VESEL: VISUAL EXPLORATION OF SCHEMA EVOLUTION USING PROVENANCE QUERIES

Christos Athinaïou, Haridimos Kondylakis
Institute of Computer Science, FORTH-ICS, Heraklion, Crete, Greece
Computer Science Department, University of Crete, Crete, Greece

BIGVIS@EDBT 2019
SOME FACTS ABOUT DATABASES

Everything that exists it is only change”
-Heraclitus 535 BCE

- Database schemata are subject to continuous change
- Recent databases do not proactively support schema evolution
  - Developers have to migrate data from one database version to another using error-prone and complex ETL scripts
IDEALLY

➢ We would like tools enabling automatic data migration
➢ Tools enabling the understanding of what has been changed!
   ▪ Identify modelling choices of the past
   ▪ Do not reinvent the wheel, avoid the same mistakes again!
PROPOSED SOLUTIONS FOR SCHEMA EVOLUTION

- **Model Management approach**
  - tools to match, diff, merge and extract mappings between schema versions

- **Multiple tools offer Schema Modification Operations (SMOs)**
  - implemented for data migration and for rewriting past queries to work in the new versions

```sql
EVOLUTION START [FROM nameOld]
EVOLUTION COMMIT AS nameNew;
CREATE TABLE R(\(e_1, \ldots, e_n\))
DROP TABLE R
RENAME TABLE R INTO R'
ADD COLUMN a AS \(f(r_1, \ldots, r_n)\) INTO \(R_i\)
DROP COLUMN r FROM \(R_i\) DEFAULT \(f(r_1, \ldots, r_n)\)
RENAME COLUMN r IN \(R_i\) TO \(r'\)
MERGE TABLE R(\(e_R\)), S(\(e_S\)) INTO T
DECOMPOSE TABLE R INTO S(\(s_1, \ldots, s_n\))
\([,T(t_1, \ldots, t_n)\) on (PK|FK|fk|cond)]
```
RUNNING EXAMPLE

MERGE TABLE revisiona, revision_old INTO revision

S3

⋯

RENAME TABLE text into new_text

DECOMPOSE TABLE revisionb
INTO revisiona, text on PRIMARY KEY

DECOMPOSE TABLE cur
INTO page, revisionb on PRIMARY KEY

CREATE TABLE cur

S2

⋯

RENAME TABLE text_old into prev_text

DECOMPOSE TABLE old
INTO revision_old, text_old on PRIMARY KEY

CREATE TABLE old

S1

⋯

BIGVIS@EDBT2019
ARE WE DONE YET?

Give the SMOs to the users!

Which SMO introduced table Revision?

In which schema version Revision was introduced?

Which is the sequence of SMOs that introduced Revision?
VESEL

A visual approach enabling the active exploration of schema evolution

Answers the following provenance questions

- **How**: Which was the operation that introduced a specific table or a specific column?
- **When**: At which version the specific table/column was introduced?
- **Why**: Which was the sequence of operations that led to the introduction of a specific table/column?
AN AFFECTING SCHEMA MODIFICATION OPERATION

Let $\Delta^{S_k, S_m}$, ($k < m$) be the SMOs between versions $S_k$ and $S_m$

An SMO $s$ in $\Delta^{S_k, S_m}$ affects a table/field $t$, if $t$ in $S_m$ and $t$ in $\delta_s(s)$

It captures the way a table/column was introduced between two schema versions
ANSWERING PROVENANCE QUERIES

- **How**: Which was the operation that introduced a specific table or a specific column?
  - Identify the affecting SMO scanning the delta log once

- **When**: At which version the specific table/column was introduced?
  - Identify to which version this affecting SMO belongs
AN EXAMPLE — ANSWERING HOW/WHEN QUERIES

MERGE TABLE revisiona, revision_old INTO revision

Which SMO introduced table Revision?

RENAME TABLE text into new_text

DECOMPOSE TABLE revisionb INTO revisiona, text on PRIMARY KEY

DECOMPOSE TABLE cur INTO page, revisionb on PRIMARY KEY

CREATE TABLE cur

In which schema version Revision was introduced?

RENAME TABLE text_old into prev_text

DECOMPOSE TABLE old INTO revision_old, text_old on PRIMARY KEY

CREATE TABLE old
ANSWERING PROVENANCE QUERIES

Why: Which was the sequence of operations that led to the introduction of a specific table/column?
CHANGE SEQUENCE

A change sequence for a schema modification operation \( s \) in \( \Delta^{Sk, Sm} \) denoted by \( CS^s \), is the \textit{minimal} sequence of schema modification operations in \( \Delta^{Sk, Sm} \) such that

\begin{itemize}
  \item \( s \) in \( CS^s \)
  \item when \( CS^s \) is applied to \( Sk \) we get the fields/tables participating in \( s \)
  \item one cannot remove any of the SMOs in the change sequence, and still when applied to the version \( Sk \) you get the fields/tables participating in \( s \)
\end{itemize}
AN EXAMPLE — ANSWERING WHY QUERIES

MERGE TABLE revisiona, revision_old INTO revision

RENAME TABLE text into new_text

DECOMPOSE TABLE revisionb INTO revisiona, text on PRIMARY KEY

DECOMPOSE TABLE cur INTO page, revisionb on PRIMARY KEY

CREATE TABLE cur

RENAME TABLE text_old into prev_text

DECOMPOSE TABLE old INTO revision_old, text_old on PRIMARY KEY

CREATE TABLE old

Which is the sequence of SMOs that introduced Revision?
THE CHANGE SEQUENCE (TREE)

MERGE TABLE revisiona, revision_old INTO revision

DECOMPOSE TABLE revisionb INTO revisiona, text on PRIMARY KEY

DECOMPOSE TABLE cur INTO page, revisionb on PRIMARY KEY

CREATE TABLE cur

DECOMPOSE TABLE old INTO revision_old, text_old on PRIMARY KEY

CREATE TABLE old
INTERESTING PROPERTIES

- A change sequence for the BiDEL language if it exists is unique.
- The change sequence is reversible and bidirectional.

- This means 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 $\Delta^{S_m, S_k}$ to our algorithms – constructed by directly inverting $\Delta^{S_k, S_m}$. 
DEMONSTRATION

Wikimedia containing more than 170 schema versions with the corresponding SMOs
CONCLUSIONS

VESEL is a novel system enabling the visual exploration of multiple schema versions, able to visualize the identified change sequence.

Advantages
- Scalability
- Simplicity

Future Work
- Investigate other SMO languages as well
- Provide also statistical information on the evolution
- Study the effects of visualizing schema evolution in collaborative/multi-user distributed environments
- What about conflicts?
QUESTIONS ?