Managing the History of Metadata in support for DB Archiving and Schema Evolution

Carlo A. Curino[1], Hyun J. Moon[2], and Carlo Zaniolo[2]

[1] Politecnico di Milano *carlo.curino@polimi.it*
[2] University of California, Los Angeles {*hjmoon, zaniolo*}*@cs.ucla.edu*

Abstract. Modern information systems, and web information systems in particular, are faced with frequent database schema changes, which generate the necessity to manage them and preserve the schema evolution history. In this paper, we describe the *Panta Rhei Framework* designed to provide powerful tools that: (i) facilitate schema evolution and guide the Database Administrator in planning and evaluating changes, (ii) support automatic rewriting of legacy queries against the current schema version, (iii) enable efficient archiving of the histories of data and metadata, and (iv) support complex temporal queries over such histories. We then introduce the *Historical Metadata Manager (HMM)*, a tool designed to facilitate the process of documenting and querying the schema evolution itself. We use the schema history of the Wikipedia database as a telling example of the many uses and benefits of *HMM*.

1 Introduction

The main goal of a traditional database, i.e., capturing the "current state of the modelled reality", is satisfactory only in simple applications [1]. More sophisticated information systems require a systematic archiving and management of the history of the database. In particular, in web information systems, such as Wikipedia, ethical and legal expectations for accountability and preservation arise for information that was once placed in the public domain and divulged to the world; thus policies and systems are needed to support historical archives for web information. This situation has generated renewed interest in broad research on temporal databases and on transaction-time databases, in particular. In addition to data evolution [2], these systems experience intense evolution of database schema, as we reported in our analysis [3] of the Wikipedia DB, which has experienced more than 170 different schema versions in its 4.5 years of lifetime. Schema evolution, already a serious problem in traditional information systems [4, 5], becomes even more critical in web information systems.

In order to effectively manage information systems under the realistic assumption that *Panta Rhei—everything is in flux*—so that not only data, but also schemas, queries and applications are in continuous evolution, we propose the *Panta Rhei Framework*[3]. This framework provides a unified solution to the

[3] The project homepage is: `http://yellowstone.cs.ucla.edu/schema-evolution/`.

problems of: (i) graceful database evolution in the presence of schema changes, (ii) transaction-time history archiving and querying of databases under schema evolution.

The first objective is achieved by $\mathcal{PRISM}$ [6,7], a tool that assists the Database Administrator (DBA) in designing a new schema and that automatically revises legacy queries to work on it. The second objective is achieved by the transaction-time database system *ArchIS* [8,9], and its extension $\mathcal{PRIMA}$ [10]: *ArchIS* supports efficient temporal queries on the archived history of (fixed-schema) databases, while $\mathcal{PRIMA}$ introduces the transparent support for complex temporal queries over archives with evolving schema. Both $\mathcal{PRISM}$ and $\mathcal{PRIMA}$ exploit an intuitive operational language of Schema Modification Operators (SMOs) to explicitly capture the semantics of the schema evolution.

In this paper, we further extend the framework by introducing the *Historical Metadata Manager (HMM)*, a tool capable of archiving and querying rich metadata histories. Most modern DBMS provide their meta-level information in the form of a virtual DB named `information_schema` (SQL:2003 standard). With the *HMM* we extend the scope of the `information_schema`, inasmuch as the complete history of the DB schema, not just its current snapshot, is preserved and queried. Moreover, the *HMM* exploits the SMO-based representation of the schema evolution to provide users with a better understanding of the semantics of each schema change.

Contributions The novel contributions of this paper are the following:

- the overall architecture of the *Panta Rhei Framework* for graceful schema evolution and historical archive management under schema evolution;
- the *History Metadata Manager*, a tool capable of providing an effective archival mechanism for metadata and a query facility enabling complex temporal queries on schema histories.

The paper is organized as follows: Section 2 provides an overview of the *Panta Rhei Framework* architecture. Section 3 describes the *Historical Metadata Manager (HMM)* and its capabilities. In Section 4, we discuss related works, while Section 5 is devoted to future developments. Section 6 draws our conclusions.

