Supporting Database Provenance under Schema Evolution

Shi Gao and Carlo Zaniolo

University of California, Los Angeles
{gaoshi, zaniolo}@cs.ucla.edu

Abstract. Database schema upgrades are common in modern information systems, where the provenance of the schema is of much interest, and actually required to explain the provenance of contents generated by the database conversion that is part of such upgrades. Thus, an integrated management for data and metadata is needed, and the Archived Metadata and Provenance Manager (AM&PM) system is the first to address this requirement by building on recent advances in schema mappings and database upgrade automation. Therefore AM&PM (i) extends the Information Schema with the capability of archiving the provenance of the schema and other metadata, (ii) provides a timestamp based representation for the provenance of the actual data, and (iii) supports powerful queries on the provenance of the data and on the history of the metadata. In this paper, we present the design and main features of AM&PM, and the results of various experiments to evaluate its performance.

1 Introduction

The importance of recording the *provenance*, or *lineage*, of any information of significance is now widely recognized, and a large body of research was produced on provenance management for scientific workflows and databases [4,15,18]. Previously proposed provenance systems focus on capturing the "why", "where" and "how" facets of provenances [5,13], and support a rich set of provenance-related functions and queries. Unfortunately, current proposals assume that the whole database content was created by the external actions of users and applications. In reality, modern information systems, particularly big science projects, undergo frequent schema changes whereby the database under the old schema is migrated into the new one conforming to the new schema. Thus the current database is the combined result of (i) the external actions that entered the original information (e.g., via SQL inserts or updates), and (ii) the migration steps that then transformed the data as part of the schema evolution process. Therefore explaining the provenance of current data often requires 'flashing-back' to the original, schema, external transaction, and data as this was originally created, i.e., before the conversion step (or the succession of steps) that transformed the original information into the equivalent one that is now stored under the current schema. Thus supporting provenance under schema evolution can be challenging and is hardly surprising that previous works have ignored this challenge by assuming that the database schema is fixed and does not change with time. However this assumption is utterly unrealistic as illustrated by the UCLA testbed collecting the schema history for 20+ large information systems, including Mediawiki/Wikipedia, Ensembl GeneticDB and various CERN Physics DBs [8, 9]. For instance, the database of Mediawiki software supporting Wikipedia has experienced more than 300 schema versions in its nine years of life and similar observations hold for the other information systems. Fortunately, recent results obtained by the UCLA Panta Rhei project [10, 11] have significantly simplified the problems of preserving the history of the database schema and flashing back to the original schema and source data.

The results presented in [10,11] provide the conceptual basis for the Archived Metadata and Provenance Manager (AM&PM) described in this paper; AM&PM is the first system designed to support the provenance of both data and metadata in the presence of schema evolution. Thus, AM&PM (i) extends the SQL information schema with timestamp based archival, (ii) uses Schema Modification Operators (SMOs) and Integrity Constraints Modification Operators (ICMOs) [10,11] to map between different versions of the database schema, and (iii) supports powerful queries for tracing the provenance of data and metadata, to support functions such as database auditing and data-recovery [6, 18].

The reminder of this paper is organized as follows: In Section 2, we review the schema evolution language used to describe schema evolution. In Section 3, we present the details of our approach for archiving the provenance of data and metadata. In Section 4, we discuss the design and features of the AM&PM system. Section 5 present an evaluations on the space and times required by the provenance tables and queries. Related work is summarized in Section 6, and our conclusions are reported in Section 7.

2 Schema Evolution Languages

The Schema Modification Operators (SMOs) were introduced in [10] as a very effective tool for characterizing schema upgrades and automating the process of migrating the database and upgrading the query-based applications. Since in many situations, the schema integrity constraints are also changed along with schema structure, Integrity Constraints Modification Operators (ICMOs) were introduced in [11] to characterize integrity constraints and automate the process of upgrading applications involving integrity constraint updates. Three types of integrity constraints are considered: primary key, foreign key, and value constraint. Extensive experience with the schema evolution testbed [8] shows that the combination of SMOs and ICMOs displayed in Table 1 can effectively capture and characterize the schema evolution history of information systems. In Table 1, (R, S) denote relational tables and (a, b, c, d) denote columns in tables. Here we present a simple example to show how schema evolution language works.

