VESEL: VisuaL Exploration of Schema Evolution Using Provenance Queries

Christos Athinaiou
Computer Science Department,
University of Crete,
Heraklion, Greece
csd3258@csd.uoc.gr

Haridimos Kondylakis
Institute of Computer Science,
FORTH,
Heraklion, Greece
kondylak@ics.forth.gr

ABSTRACT

Database schemata are not static artifacts but subject to continuous change as new requirements daily occur and the modeling choices of the past should be updated or adapted. To this direction, multiple approaches available already try to keep multiple co-existing schema versions in parallel or model schema evolution through Schema Modification Operations, known as SMOs. However, to the best of our knowledge, in the era of big data, where thousands of SMOs might appear, it is really hard for developers to identify the modeling choices of the past and to explore how a specific column or table has been evolved. In this demo, we present VESEL, the first system enabling the VisuaL Exploration of Schema Evolution using *provenance* queries. Our approach relies on a state of the art database evolution language, and can efficiently answer queries about *when* a specific table or column has been introduced, *how* - with which SMO operation and *why* - which is the sequence of changes that led to the creation of the specific table/column. In the demonstration we will present the architecture of our system and the various algorithms implemented, enabling end-users to visually explore schema evolution. Then we will allow conference participants to interact directly with the system to test its capabilities.

1 INTRODUCTION

As recent databases do not proactively support schema evolution, developers have to migrate data from one database version to another using error-prone and complex Extract-Transform-Load scripts [1–3]. The problem has been recognized by multiple research works that try to offer tools for the uninterrupted evolution of either semantic [4] and relational schemata. More specifically for the relational world, the Model Management approach provides tools to match, diff, merge and extract mappings between schema versions [5]. PRISM and PRISM++ [6, 7] allow the developers to specify the evolution steps using Schema Modification Operations (SMOs). Those SMOs are then implemented for data migration and for rewriting past queries to work in the new versions. PRIMA [8] proposes SMOs that enable forward but not backward query rewriting and data migration, CoDEL [9] presents relationally complete SMOs whereas Symmetric Relational Lenses [10] define bidirectional SMOs not relationally complete. Finally, BiDEL [11] presents SMOs that are relationally complete, inversible and enable forward and backward query rewriting and data migration, whereas they can guarantee bidirectionality.

Independent of the database evolution language used, complex evolution scenarios can easily lead to hundreds of schema modification operations, e.g. the evolution of Wikipedia's schema through 171 schema versions [12], making it extremely difficult for developers to identify the modeling changes of the past and to be able to observe specific decision paths over this complex evolution.

In this demonstration, we present the VisuaL Exploration of Schema Evolution (VESEL) system, enabling developers to issue provenance queries, in order to actively explore the evolution of the schema. The database evolution language on top of which our system works is BiDEL, but the modular nature of the system makes it possible to be extended with other languages in the near future. It supports *how* (which schema modification operation), *when* (which version) and *why* (sequence of schema modification operation) provenance queries for a specific table or column and enables the graphical visualization of the results. As the selected database evolution language guarantees bidirectionality, queries can be issued using a recent or a past schema version and we can track the evolution of the specific table/column through multiple schema versions.

To the best of our knowledge, our approach is the only visual approach enabling active exploration of schema evolution. A preliminary work to the same direction is [13]. However it lacks a visual exploration interface, the corresponding algorithms are not presented, and it uses PRISM/PRISM++ SMOs not guaranteeing backward query rewriting and data migration. A similar approach but for ontologies has already shown the benefits of such an approach [14, 15].

The remaining of this paper is structured as follows: In section 2 we describe system's architecture, the various components used and the implemented algorithms. Then, in Section 3 we present an outline of our demonstration. Finally, Section 5 concludes this paper and presents directions for future work.

2 ARCHITECTURE

VESEL has been implemented using python and flask micro-framework [16], whereas for the visualization the Cytoscape library [17] has been used. VESEL employees a three layer architecture, shown in Figure 1.