2 The *Panta Rhei Framework*

The *Panta Rhei Framework*, presented in Figure 1, aims at providing seamless support for evolution of both data and schema and complete history management. The objectives of the framework are as follows: (i) a workbench to assist the DBA designing schema revisions with tools for change impact analysis and automatic query rewriting, (ii) efficient archiving of DB histories, (iii) support for complex temporal queries over a historical archive under schema evolution, and (iv) archiving and temporal querying of the metadata history.

Objective (i—iii) were largely realized by $\mathcal{PRISM}$ [6,7], *ArchIS* [8,9] and $\mathcal{PRIMA}$ [10,11], but this paper is the first to describe the overall architecture of our *Panta Rhei Framework* and the design of the *History Metadata Manager (HMM)* addressing objective (iv).

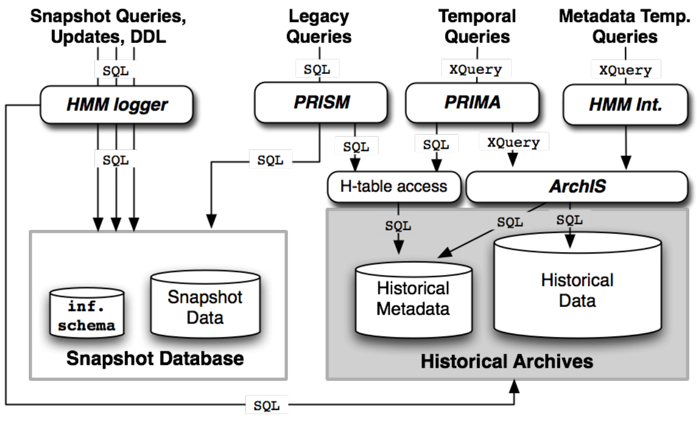


Fig. 1. *Panta Rhei Framework* Architecture

2.1 $\mathcal{PRISM}$

$\mathcal{PRISM}$ aims at automating the error-prone and time-consuming activity of schema evolution. The system provides: (i) a language of Schema Modification Operators (SMO) used for expressing complex schema changes in a concise way, (ii) tools that allow the DBA to evaluate the effects of schema changes, (iii) automated data migration, and (iv) optimized support for legacy queries on the current schema version.

SMOs provide an operational and concise way to capture schema evolution [3, 6], that specify how both schema and data evolve. By analyzing SMO sequences specified by the DBA, the system provides feedback on the proposed evolution, thus improving predictability of the evolution process. The same SMO representation is used to automate data migration and query adaptation by (i) generating SQL data migration scripts, and (ii) automatically rewriting legacy queries to operate on the new schema. Legacy queries are automatically supported in two alternative ways: by means of SQL views (among schema versions) or by automatically translating queries across schema versions by the query rewriting engine [12]. Wikipedia and its 170+ schema versions provided an invaluable testbed for validating $\mathcal{PRISM}$ and its query rewriting capabilities.

2.2 ArchIS

ArchIS [9] provides, in the context of *Panta Rhei Framework*, a powerful temporal data model based on XML, called V-Documents. V-Documents represent, in a temporally-grouped fashion, the attribute-level-timestamped history of the snapshot DB. This simplifies issues such as temporal coalescing [13]; moreover, the choice of XML allows the use of XQuery, which proved well-suited to express complex temporal queries [8]. To overcome the current performance limitations of XML, *ArchIS* exploits mature RDBMS technology, by shredding V-Documents and storing them in a relational format named *H-tables*. It also translates XQuery into the equivalent SQL/XML queries that can be executed over *H-tables*.

2.3 $\mathcal{PRIMA}$

$\mathcal{PRIMA}$ [10, 11] extends the functionalities of *ArchIS* by allowing the schema to evolve as data evolve. At every schema change, the snapshot data are archived using the schema version under which they first appeared, to achieve perfect archival quality[4]. This produces a transaction-time database that archives the data history under multiple schema versions. On the other hand, the task of writing temporal queries on such archive might become taxing for the users, since queries may potentially span over several schema versions (tens or even hundreds in the Wikipedia scenario). $\mathcal{PRIMA}$ addresses this issue by letting the users express temporal queries under a selected schema version, typically the current one, and then automatically rewriting them against pertinent schema versions.

