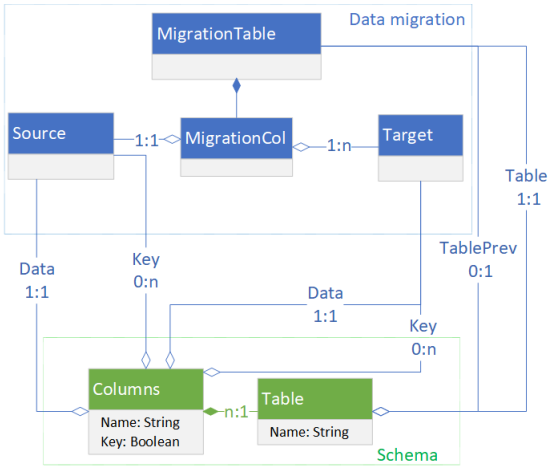
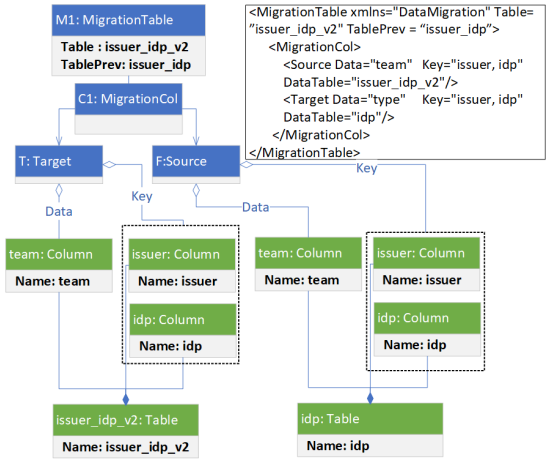


Figure 5: Model that specifies the data migration. Blue: Data Migration. Green: Schema



attributes Key_a, if $\mathbf{Key_a} \rightarrow \mathbf{Data_a}$ then $\mathbf{Key_c} \rightarrow \mathbf{Data_c}$.

When applying MoDEvo to the case study from the motivating example, it first analyzes the conceptual model, the schema and the schema change to create the migration model (displayed in Figure 5 graphically and textually). MoDEvo then sets 'team' as 'Data' column in both the Source table "issuer_idp_v2" and the Target table "idp". Likewise, columns 'issuer' and 'idp' are set as 'Key' columns, as they will be used to determine which value of 'team' stored in the source table is inserted in each row of the target table. Additionally, the element MigrationTable references the target table and its previous version through the attributes Table and TablePrev.

Using the generated model, MoDEvo creates through M2T transformation the data migration script. This script contains the database statements to: 1) obtain the data stored in each row of the previous table "issuer_idp", 2) associate these data with the appropriate value for the column 'team' and, 3) insert the data in the target table "issuer_idp_v2". Finally, this script will be sent to a migration engine. The resulting migration was illustrated in the previous section in Figure 2.

After the migration is performed, the previous table "issuer_idp" can be removed, avoiding the problems related to data integrity that were detailed in Section 2.

4. Related Work

Database migrations for column family databases have been approached in several work by Störl et al. [6, 7, 8, 9]. In two of these works [9, 7], the authors also addressed the maintenance of data integrity on document-oriented databases such as MongoDB by determining the required data migrations. Focusing on column family DBs, Störl et al have addressed other issues, such as optimizing the performance of these databases [8] as well as reducing the monetary cost of the infrastructures where they are deployed [6].

Data integrity in column family databases have been researched for several problems, such as analyzing how malicious attacks can affect data integrity [10]. We also addressed the maintenance of the data integrity when the data changes [4]. However there has not been any research that addressed the maintenance of data integrity when the schema evolves in column-family databases., which is what we address in this work.

We made a first approach for database evolution [11] by identifying the conceptual model changes that occurred in a set of open-source projects. We determined that after one of these changes three processes must be performed: 1) schema evolution, 2) data migrations to maintain data integrity and 3) update client application. In another work [3] we also considered the scenario where the schema changes and the conceptual model needs to be updated. However, in neither of these works [11, 3] we focused on the details of the aforementioned processes.