Example 1: Consider one database upgrade scenario as follows:

V_1: *Employee($\underline{ID}$, Name, Pay)*

V_2: *Employee_Info($\underline{ID}$, Name)* *Employee_Pay($\underline{ID}$, Salary)*

The initial schema version is V_1. Then the schema evolves into V_2 and the table *Employee* is decomposed into two tables: *Employee_Info* and *Employee_Pay.*

SMO
`CREATE TABLE R(a,b,c)`
`DROP TABLE R`
`RENAME TABLE R INTO T`
`COPY TABLE R INTO T`
`MERGE TABLE R, S INTO T`
`PARTITION TABLE R INTO S WITH cond, T`
`DECOMPOSE TABLE R INTO S(a,b), T(a,c)`
`JOIN TABLE R,S INTO T WHERE conditions`
`ADD COLUMN d INTO R`
`DROP COLUMN c FROM R`
`RENAME COLUMN b IN R TO d`

ICMO
`ALTER TABLE R ADD PRIMARY KEY pk1(a; b) [policy]`
`ALTER TABLE R ADD FOREIGN KEY fk1(c; d) REFERENCES T(a; b) [policy]`
`ALTER TABLE R ADD VALUE CONSTRAINT vc1 AS R:e = 0 [policy]`
`ALTER TABLE R DROP PRIMARY KEY pk1`
`ALTER TABLE R DROP FOREIGN KEY fk1`
`ALTER TABLE R DROP VALUE CONSTRAINT vc1`

Table 1: Schema Evolution Language: SMO and ICMO

Underlines indicate the primary keys. The transformation from V_1 to V_2 can be described as:
1. RENAME COLUMN *Pay* IN *Employee* To *Salary*
2. DECOMPOSE TABLE *Employee* INTO *Employee_Info(ID, Name)*, *Employee_Pay(ID, Salary)*

The composition of these two SMOs produces a *schema evolution script*, which defines the mapping between different versions of database schema in a concise and unambiguous way. Schema evolution scripts can be characterized as Disjunctive Embedded Dependencies (DEDs) and converted into SQL scripts [10]. We will use the schema evolution scripts to archive schema changes and perform provenance query rewriting, as discussed in Section 4.

3 Provenance Management in AM&PM

The AM&PM system is designed to provide the integrated management for the provenance of data and metadata, in order to support a range of queries, including the following ones:
Data Provenance Queries: Given a data tuple *d* in the current database, where does *d* come from (e.g. schema, table, and column)? When was *d* created? and how (e.g. by which transaction and user)?
Schema Provenance Queries: AM&PM allows users to retrieve past versions of database schema along with the SMOs and ICMOs which describe the schema evolution process in databases. Similarly, we can answer where-, when-, and how-queries for the basic schema elements in databases.
Evolution Statistic Queries:: AM&PM supports statistical queries about the evolution of the database content and schema.

Information Schema History. In a relational database management system, the *information schema*, also called *data dictionary* or *system catalog*, is the

(a) Employee_Info

ID	Name
100	Sam
200	Bob

(b) Employee_Pay

ID	Salary
100	2000
200	3000

(c) Employee_Info_Mapping

Tuple_ID	Attribute	TID
100	*ID*	t1
100	*Name*	t1
200	*ID*	t2
200	*Name*	t2

(d) Employee_Pay_Mapping

Tuple_ID	Attribute	TID
100	*ID*	t1
100	*Salary*	t1
200	*ID*	t2
200	*Salary*	t2

(e) SMO

SID	SMO	Source	Target	Timestamp
s1	smo_1	V_1	V_2	2007-11-20 12:38:44
s2	smo_2	V_1	V_2	2007-11-20 12:42:07