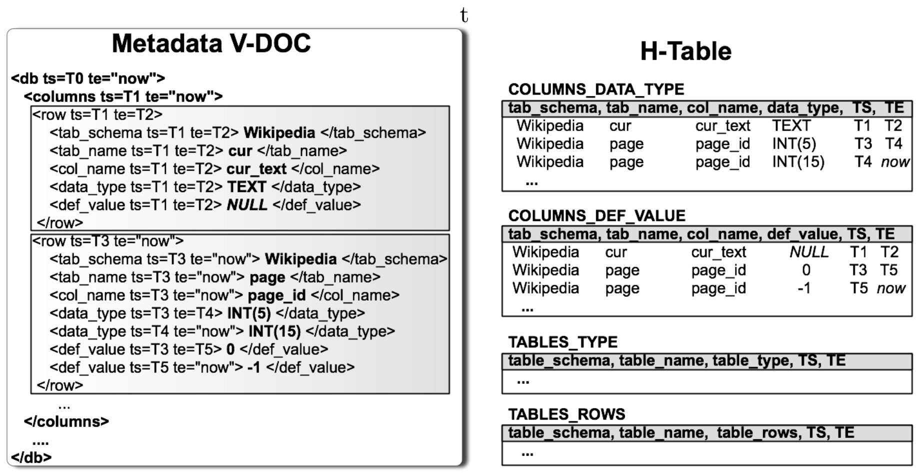
2.4 Putting all together: the overall architecture

The overall architecture of the *Panta Rhei Framework* is shown in Figure 1. The left-hand side of Figure 1 represents a snapshot database, augmented by the logging functionality, which updates the historical archives of both data and metadata (Section 3.3). The right-hand side of Figure 1 shows the core components of our architecture: $\mathcal{PRISM}$, *ArchIS*, $\mathcal{PRIMA}$, and the *HMM* querying interface. These components share a common data layer based on two archives, capturing data and metadata histories, and the SMO representation of the schema evolution. In particular, the *Historical Data DB* represents the transaction-time archive of the snapshot database—maintained by *ArchIS* in the relational format of *H-table*. In a similar way, the *Historical Metadata DB (HMDB)* stores the history of the meta-information that the SQL-compliant DBMSs offer in the `information_schema` and the SMO-based description of the schema changes. These data, besides being used by $\mathcal{PRISM}$ and $\mathcal{PRIMA}$, provide invaluable documentation when queried by the *Historical Metadata Manager (HMM)*, as further discuss in Section 3. One of the benefits of this integrated solution is that the effort needed to adopt one component, e.g., $\mathcal{PRISM}$, enables at no extra cost the entire set of functionalities of the *Panta Rhei Framework*.

3 Historical Metadata Manager

The *Historical Metadata Manager (HMM)* is a tool capable of storing and querying metadata histories. In this section, we discuss its main components, namely (i) the underlying data layer *Historical Metadata DB (HMDB)*, (ii) the querying interface allowing to pose complex temporal queries over the *HMDB*, and (iii) the *HMM logger*, a component that maintains the historical archives updated, based on the set of changes occurred in the snapshot data and metadata.

[4] Data migration is, in general, not information-preserving, thus can potentially compromise archival quality.


t

Metadata V-DOC

```
<db ts=T0 te="now">
  <columns ts=T1 te="now">
   <row ts=T1 te=T2>
     <tab_schema ts=T1 te=T2> Wikipedia </tab_schema>
     <tab_name ts=T1 te=T2> cur </tab_name>
     <col_name ts=T1 te=T2> cur_text </col_name>
     <data_type ts=T1 te=T2> TEXT </data_type>
     <def_value ts=T1 te=T2> NULL </def_value>
   </row>
   <row ts=T3 te="now">
     <tab_schema ts=T3 te="now"> Wikipedia </tab_schema>
     <tab_name ts=T3 te="now"> page </tab_name>
     <col_name ts=T3 te="now"> page_id </col_name>
     <data_type ts=T3 te=T4> INT(5) </data_type>
     <data_type ts=T4 te="now"> INT(15) </data_type>
     <def_value ts=T3 te=T5> 0 </def_value>
     <def_value ts=T5 te="now"> -1 </def_value>
   </row>
       ...
  </columns>
   ....
</db>
```

H-Table

COLUMNS_DATA_TYPE

tab_schema	tab_name	col_name	data_type	TS	TE
Wikipedia	cur	cur_text	TEXT	T1	T2
Wikipedia	page	page_id	INT(5)	T3	T4
Wikipedia	page	page_id	INT(15)	T4	*now*
...					