5. Conclusions

In this work, we propose MoDEvo, a MDE approach that provides the migrations required to maintain data integrity when a column family database schema changes. MoDEvo helps developers to avoid scenarios like the one presented in Section 2 where a deprecated table was kept

in the schema because their data was not migrated to the tables that are being used at the moment. Maintaining deprecated tables increases the risk of having issues regarding data integrity, as they are storing data that is not consistent with the rest of the database. MoDEvo reduces this risk by making legacy tables unnecessary.

MoDEvo also reduces the time that developers need to employ when evolving the database, as it automatically determines what needs to be migrated. Developers will have a script with the database statements that are required to perform all the data migrations needed to maintain the data integrity.

As future work, we plan to complete MoDEvo by developing its third phase and migrate data by sending the data migration script to a migration engine. We also plan to combine our previous work for schema evolution [3] with MoDEvo. With this combination, we will provide at the same time what needs to be changed in the schema and what data migration must be executed to maintain the data integrity for any requirement change that modifies the conceptual model.

Acknowledgments

This work was supported in part by projects [TIN2016-76956-C3-1-R] funded by the Spanish Ministry of Economy and Competitiveness, [PID2019-105455GB-C32] funded by MCIN/ AEI/10.13039/501100011033 and the Severo Ochoa pre-doctoral grant PA-21-PF-BP20-184.

References

[1] A. Chebotko, A. Kashlev, S. Lu, A big data modeling methodology for Apache Cassandra, in: 2015 IEEE International Congress on Big Data, IEEE, 2015, pp. 238–245. doi:10.1109/BigDataCongress.2015.41.

[2] M. J. Mior, K. Salem, A. Aboulnaga, R. Liu, Nose: Schema design for NoSQL applications, IEEE Transactions on Knowledge and Data Engineering 29 (2017) 2275–2289. doi:10.1109/TKDE.2017.2722412.

[3] P. Suárez-Otero, M. J. Mior, M. J. Suárez-Cabal, J. Tuya, An integrated approach for column-oriented database application evolution using conceptual models, in: International Conference on Conceptual Modeling, Springer, 2021, pp. 26–32. doi:js5t.

[4] M. J. Suárez-Cabal, P. Suárez-Otero, C. de la Riva, J. Tuya, MDICA: Maintenance of data integrity in column-oriented database applications, Computer Standards & Interfaces 83 (2023) 103642. doi:10.1016/j.csi.2022.103642.

[5] F. Jouault, F. Allilaire, J. Bézivin, I. Kurtev, ATL: A model transformation tool, Science of computer programming 72 (2008) 31–39. doi:10.1016/j.scico.2007.08.002.

[6] A. Hillenbrand, M. Levchenko, U. Störl, S. Scherzinger, M. Klettke, Migcast: putting a price tag on data model evolution in NoSQL data stores, in: Proceedings of the 2019 International Conference on Management of Data, 2019, pp. 1925–1928. doi:10.1145/3299869.3320223.

[7] A. Hillenbrand, U. Störl, S. Nabiyev, M. Klettke, Self-adapting data migration in the context of schema evolution in NoSQL databases, Distributed and Parallel Databases (2021) 1–21. doi:10/js5k.

[8] M. L. Möller, M. Klettke, U. Störl, Evobench–a framework for benchmarking schema evolution in NoSQL, in: 2020 IEEE International Conference on Big Data (Big Data), IEEE, 2020, pp. 1974–1984. doi:10.1109/BigData50022.2020.9378278.

[9] M. Klettke, U. Störl, M. Shenavai, S. Scherzinger, NoSQL schema evolution and big data migration at scale, in: 2016 IEEE International Conference on Big Data (Big Data), IEEE, 2016, pp. 2764–2774. doi:10.1109/BigData.2016.7840924.

[10] G. Weintraub, E. Gudes, Data integrity verification in column-oriented NoSQL databases, in: Data and Applications Security and Privacy XXXII: 32nd Annual IFIP WG 11.3 Conference, DBSec 2018, Bergamo, Italy, July 16–18, 2018, Proceedings 32, Springer, 2018, pp. 165–181. doi:10.1007/978-3-319-95729-6_11.

[11] P. Suárez-Otero, M. J. Mior, M. J. Suárez-Cabal, J. Tuya, Maintaining NoSQL database quality during conceptual model evolution, in: 2020 IEEE International Conference on Big Data (Big Data), IEEE, 2020, pp. 2043–2048. doi:10.1109/BigData50022.2020.9378228.