Provenance of Data and Metadata in Databases under Schema Evolution

Shi Gao
University of California, Los Angeles

Carlo Zaniolo
University of California, Los Angeles

Abstract

Due to external actions and internal changes combined with schema evolution in databases, an integrated provenance management for data and metadata is required in modern information systems. In this paper, we introduce Archived Metadata and Provenance Manager (AM&PM) system which addresses this requirement by (i) extending the Information Schema [11] with the capability of representing the provenance of the schema and other metadata, (ii) providing a simple timestamp based representation of the provenance of the actual data, and (iii) supporting powerful queries on the provenance of the data and on the history of the metadata.

1 Introduction

The importance of recording the *provenance*, or *lineage*, about information of significance is now widely recognized, and a large body of research has been produced on provenance management in scientific workflows and databases [4, 15, 19]. Existing provenance systems focus on capturing the "why", "where" and "how" facets of provenances [5, 12] and support a rich set of provenance-related functions and queries. Unfortunately, most previous works assume that the database schemas and workflows are fixed and do not change with time.

In reality, modern information systems, particularly big science projects, undergo frequent database schema changes as demonstrated by the UCLA testbed collecting the schema history for 20 other large information systems, including Mediawiki/Wikipedia, Ensembl GeneticDB and various CERN Physics DBs [1]. For instance, the database of Mediawiki software supporting Wikipedia has experienced more than 300 schema versions in its nine years history and similar observations hold for the rest. Very often, when a mistake is found in the latest version of the database, it is hard to trace the provenance of this mistake in early versions since the schema changes block the tracing path.

When the schema evolves, the database under old schema is migrated into the new one conforming to new schema. Therefore, the current database snapshot 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 have then transformed the data as part of the schema evolution process. Thus the history of schema changes since a piece of information was first recorded becomes an integral part of its provenance. A combined provenance management system for data and metadata can be used to meet many important requirements [6, 15], including the following ones:

Provenance Tracing. Users may be interested in the provenance of both data and metadata. The provenance of metadata allows users to audit the process of schema evolution and examine its history.

Provenance Verifying. The combined provenance of data and metadata gives users an effective way to audit suspectable data updates and schema changes.

Source Data Recovery. The source database can be recovered from data and archived versions of database. This can also be used to correct faulty source data.

The Archived Metadata and Provenance Manager (AM&PM) proposed in this paper is the first system designed to address the problem of supporting provenance under schema evolution. AM&PM achieves this goal by (i) extending the SQL information schema facility with timestamp based archival capability to store the provenance of data and metadata, (ii) using Schema Modification Operators (SMOs) and Integrity Constraints Modification Operators (ICMOs) [9, 10] to express the schema mapping between different versions of the database after database upgrades, and (iii) supporting efficiently powerful queries on both data provenance and schema provenance.

2 Background

Schema Evolution Language The Schema Modification Operators (SMOs) were introduced in [10] as a

Table 1: Schema Evolution Language: SMO and ICMO

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 cond
ADD COLUMN d [AS constjfunc(a; b; c)] 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

very effective tool for characterizing schema upgrades and automating the process of migrating the database and upgrading 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 [9] to characterize integrity constraints and automate the upgrading of 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 [1] shows that the combination of SMOs and ICMOs displayed in Table 1 can effectively capture and characterize the schema evolution history of information systems. Here we present a simple example to show how our schema evolution language works. Consider one table *customer* in SALES database, as shown in Figure 1. Underline indicates the primary keys. The initial schema version is V_1. Then the schema evolves into V_2. Assuming that the data type of the attribute *city* in V_2 is the same with the attribute *location* in V_1, the transformation from V_1 to V_2 can be described as:

1. ADD COLUMN *birth* DATETIME INTO *customer*
2. RENAME COLUMN *location* IN *customer* TO *city*

The combination of these two SMOs is *schema evolution script*, which describes the mapping between different versions of database schemas in a concise and unambiguous way. Moreover, it is easy to convert *schema evolution script* into standard SQL scripts, as proved in [10].