COLUMNS_DEF_VALUE

tab_schema	tab_name	col_name	def_value	TS	TE
Wikipedia	cur	cur_text	*NULL*	T1	T2
Wikipedia	page	page_id	0	T3	T5
Wikipedia	page	page_id	-1	T5	*now*
...					

TABLES_TYPE

table_schema	table_name	table_type	TS	TE
...				

TABLES_ROWS

table_schema	table_name	table_rows	TS	TE
...				

Fig. 2. *Historical Metadata DB*: a portion of V-Document and the corresponding H-table representations.

3.1 Historical Metadata DB

The *HMDB* captures the history of the `information_schema`, together with additional metadata needed in the *Panta Rhei Framework* (SMOs, user queries, etc.). Due to space limitation, we discuss only a selected subset of the *HMDB* content. We introduce it in a snapshot format, to later present the corresponding temporal archive.

```
schemata(schema_name, default_character_set_name)
tables(table_schema, table_name, table_type, table_rows)
columns(table_schema, table_name, column_name, data_type, default_value)
queries(query_id, query_text, schema_name, issue_timestamp)
query_exec(query_id, exec_timestamp, exec_success)
smos(smo_id, smo_text, smo_type, timestamp)
```

The first three relations, `schemata`, `tables`, and `columns`, come from the `information_schema`, while the relations named `queries`, `query_exec`, and `smos` store additional metadata needed by $\mathcal{PRISM}$ and $\mathcal{PRIMA}$. The `queries` relation archives the user queries (as templates), the target schema, and query issuing time. The table `query_exec` stores the results (boolean flag `exec_success`) of testing each of above query templates against the subsequent schema versions, used in the $\mathcal{PRISM}$ impact analysis. Lastly, the table `smos` stores the SMOs describing the semantics of each schema change.

In order to effectively and efficiently archive the history of these metadata, we represent them in the V-Document format. Figure 2 shows a portion of the V-document, capturing the evolution of two columns in the Wikipedia schema history, and its corresponding *H-table* representation—`ts` and `te` represent respectively the start and end time of the period in which a given value was valid in the snapshot DB. The first row element represents a column `cur.cur_text`

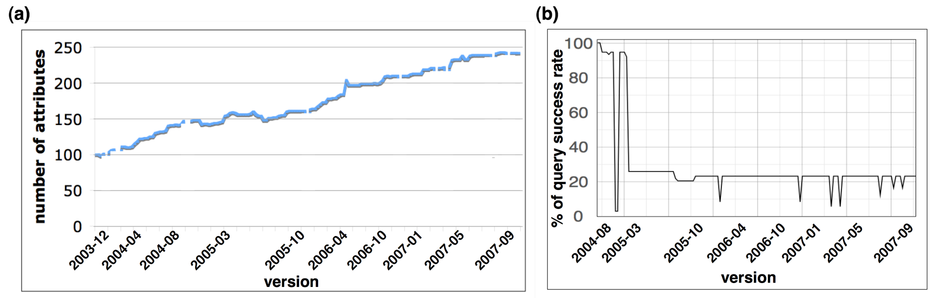


Fig. 3. Wikipedia results for: (a) Result of Query 2: Wikipedia schema size, and (b) Result of Query 5: success rate of Wikipedia legacy queries against the subsequent schema version.

which did not change between time T1 and T2, when it disappears forever. The second row element, in contrast, shows a column `page.page_id`, which was newly introduced at time T3, changed its type at time T4 (from INT(5) to INT(15)) and its default value at time T5 (from 0 to -1). The *H-table* representation in Figure 2 highlights how the XML data have been shredded in a set of tables with a regular structure: key-columns, nonkey-column, ts, te.