On the top layer, we find the graphical user interface (GUI), in the middle the query engine and at the bottom the persistence layer. In the sequel we introduce the various components used and we describe their functionality.

2.1 Graphical User Interface

The whole process is started as soon as a file with BiDEL SMOs is uploaded to the system or an existing one is selected. The BiDEL database evolution language includes SMOs for creating a

Published in the Workshop Proceedings of the EDBT/ICDT 2019 Joint Conference (March 26, 2019, Lisbon, Portugal) on CEUR-WS.org.

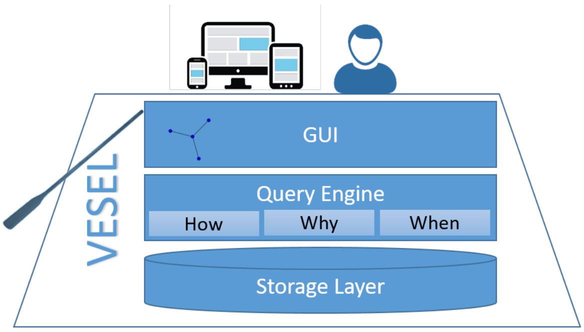


Figure 1: High-level system architecture.

table with specific fields, dropping and renaming tables, adding and updating columns and merging and decomposing tables. The complete list of the SMOs available are depicted in Table 1. We have to note that such files can be manually generated or generated by tools like the InVerDa[1] [18]. For more information on the SMOs and the corresponding rules that add and delete specific schema information, the interested reader is forwarded to [11].

Table 1: The SMOs of the BiDEL schema evolution language.

EVOLUTION START [FROM nameOld] EVOLUTION COMMIT AS nameNew;
CREATE TABLE $R(c_1,....,c_n)$ DROP TABLE R RENAME TABLE R INTO R' ADD COLUMN a AS $f(r_1,...r_n)$ INTO R_i DROP COLUMN r FROM R_i DEFAULT $f(r_1,...,r_n)$ RENAME COLUMN r IN R_i TO r' MERGE TABLE $R(c_R)$, $S(c_S)$ INTO T DECOMPOSE TABLE R INTO $S(s_1,...,s_n)$ [,T(t$_1$,...,t_n) on (PK|FK fk|cond)]

As soon as the file is uploaded, it is parsed and stored to the internal database. Instead of the database, the uploaded file could also be directly parsed, but for reasons of efficiency we selected to store the SMOs in a database. Then the system is able to answer provenance queries. The queries available to the end-users are the following:

(1) *How.* Which was the operation that introduced a specific table or a specific column?
(2) *When.* At which version the specific table/column was introduced?
(3) *Why.* Which was the sequence of operations that led to the introduction of a specific table/column?

When the answers are returned, both a graphical and a textual overview of the answer are presented to the user. A screenshot of the system is shown in Figure 2. Each node in the graph represents an SMO in its short form. When the mouse hovers on top of a node, it additional information is presented as a tooltip, whereas the whole list with the retrieved SMOs is shown on the right panel. A user can click on a specific SMO and refine the exploration based on this specific SMO. This will redraw the canvas and the textual overview, enabling the user to actively move between versions and SMOs.

In the example, shown in Figure 2, we demonstrate the evolution of the "revision" table from the wikimedia dataset. The "revision" table has been the result of merging two tables, i.e. the "revisiona" and the "revision_old" table. Before that the table "revisiona" had been created out of the decomposision of the table "revisionb" into "revisiona" and "text". On the other hand the table "revision_old" was the result of a decomposition of the table "old". The graph continues until the tables "cur" and "old" are created for the first time.

2.2 Query Engine

The query engine receives requests by the GUI and subsequently answers the issued queries. To answer queries about the evolution of a specific table/column we define first the notion of an *affecting schema modification operation.*

Definition 2.1. Let t be a table/column of a schema version m, namely the S_m, and Δ^{S_k,S_m}, $(k < m)$ be the sequence of schema modification operations between S_k and S_m. In addition, let $\delta_a(s)$ be the new schema information introduced with the SMO s and $\delta_d(s)$ the schema information that is being deleted using s. An SMO $s \in \Delta^{S_k,S_m}$ *affects* the table/column t, denoted by $af(t)$, if $t \in S_m$ and $t \in \delta_a(s)$.