(f) ICMO

ICID	ICMO	Source	Target	Timestamp
i1	$icmo_1$	V_2	V_3	2008-02-01 13:04:46

(g) Transaction

TID	DB_User	Schema	Timestamp
t1	Guest1	V_1	2006-09-15 11:09:12
t2	Guest1	V_1	2006-10-16 17:26:09

(h) Tran_Text

TID	Trans
t1	$tran_1$
t2	$tran_2$

(i)

smo_1: RENAME COLUMN *Pay* IN *Employee* To *Salary*
smo_2: DECOMPOSE TABLE *Employee* INTO *Employee_Info(ID, Name)*, *Employee_Pay(ID, Salary)*
$icmo_1$: ALTER TABLE *Employee_Info* ADD FOREIGN KEY pk1 (*ID*) REFERENCE *Employee_Insurance* (*Employee_ID*)
$tran_1$: INSERT INTO *Employee* VALUES (100, 'Sam', 2000)
$tran_2$: INSERT INTO *Employee* VALUES (200, 'Bob', 3000)

Fig. 1: An example of AM&PM provenance database

database for the metadata. As specified in the SQL:2003 standards [12], the information schema consists of a set of tables and views that contain information about the schema and other metadata. In order to answer queries on past versions of the schema and the provenance of the current database, AM&PM stores the transaction time history of the information schema.

Data Provenance. The provenance of each attribute value in the current database is recorded by preserving the information about the transactions that generated it, including the transaction ID, the timestamp at which the transaction executed, and the user who issued the transaction. Thus, AM&PM introduces the tables *Transaction* and *Tran.Text* to archive the transactions with their timestamps. The mapping tables are introduced to specify the relationship between transactions and tuples. This is all it is needed for data generated under the current versions of the schema. However, the current data could be the result of database migration after schema upgrades. Therefore, we must use the SMOs and ICMOs to reconstruct the source schema and data value, using the process described next under 'schema provenance'.

Schema Provenance. Schema provenance refers to the combination of the past versions of database schema and the schema evolution history. AM&PM exploits the information schema tables to store the metadata of past schemas. Two tables, *SMO* and *ICMO*, are created to archive the schema evolution history. Every schema change in the database upgrade is converted to a schema evolution operator (SMO or ICMO) and stored with its timestamp. Examples of these two tables are shown in tables (e) and (f) of Fig. 1. The tuple *s1* in the *SMO* table represents one schema change in the transformation from V_1 to V_2.

Auxiliary Information. AM&PM can also support some optional tables to provide more information about provenance (e.g., the values before the last data update and statistics).

Example 2: Fig. 1 shows the provenance database for the case of *Example 1.* The relationship between data and transactions is stored in mapping tables (c)

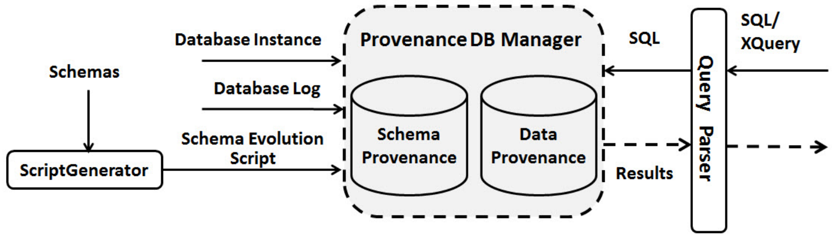


Fig. 2: Architecture of AM&PM system

and (d). For instance, the transaction which affects the value "Sam" in attribute *Name* of table *Employee_Info* can be found in table *Employee_Info_Mapping*, as "t1". This transaction id denotes $tran_1$ as the transaction that created/updated the value. The provenance queries are answered by evaluating the timestamps. For instance, say that a user issues a how-provenance query of the value "Sam" to verify this data. The last transaction which affects this data is $tran_1$ with timestamp "2006-09-15 11:09:12". All the schema evolution operators that have occurred from this timestamp until now may affect the data. Thus, we obtain smo_1, smo_2, and $icmo_1$, but then we find that only smo_1 and smo_2 affect the attribute *Name*. We return the timestamped $tran_1$, smo_1, and smo_2 as the how-provenance of "Sam".