3.2 Querying the *HMDB*

The *HMM Interface* component of Figure 1 builds on top of the *ArchIS* storage engine to provide a simple query interface over the *HMDB*. By means of simple query examples on the Wikipedia schema evolution history, we illustrate how the *HMM* can be leveraged to obtain a deep insight on the evolution itself. The queries we present are grouped in two classes: (i) queries over the history of the `information_schema` (*Query 1*-*Query 3*), and (ii) queries exploiting SMOs and logged queries (*Query 4* and *Query 5*).

Simple but very useful temporal queries are snapshot queries as the following:

Query 1. What was the Wikipedia database schema valid on 2007-01-01?

```
for $db in document("wiki-hmdb.xml")/db,
  $tn in $db/Tables/row/table_name[@ts<="2007-01-01" and @te>"2007-01-01"]
return <table> {$tn/@text}, <cols>
        {$db/Columns/row[table_name=$tn]/
        column_name[@ts<="2007-01-01" and @te>"2007-01-01"]/text()}</cols>
       </table>
```

Similarly, it is possible (by range queries) to inspect specific portions of the history, retrieving all the subsequent modifications occured in a given period. Next, we give an example of temporal aggregate queries that can be exploited to observe general trends of the schema evolution:

Query 2. Retrieve the number of columns in the Wikipedia DB throughout the history.

```
for $db in document("wiki-hmdb.xml")/db,
     $t in $db/Schemata/rows/timestamp
return <tuple> <timestamp>{$t}</timestamp>,
        <cnt>{count($db/Columns/row/column_name[@ts<=$t and @te>$t])}</cnt>
       </tuple>
```

The output of such query is nicely rendered in Figure 3a, where it is easy to spot a net growing trend in the Wikipedia schema size, roughly 31% a year.

Furthermore, to analyze quality and stability of the design it is possible to pose queries retrieving stable or unstable portions of the schema as the following:

Query 3. Which table in Wikipedia DB schema that remained valid for the longest period?

```
let maxInterval:=
      max(for $tn in document("wiki-hmdb.xml")/db/Tables/row/table_name
          return $tn/@te-$tn/@ts)
for $tn in document("wiki-hmdb.xml")/db/Tables/row/table_name
where $tn/@te-$tn/@ts = maxInterval
return $tn/text()
```

The answer of this query reveals how the *user* table was the most stable in the DB schema. Another interesting class of queries exploits the explicit characterization of the change semantics provided by SMOs to track back (or forward) the evolution of information stored in a given schema element. Consider the following example:

Query 4. Retrieve the previous version of the information currently stored in table 'page'.

```
let $last_smo_timestamp := max(document("wiki-hmdb.xml")/db/Tables/row/
      smo[affected_table_name="page"]/timestamp)
let $last_smo := document("wiki-hmdb.xml")/db/Tables/row/
      smo[affected_table_name="page" and timestamp=$last_smo_timestamp]
return $last_smo/input_tables
```

Finally, by exploiting the information about query templates and their execution, it is possible to retrieve and visualize the impact of the schema evolution on the original user queries, as exemplified by the following query:

Query 5. What's the success rate of legacy queries (valid on 2004-08-15) after each Wikipedia schema change?

```
for $db in document("wiki-hmdb.xml")/db,
    $t in $db/Schemata/rows/timestamp
return
 <tuple>
   <schema-change-time>{$t}</schema-change-time>,
   <success-rate>{avg($db/query_exec/row
[query_id=$db/queries/row[execution_timestamp="2004-08-15"]/query_id]/
           success_flag)}</success-rate>
 </tuple>
```

The result of the query is shown in Figure 3b. The graph effectively highlights that a sudden drop occurred around 2005-03, due to a deep change of the article revision management in the Wikipedia DB [3]; this impacted over 70% of the queries. The spikes in the graph corresponds to syntactically incorrect schema versions, for which most of the queries failed, see [3] for details.