An affecting schema modification operation captures the way a table/column was introduced between two schema versions. The intuition behind this is that, as we usually use the latest schema version, we base our exploration on this version and we would like to track how a specific table/field has been introduced.

2.2.1 *Answering how provenance queries.*

Assuming that we have the Δ^{S_k,S_m} available, it is trivial to identify the affecting SMO by just scanning the delta log once. This affecting SMO will be the answer to *how* provenance queries. As BiDEL guarantees that neither the rules for a single SMO, nor the version genealogy have cycles, the affecting SMO if it exists is *unique.*

2.2.2 *Answering when provenance queries.*

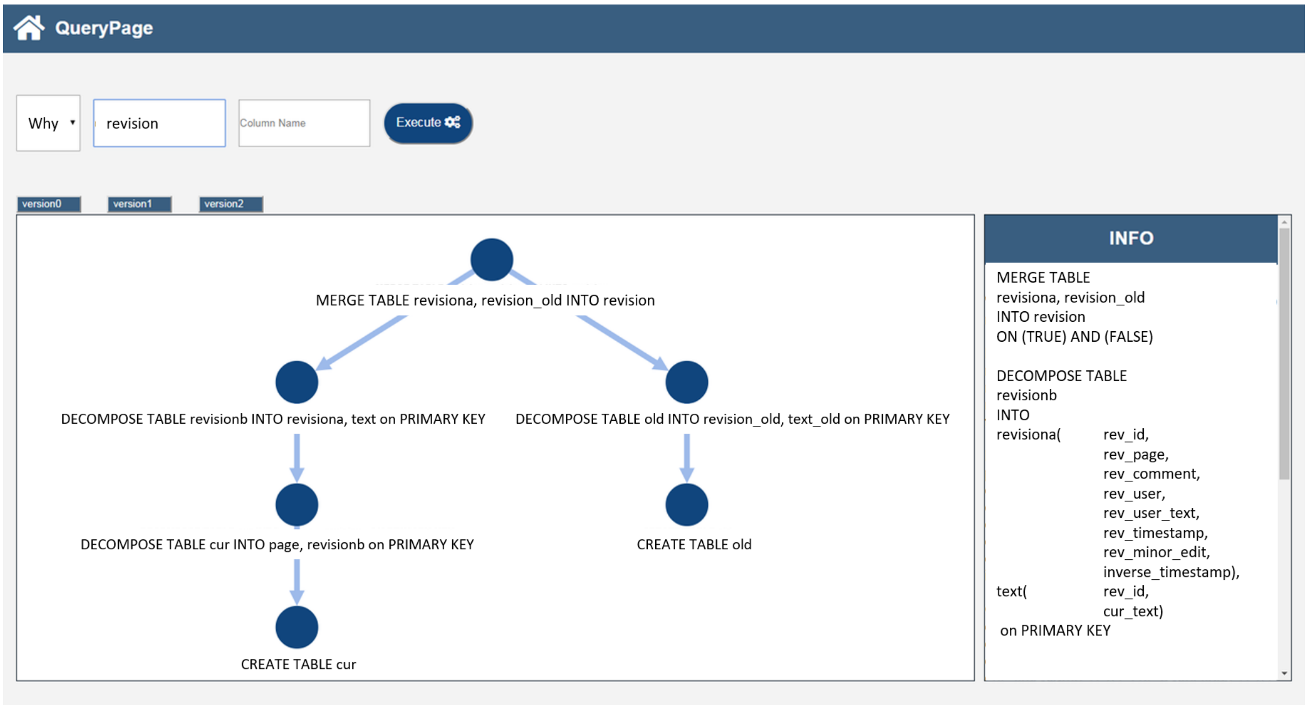
Next, for answering *when* provenance queries we are interested in identifying the version at which the affecting SMO appeared. This is again trivial as for each evolution step, we have the complete list of SMOs and we can easily identify the specific evolution step that included the specific affecting SMO.