4 AM&PM System

Fig. 2 depicts the high-level architecture of AM&PM system. The AM&PM system consists of two main components: *the provenance database manager* and *the query parser*.

The AM&PM provenance database manager analyzes the input data and constructs the provenance database, which is a combined relational database of *schema provenance tables* and *data provenance tables*. Schema provenance tables include the schema information of past database versions, SMOs, ICMOs, and database statistics, while data provenance tables (e.g. transaction table) store the information of transactions applied to the content of database and the relationship between transactions and data. The schema provenance tables are filled by parsing the past schemas and the schema evolution scripts. A schema evolution script is the set of SMOs and ICMOs used to upgrade database, as defined in Section 2. In AM&PM, the schema evolution script is produced by a module called *ScriptGenerator*. The basic idea of the ScriptGenerator is to compare different schemas and express the changes using the schema evolution language. The content of data provenance tables is extracted from the current database and the database transaction log. The database transaction log offers the source values and relevant transactions.

Once provenance database is filled, users can issue the provenance queries in SQL or XQuery. Many provenance queries are temporal queries with time constraints, and XQuery can facilitate the expression of complex temporal provenance queries [19]. Provenance queries written in SQL are directly executed in our provenance database. However, queries expressed in XQuery are rewritten to

SQL statements by the AM&PM query parser. For that, we utilize the algorithm presented in [19], based on the SQL/XML standard [16].

Extended Functionality: At the present time, AM&PM supports the queries and functions described above; we plan to support the functions described next in future extensions.
Provenance Query Rewriting Suppose we have a provenance query set, and some changes occur in the current database schema. Can the current provenance query be automatically upgraded to work on the new schema? To solve this problem, we are investigating the extensions of query rewriting techniques proposed in PRISM++ [11].
Provenance Propagation in Data Warehouses If a value in the source database is changed, how to propagate this change to the current database? This problem is studied in [14] in data warehousing and we plan to develop similar techniques in AM&PM.

5 Experimental Evaluation

Our experiments seek to evaluate (i) the space overhead of provenance capture, and (ii) the execution time of provenance queries over AM&PM provenance database. Note that there is no comparison with other provenance management systems since AM&PM is the first system using a temporal model to archive the provenance of data and metadata under schema evolution. Most queries discussed in Section 3 are not supported by other existing systems.

Experiment Setup Our experiments used four real-world data sets: Mediawiki, Dekiwiki, KtDMS, and California Traffic [1,8]. The first three data sets are the database schemas of web information systems. As shown in Table 2, these three information systems experienced many schema changes during their lifetime. We obtained the database schemas of these information systems from [8] and used them in our experiments on schema provenance.

In AM&PM, the database transaction log is required to fill the data provenance tables. Thus, we constructed a California Traffic database according to the website of California transportation department [1]. Every update of highway information in the website was captured and rewritten to a transaction applied to traffic database.Thus, there are two main tables in the traffic database: *highway_accident* and *highway_condition*. For instance, When there is a new accident in website, a transaction inserting a tuple to *highway_accident* is generated in the traffic database. Our real-world California Traffic database contains 5369 transactions. Since the California Traffic database is relatively small, we also generated four large synthetic traffic databases to verify the scalability of our approach.

The values in synthetic traffic databases are uniformly sampled from the real-world California Traffic database. The number of transactions and the size of data provenance tables of the four synthetic traffic databases are (50000, 22M), (100000, 42M), (200000, 91M), (500000, 231M) respectively.