These relatively simple queries are naturally supported by the *HMM* on the history of metadata we archive. This querying capability provides a powerful tool for dissecting the metadata history, enabling a better understanding of undergoing schema evolution.

3.3 Archive Maintenance

This section briefly discusses how we maintain the historical archives up to date. The *HMM logger* of Figure 1 integrates the following functionalities:

Data Archive Update The problem of creating and maintaining a temporal archive of the snapshot DB has been addressed in [9]. Two solutions have been proposed, one based on active rules and the other on update logs. In the *ArchIS-DB2* experience, a set of DML triggers have been exploited to maintain the *H-table* archive updated by propagating every update of the snapshot DB to the temporal archive, while the *ArchIS-ATLaS* prototype obtains the same results by processing the update log. The choice between the two approaches depends on performance issues, the interested reader can refer to [9].

Metadata Archive Update To adapt the above methods to work on metadata, we exploit DDL triggers and SQL DDL logs analysis. The `information_schema` is, in fact, a virtual DB and no direct updates are issued on it. Recent versions of Oracle and MS SQL Server support DDL triggers. For MySQL and DB2, simple workarounds have been designed, based on the fact that the `information_schema` provides timestamps of the last modifications and that the frequency of schema changes is rather low. It is thus possible to monitor[5] it to detect changes, and use the timestamps stored in the snapshot `information_schema` itself, to obtain an accurate temporal archive. The *HMDB* also stores the sequences of Schema Modification Operators describing the semantics of the schema change. This is achieved in two alternative ways: (i) by systematically exploiting $\mathcal{PRISM}$ to explicitly model the schema evolution of the snapshot DB, or (ii) by observing the evolution and semi-automatically mining the corresponding SMOs.

4 Related Works

The long standing problems of schema evolution and versioning [14, 15] have been the subject of several research efforts surveyed in [16], and more recent approaches were proposed in [17–20]. Archiving and querying metadata histories has been discussed, to a limited extent, in [21], as a byproduct of temporal data

[5] This can be automated within the DBMS by means of *event* triggers (MySQL 5.1) or scheduled administration tasks (DB2).

archiving. Recent theoretical and practical advances on the related problems of query rewriting, mapping composition and invertibility appeared in [12, 22–24]. The *Panta Rhei Framework* builds on top of this solid theoretical foundations and provides, at the best of our knowledge, one of the most advanced, unified approaches to the issues of schema evolution and data and metadata history management.

5 Future Developments

Much of the *Panta Rhei Framework* is still under development; along with the implementation of new features and optimizations techniques, a substantial integration effort is taking place to provide an easily-deployable, unified tool-suite.

Mapping and query rewriting are two core features that our *Panta Rhei Framework* shares with most of the existing approaches to the problem of data integration. It is part of our research agenda to investigate the use of the techniques we exploit in $\mathcal{PRISM}$ and $\mathcal{PRIMA}$ to tackle data integration and schema evolution in a unified way, as we discuss in [25]. Moreover, metadata histories can be effectively exploited in the context of automatic schema matching to develop history-aware matching heuristics. Further investigation is on-going about the use of semantic data models, to capture uniformly schema changes and schema mappings [26] in a semantically rich model, which might enable automatic reasoning.

6 Conclusions

In this paper, we presented the overall architecture of the *Panta Rhei Framework*, that unifies the concurrently ongoing endeavors on temporal databases and schema evolution that have produced $\mathcal{PRISM}$ [6, 7], *ArchIS* [8, 9], and $\mathcal{PRIMA}$ [10, 11]. Thus, this is the first paper describing the *History Metadata Manager (HMM)*, a powerful tool we have developed for archiving and querying the history of the metadata evolution. Such information, representing the core knowledge on which $\mathcal{PRISM}$ and $\mathcal{PRIMA}$ operate, is invaluable for a Database Administrator trying to understand the schema evolution of an Information System. The unified XML model exploited allows the use of XQuery, which is well-suited to express complex temporal queries. The *Panta Rhei Framework* has been validated on the real-life schema evolution of the Wikipedia database.