2.2.3 *Answering why provenance queries.*

Presenting only answers to when and how provenance queries is not enough when complex evolution has occurred and we would like to see the sequence of events that led to the creation of a specific table or column. This sequence can be identified by answering *why* provenance queries. To be able to answer such queries we first define the *change sequence* of a specific SMO.

Definition 2.2. A *change sequence* for a schema modification operation $s \in \Delta^{S_k,S_m}$, denoted by CS^s, is the minimal sequence of schema modification operations in Δ^{S_k,S_m} such that $s \in CS^s$ and that when CS^s is applied to version k we get the fields/tables participating in s.

[1] https://db47122.inf.tu-dresden.de/



MERGE TABLE revisiona, revision_old INTO revision

DECOMPOSE TABLE revisionb INTO revisiona, text on PRIMARY KEY

DECOMPOSE TABLE old INTO revision_old, text_old on PRIMARY KEY

DECOMPOSE TABLE cur INTO page, revisionb on PRIMARY KEY

CREATE TABLE old

CREATE TABLE cur

```
MERGE TABLE
revisiona, revision_old
INTO revision
ON (TRUE) AND (FALSE)

DECOMPOSE TABLE
revisionb
INTO
revisiona(      rev_id,
                rev_page,
                rev_comment,
                rev_user,
                rev_user_text,
                rev_timestamp,
                rev_minor_edit,
                inverse_timestamp),
text(           rev_id,
                cur_text)
 on PRIMARY KEY
```

Figure 2: Showing the evolution of the "revision" table from the wikimedia dataset .

As in essence the whole Δ^{S_k,S_m} is a set of changes that lead to the fields/tables participating in s, we are looking for the minimal sequence of SMOs, in the sense that one cannot remove any of the SMOs in it, and still when applied to the version k you get the fields/tables participating in s. As such, a change sequence is providing the minimal information in order to understand how the specific table or column has been evolved to reach version m.

Similarly to an SMO , $\delta_a(CS)$ is the new schema information introduced when CS is being applied to S_k and $\delta_d(CS)$ the schema information that is being deleted from S_k using CS.

It can be easily proved that a change sequence for the BiDEL language if it exists, is *unique*.

Next we present an algorithm that given a sequence of SMOs capturing the Δ^{S_k,S_m}, it produces the change sequence for an SMO s. The algorithm is shown in Algorithm 1. The idea is that the algorithm starts by an input SMO and identifies the tables/columns that are also added to the schema, possibly by deleting other schema parts. Then it searches for the SMOs that delete the added information in order to add new information and so on.

Algorithm 1 Compute Change Sequence

```
Input: $s, \Delta^{S_k,S_m}$
Output: $CS^s$
procedure ComputeChangeSeq($s, \Delta^{S_k,S_m}$)
    $CS^s \leftarrow []$
    $CS^s.push(s)$
    for each $s' \in \Delta^{S_k,S_m}$ do
        if $\delta_a(s') \in \delta_d(CS^s)$ then
            $CS^s.push(s')$
    return $CS^s$
```

The algorithm is immediately proved by construction and has a complexity of $O(N)$ where N is the number of SMOs in Δ^{S_k,S_m}. As such we argue that exploring the evolution of big schemata with multiple changes is feasible and efficient.

It is quite easy to adapt the aforementioned algorithm for presenting the change sequence for a specific table/column instead of a specific SMO. The idea is first to look for the affecting SMO and then to use Algorithm 1 for identifying the corresponding change sequence.

The aforementioned algorithm is not only applicable for visualizing the change sequence of a table/column of the latest schema version, but can be also used to visualize the change sequence of a table/column from an arbitrary schema version of the past. In that case, instead of providing as input to our algorithms the Δ^{S_k,S_m} we could provide the Δ^{S_k,S_n} where $n < m$.