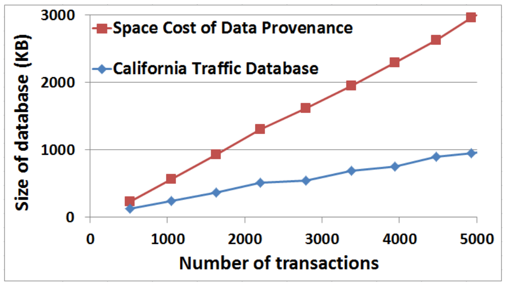


Fig. 3: Size of data provenance tables

Table 2: Size of schema evolution tables

System	Lifetime (years)	# of Versions	Space (MB)
Dekiwiki	4.0	16	0.933
KtDMS	6.3	105	10.717
Mediawiki	9.0	323	18.331

Our experiment environment consisted of a workstation with Intel Xeon E5520 CPU and 4GB RAM running Ubuntu 11.10. MySQL 5.1.61 was used as the backend database for AM&PM. We restarted the database server between query runs and used the average of five runs to report our the results.

Space Overhead of Provenance Capture: The AM&PM provenance database consists of data provenance tables and schema provenance tables, as shown in Section 4. We evaluate the space overhead of these two parts respectively.

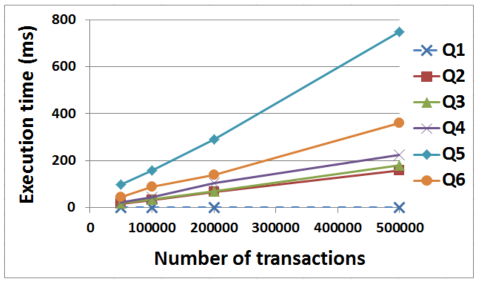
Data Provenance Tables Figure 3 shows the space overhead of such tables as the number of transactions of real-world California Traffic database increases. We observe that the size of data provenance tables is proportional to the number of transactions. Since AM&PM allows users to retrieval the content of transactions, the space overhead for storing transactions is unavoidable. The relationship between the space overhead of data provenance tables and the size of California Traffic database is also approximately linear. Archiving the data provenance leads to about 300% space overhead for he traffic database. This high relative overhead reflects the fact that transactions and timestamps require several bytes, whereas the data elements in the traffic database are small—e.g., names of highways and regions.

Schema Provenance Tables We investigate the space overhead of schema provenance tables in three information systems: Mediawiki, Dekiwiki and KtDMS, as shown in Table 2. The versions in Table 2 are defined according to their SVN (Apache Subversion) versions. The result shows that the space cost of schema provenance tables depends on the number of history schema versions. KtDMS and Mediawiki need more space for schema provenance than Dekiwiki because they have more database versions and schema changes. In particular, for Mediawiki which experienced many schema changes, it only takes 18 MB to store the schema evolution history of 323 schemas. Comparing with the size of Wikipedia database [2] under the latest schema, which is about 34.8 GB uncompressed, the space overhead of schema provenance tables is very small. Therefore, the space overhead of schema provenance is typically modest.

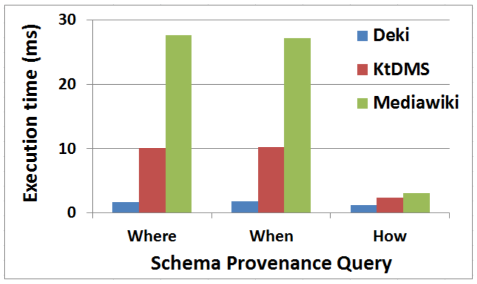
Query Execution Time: We study the query performance by measuring the execution time. To be specific, the execution time refers to the time used by