References

1. Snodgrass, R.T.: The TSQL2 Temporal Query Language. Kluwer (1995)
2. Almeida, R.B., Mozafari, B., Cho, J.: On the evolution of wikipedia. In: Int. Conf. on Weblogs and Social Media. (March 2007)
3. Curino, C.A., Moon, H.J., Tanca, L., Zaniolo, C.: Schema Evolution in Wikipedia: toward a Web Information System Benchmark. ICEIS (2008)

4. Sjoberg, D.I.: Quantifying schema evolution. Information and Software Technology **35**(1) (1993) 35–44
5. Marche, S.: Measuring the stability of data models. European Journal of Information Systems **2**(1) (1993) 37–47
6. Moon, H.J., Curino, C.A., Deutsch, A., Hou, C.Y., Zaniolo, C.: Managing and querying transaction-time databases under schema evolution. In: VLDB. (2008)
7. Curino, C.A., Moon, H.J., Ham, M., Zaniolo, C.: Architecture and optimization for transaction-time dbs with evolving schemas. Demo proposal submitted for publication (2009)
8. Wang, F., Zaniolo, C.: An XML-Based Approach to Publishing and Querying the History of Databases. World Wide Web: Web Information Systems Engineering **8**(3) (2005) 233–259
9. Wang, F., Zaniolo, C., Zhou, X.: Archis: An xml-based approach to transaction-time temporal database systems. The International Journal of Very Large Databases (2008)
10. Curino, C.A., Moon, H.J., Zaniolo, C.: Graceful database schema evolution: the prism workbench. In: VLDB. (2008)
11. Moon, H.J., Curino, C.A., Zaniolo, C.: Architecture and optimization for transaction-time dbs with evolving schemas. Submitted for publication (2009)
12. Deutsch, A., Tannen, V.: Mars: A system for publishing XML from mixed and redundant storage. In: VLDB. (2003)
13. Clifford, J., Croker, A., Grandi, F., Tuzhilin, A.: On Temporal Grouping. In: Recent Advances in Temporal Databases, Springer Verlag (1995) 194–213
14. Roddick, J.: A Survey of Schema Versioning Issues for Database Systems. Information and Software Technology **37**(7) (1995) 383–393
15. Castro, C.D., Grandi, F., Scalas, M.R.: Schema versioning for multitemporal relational databases. Information Systems **22**(5) (1997) 249–290
16. Ram, S., Shankaranarayanan, G.: Research issues in database schema evolution: the road not taken. In: Boston University School of Management, Paper No: 2003-15. (2003)
17. Dyreson, C., Snodgrass, R.T., Currim, F., Currim, S., Joshi, S.: Validating quicksand: Schema versioning in auxschema. icdew **0** (2006) 82
18. Rizzi, S., Golfarelli, M.: X-time: Schema versioning and cross-version querying in data warehouses. In: ICDE. (2007) 1471–1472
19. Grandi, F., Mandreoli, F., Tiberio, P.: Temporal modelling and management of normative documents in xml format. Data Knowl. Eng. **54**(3) (2005) 327–354
20. Jørgensen, P.S., Böhlen, M.H.: Versioned relations: Support for conditional schema changes and schema versioning. In: DASFAA. (2007) 1058–1061
21. Wang, F., Zaniolo, C.: Representing and Querying the Evolution of Databases and their Schemas in XML. In: Intl. Workshop on Web Engineering, SEKE. (2003)
22. Nash, A., Bernstein, P.A., Melnik, S.: Composition of mappings given by embedded dependencies. In: PODS. (2005)
23. Bernstein, P.A., Green, T.J., Melnik, S., Nash, A.: Implementing mapping composition. VLDB J. **17**(2) (2008) 333–353
24. Fagin, R.: Inverting schema mappings. ACM Trans. Database Syst. **32**(4) (2007)
25. Curino, C.A., Tanca, L., Zaniolo, C.: Information systems integration and evolution: Ontologies at rescue. In: International Workshop on Semantic Technologies in System Maintenance (STSM). (2008)
26. Artale, A., Parent, C., Spaccapietra, S.: Evolving objects in temporal information systems. Ann. Math. Artif. Intell. **50**(1-2) (2007) 5–38