2.2.4 *Exploiting bidirectionality and inversibility.*

In addition, since BiDEL guarantees bidirectionality and inversibility (e.g. the inverse of the merge SMO is the decompose SMO), we could get the evolution steps for a table/column of a past schema version till we reach the most recent version by providing as input the Δ^{S_m,S_k} to our algorithms. This can be directly constructed from the Δ^{S_k,S_m} using the inverse SMOs of those in the Δ^{S_k,S_n}. Although bidirectionality and inversibility are not properties of all database evolution languages, if available, they contribute to a richer exploration experience on the available schema versions.

3 DEMONSTRATION OUTLINE

For demonstrating the functionality of the system, the wikimedia dataset will be used. The dataset contains more than 170 schema versions with the corresponding SMOs and offers a rich real dataset for experimentation. The demonstration will proceed in four phases.

(1) Exploring wikimedia's schema and raw evolution log. The demonstration will start by visualizing the latest version of the schema and exploring the various tables and columns. Then the raw evolution log will be opened and the difficulty to identify the actions that led to the generation of specific column/tables will be explained.
(2) System overview. Then an overview of the system will be presented along with the corresponding components. The ideas behind how, when and why provenance queries will be explained and the algorithms will be briefly described as well.
(3) Small tutorial. Next the users will be informed about the functionalities provided through the graphical user interface and some examples will be executed demonstrating various how, when and why provenance queries. More specifically, we will demonstrate how and when queries for selected tables and fields and then we will focus on answering why queries on tables that have been constructed with multiple SMOs. This will enable users to better depict the tool's usage and usefulness.
(4) "Hands-on" Phase. In this phase the conference participants will be invited to directly interact with the system, having the chance to explore the evolution of tables/columns through all versions available for the wikimedia dataset.

We also intend to release the system online before the conference, so that the conference participants to have a chance of exploring system's functionalities at their own pace and location.

4 CONCLUSIONS

In this demonstration we present a novel system enabling the VisuaL Exploration of multiple Schema Versions (VESEL) using provenance queries. The system gets as input the change log with the BiDEL SMO's over multiple schema versions and is then able to visually answer queries about how a specific table/field was introduced, in which schema version and which was the sequence of operations that led to the creation of the specific table/field.

As future work, several challenging issues need to be further investigated. For example, we already see BiDEL as just a language describing SMO's, with nice features. It is worth investigated the other proposed languages as well. This would obviously lead to a redefinition of some of our algorithms. In addition we intend to combine our approach with statistical information on the evolution, further guiding the user understanding on the schema evolution through multiple versions. Finally, it would be interesting to to study the effects of visualizing schema evolution in collaborative/multi-user or distributed environments that usually are prone to schema conflicts.

5 ACKNOWLEDGMENTS

Work presented in the paper is part of the BOUNCE project (H2020 #777167) that has received funding from the European Union's Horizon 2020 Research and Innovation Programme. Any opinions, results, conclusions, and recommendations expressed in this material are those of the authors and do not necessarily reflect the views of BOUNCE or the European Commission.

6 REFERENCES

[1] H. Kondylakis and D. Plexousakis, "Ontology evolution in data integration: Query rewriting to the rescue," in *ER*, vol. 6998 of *Lecture Notes in Computer Science*, pp. 393–401, Springer, 2011.

[2] H. Kondylakis and D. Plexousakis, "Ontology evolution: Assisting query migration," in *ER*, vol. 7532 of *Lecture Notes in Computer Science*, pp. 331–344, Springer, 2012.

[3] H. Kondylakis and D. Plexousakis, "Exelixis: evolving ontology-based data integration system," in *SIGMOD Conference*, pp. 1283–1286, ACM, 2011.

[4] H. Kondylakis and D. Plexousakis, "Ontology evolution without tears," *J. Web Semant.*, vol. 19, pp. 42–58, 2013.