Query	Description
Q1	find the creation time of the tuple with id 2357 in highway_accident
Q2	find the transaction which generates the tuple with id 19009 in highway_condition
Q3	find the number of accidents happening on 04/02/2012
Q4	find the number of highway condition records on 04/03/2012
Q5	find the ids of accidents happening in the area of West Los Angeles between '04/04/2012 18:00:00' and '04/04/2012 23:00:00'
Q6	find the descriptions of highway condition updates happening in the area of Central LA between '04/04/2012 18:00:00' and '04/04/2012 23:00:00'

Table 3: Data provenance queries for evaluation



(a) Data provenance query



(b) Schema provenance query

Fig. 4: Execution time for data and schema provenance queries

the backend database to execute queries written in standard SQL. The types of provenance queries tested are those discussed in Section 3.

Data Provenance Queries Since the real-world California Traffic database is relatively small, most provenance queries are answered within 1 ms. We use the synthetic traffic databases to evaluate the performance of data provenance query. Table 3 shows the set of data provenance queries we prepared for experiments. Q1 and Q2 are the when- and how- provenance queries. Q2 also involves join operation. Q3 and Q4 are temporal provenance queries with aggregates. Q5 and Q6 are temporal provenance queries with joins.

Figure 4a shows the query execution time as the number of transactions in synthetic traffic databases increases. The performance of simple provenance query like Q1 is not affected by the number of transactions. For all of the four synthetic databases, it takes almost the same time (around 0.4 ms) to execute Q1. The execution time of other provenance queries is linear with the number of transactions. For queries with aggregates, they have to scan the *transaction* table to get the count. For queries with joins, they require to join the data provenance tables whose size is proportional to the number of transactions as shown in Figure 3.

Schema Provenance Query We evaluate the performance of schema provenance query using Dekiwiki, KtDMS, and Mediawiki. For each information system, we randomly pick 5 tables and 5 columns. Then the where-, why-, and how-provenance queries of these tables and columns are executed as test queries. The results of where-, when-, and how- queries on schema provenance are as follows: the source schema, the creation time and the schema evolution operators which help generate the tables and columns. Figure 4b shows the average execution time of schema provenance queries. The result indicates that the performance of schema provenance query depends on the size of schema evolution tables. It takes more time to run schema provenance queries in the system which has larger schema provenance tables. We also observe that how-query takes less time because it only scans the SMO or ICMO table, while where- and when- queries need to join the information schema tables (COLUMNS and VERSIONS) to associate the schema element with its version and timestamp.

Statistic Query We prepared a set of statistic queries to compute five statistics for each version of database schema: the number of tables, the number of columns, the number of primary keys, the number of foreign keys, and the lifetime. The overall execution time of statistic query set is 16.6 ms for Deikiwiki, 131.0 ms for KtDMS, and 217.3 ms for Mediawiki. Not surprisingly, the execution time of these statistic queries is linear with the size of schema provenance tables.

6 Related Work

There has been a large body of research on provenance in databases and, because of space restrictions, we will refer our users to the overview papers [4, 6, 15].

The differences between where- and why- provenance in databases were studied in [5], while Green et al. [13] studied the how-provenance. The challenge of archiving the provenance of metadata was discussed in [18]. Schema mapPIng DEbuggeR (SPIDER) [3, 7] used nested relational model and schema mapping language to record the data mapping between two schemas. Provenance is computed according to the routes of possibly mappings. But [3, 7] didn't propose a model to archive the schema changes in schema evolution. On the other hand, the Metadata Management System (MMS) [17] comes closer to our proposal. To associate data and metadata, MMS stores the queries as values in a relational database Traditional join mechanisms are extended to support the join specified by the queries stored in relations. However, the evolution of metadata is not considered in MMS.

7 Conclusion and Future Work

In this paper, we present an integrated approach to manage the provenance of both data and metadata under schema evolution. The provenance of data and metadata is archived using relational databases. The schema evolution history is stored using schema evolution language. Thus, AM&PM provides a simple way to support provenance management in relational databases. Powerful provenance queries expressed in SQL and XQuery are efficiently supported. We perform experiments on several real-world and synthetic data sets. The results validate the practicality of our approach.

The work presented in this paper is the first system for the provenance management under schema evolution, and leaves many topics open for further studies and improvements which were briefly discussed in the paper.

8 Acknowledgement

The authors would like to thank Mohan Yang, Kai Zeng, and Seyed Hamid Mousavi for their insightful comments and suggestions on this research. This work was supported in part by NSF Grant No. IIS 0917333, entitled "III: Small: Information Systems Under Schema Evolution: Analyzing Change Histories and Management Tools."

References

1. California department of transofrmation-highway conditions. `http://www.dot.ca.gov/hq/roadinfo`.
2. Wikipedia:database download. `http://en.wikipedia.org/wiki/Wikipedia:Database_download`.
3. Bogdan Alexe, Laura Chiticariu, and Wang Chiew Tan. Spider: a schema mapping debugger. In *VLDB*, pages 1179–1182, 2006.
4. Rajendra Bose and James Frew. Lineage retrieval for scientific data processing: a survey. *ACM Comput. Surv.*, 37(1):1–28, March 2005.
5. Peter Buneman, Sanjeev Khanna, and Wang Chiew Tan. Why and where: A characterization of data provenance. In *ICDT*, pages 316–330, 2001.
6. Peter Buneman and Wang Chiew Tan. Provenance in databases. In *SIGMOD Conference*, pages 1171–1173, 2007.
7. Laura Chiticariu and Wang Chiew Tan. Debugging schema mappings with routes. In *VLDB*, pages 79–90, 2006.
8. C. Curino and C. Zaniolo. Pantha rhei benchmark datasets. `http://yellowstone.cs.ucla.edu/schema-evolution/index.php/Benchmark_home`.
9. Carlo Curino, Hyun Jin Moon, Letizia Tanca, and Carlo Zaniolo. Schema evolution in wikipedia - toward a web information system benchmark. In *ICEIS*, pages 323–332, 2008.
10. Carlo A. Curino, Hyun J. Moon, and Carlo Zaniolo. Graceful database schema evolution: the prism workbench. *Proc. VLDB Endow.*, 1(1):761–772, August 2008.
11. Carlo A. Curino, Hyun Jin Moon, Alin Deutsch, and Carlo Zaniolo. Update rewriting and integrity constraint maintenance in a schema evolution support system: Prism++. *Proc. VLDB Endow.*, 4(2):117–128, November 2010.
12. Andrew Eisenberg, Jim Melton, Krishna G. Kulkarni, Jan-Eike Michels, and Fred Zemke. Sql: 2003 has been published. *SIGMOD Record*, 33(1):119–126, 2004.
13. Todd J. Green, Gregory Karvounarakis, and Val Tannen. Provenance semirings. In *PODS*, pages 31–40, 2007.
14. Robert Ikeda, Semih Salihoglu, and Jennifer Widom. Provenance-based refresh in data-oriented workflows. In *CIKM*, pages 1659–1668, 2011.
15. Yogesh L. Simmhan, Beth Plale, and Dennis Gannon. A survey of data provenance in e-science. *SIGMOD Rec.*, 34(3):31–36, September 2005.
16. SQL/XML. `http://www.sqlx.org/`.
17. Divesh Srivastava and Yannis Velegrakis. Intensional associations between data and metadata. In *SIGMOD Conference*, pages 401–412, 2007.
18. Wang Chiew Tan. Provenance in databases: Past, current, and future. *IEEE Data Eng. Bull.*, 30(4):3–12, 2007.
19. Fusheng Wang, Carlo Zaniolo, and Xin Zhou. Archis: an xml-based approach to transaction-time temporal database systems. *The VLDB Journal*, 17(6):1445–1463, November 2008.