[5] P. A. Bernstein and S. Melnik, "Model management 2.0: manipulating richer mappings," in *Proceedings of the ACM SIGMOD International Conference on Management of Data, Beijing, China, June 12-14, 2007*, pp. 1–12, 2007.

[6] C. Curino, H. J. Moon, A. Deutsch, and C. Zaniolo, "Automating the database schema evolution process," *VLDB J.*, vol. 22, no. 1, pp. 73–98, 2013.

[7] C. Curino, H. J. Moon, A. Deutsch, and C. Zaniolo, "Update rewriting and integrity constraint maintenance in a schema evolution support system: PRISM++," *PVLDB*, vol. 4, no. 2, pp. 117–128, 2010.

[8] H. J. Moon, C. Curino, M. Ham, and C. Zaniolo, "PRIMA: archiving and querying historical data with evolving schemas," in *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009, Providence, Rhode Island, USA, June 29 - July 2, 2009*, pp. 1019–1022, 2009.

[9] K. Herrmann, H. Voigt, A. Behrend, and W. Lehner, "Codel - A relationally complete language for database evolution," in *Advances in Databases and Information Systems - 19th East European Conference, ADBIS 2015, Poitiers, France, September 8-11, 2015, Proceedings*, pp. 63–76, 2015.

[10] M. Hofmann, B. C. Pierce, and D. Wagner, "Symmetric lenses," in *Proceedings of the 38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2011, Austin, TX, USA, January 26-28, 2011*, pp. 371–384, 2011.

[11] K. Herrmann, H. Voigt, A. Behrend, J. Rausch, and W. Lehner, "Living in parallel realities: Co-existing schema versions with a bidirectional database evolution language," in *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017*, pp. 1101–1116, 2017.

[12] C. Curino, H. J. Moon, L. Tanca, and C. Zaniolo, "Schema evolution in wikipedia - toward a web information system benchmark," in *ICEIS 2008 - Proceedings of the Tenth International Conference on Enterprise Information Systems, Volume DISI, Barcelona, Spain, June 12-16, 2008*, pp. 323–332, 2008.

[13] S. Gao and C. Zaniolo, "Supporting database provenance under schema evolution," in *Advances in Conceptual Modeling - ER 2012 Workshops CMS, ECDM-NoCoDA, MoDIC, MORE-BI, RIGiM, SeCoGIS, WISM, Florence, Italy, October 15-18, 2012. Proceedings*, pp. 67–77, 2012.

[14] H. Kondylakis and N. Papadakis, "Evordf: evolving the exploration of ontology evolution," *Knowledge Eng. Review*, vol. 33, p. e12, 2018.

[15] H. Kondylakis, M. Despoina, G. Glykokokalos, E. Kalykakis, M. Karapiperakis, M. Lasithiotakis, J. Makridis, P. Moraitis, A. Panteri, M. Plevraki, A. Providakis, M. Skalidaki, A. Stefanos, M. Tampouratzis, E. Trivizakis, F. Zervakis, E. Zervouraki, and N. Papadakis, "Evordf: A framework for exploring ontology evolution," in *ESWC (Satellite Events)*, vol. 10577 of *Lecture Notes in Computer Science*, pp. 104–108, Springer, 2017.

[16] M. Grinberg, *Flask Web Development - Developing Web Applications with Python.* O'Reilly, 2014.

[17] M. Franz, C. T. Lopes, G. Huck, Y. Dong, S. O. Sümer, and G. D. Bader, "Cytoscape.js: a graph theory library for visualisation and analysis," *Bioinformatics*, vol. 32, no. 2, pp. 309–311, 2016.

[18] K. Herrmann, H. Voigt, T. Seyschab, and W. Lehner, "Inverda - the liquid database," in *Datenbanksysteme für Business, Technologie und Web (BTW 2017), 17. Fachtagung des GI-Fachbereichs „Datenbanken und Informationssysteme" (DBIS), 6.-10. März 2017, Stuttgart, Germany, Proceedings*, pp. 619–622, 2017.