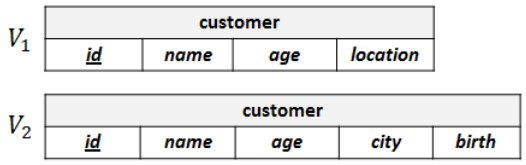

Figure 1: Two schema versions for table *customer*

Temporal Data Model The temporal data model tracks the time when data is valid. The relational schemas in the temporal data model have timestamp attributes to record the valid time intervals of tuples. Two temporal aspects are discussed in past temporal database research [14, 21]: *valid time* and *transaction time*. *Valid time* denotes the time period when entities are valid in the real world. *Transaction time* denotes the time period when entities are valid in the database system. A temporal data model is called *bi-temporal* if both of them are recorded.

In this paper, we focus on the transaction-time model and transaction-time databases. In a transaction-time database, the tuples created or updated are timestamped with the current system time. The deletion of a tuple causes the tuple to be labeled as deleted and its timestamp to be updated. Many transaction-time database implementations are now available, including those of IBM DB2 10, Teradata 13.10, PostgreSQL [18] and ArchIS [20]. These systems provide automatic temporal updates processing and support temporal queries.

Information Schema In relational databases, *Information Schema*, also called *data dictionary* or *system catalog*, is a standard database which archives the metadata of the databases.

For instance, table *COLUMNS* in information schema of MySQL 5.1 is defined with 19 attributes. These attributes include *Table_Schema*, *Table_Name*, *Column_Name*, *Column_Default*, *Data_Type*, and others which are not discussed here since they are not important for provenance. These attributes provide meta information about columns in tables. There are 28 tables in the information schema of MySQL 5.1 to store all kinds of schema structural information of database. In other words, the information schema is the database of other databases and widely supported in the commercial database systems.

3 Approach

In AM&PM, we archive the schema evolution and provenance information in an enhanced information schema, in order to support the queries on the provenance of data and metadata discussed next.

Provenance Query Our approach supports three types of provenance queries:

Data Provenance Query: Given a data tuple d in database D derived from source database S, where does d come from? When is d created? How is d derived from S? All the information about where-, when-, and how-provenance will be stored in the database of AM&PM system. For example, the when-provenance can be provided by the timestamp of relevant tuples.

Schema Provenance Query: Our system allows users to retrieve past versions of database schema along with the SMOs and ICMOs used to upgrade past data and applications. Similarly, we can answer where-, when-, and how- queries for the basic elements of the schemas, such as tables and columns.

Queries on Statistics: AM&PM supports statistical queries about the database content and schema, such as the number of primary keys and schema lifetime.

Metadata and Provenance Model We integrate the data provenance, the schema evolution history and other supporting information (such as database statistics) in one simple model. The provenance of both data and metadata is stored in the relational tables of AM&PM's *provenance database*, where the schema evolution history works as the bridge connecting the provenance of data created under different schema versions.

In order to integrate the data provenance with the schema evolution history, our provenance database enhances the SQL information schema in three aspects:

(i) AM&PM introduces two tables *Transaction* and *Transaction_Text* to archive the history of data updates. The transactions related with data updates are stored in these two tables with transaction time. Besides, for every table in the input database, there is one corresponding *timestamp* table to store the timestamps of the values in the input database.

(ii) Two tables, the SMO table and the ICMO table, are created to store the schema evolution operators (SMOs and ICMOs) generated in the transformation from one schema version to another version. Examples of these two tables are shown in Table 2. *SID* and *ICID* are the primary keys. *Source* and *Target* attributes denote the source schema and result schema. For instance, in Table 2, the tuple s1 in the *SMO* table represents one schema change in the transformation of schema V_1 to V_2.

(iii) AM&PM introduces some auxiliary tables to provide more information about provenance, such as the removed values before data updates and database statistics.

AM&PM provenance database is a point-stamped temporal database. The values are alive from their timestamps to now. For instance, a tuple with timestamp "2006-08-20 12:38:44" is alive from "2006-08-20 12:38:44" to the current timestamp. For transactions and schema changes, timestamp means the time point when the transaction or schema changes go to effect. Any tuple whose valid period overlaps the time point of a transaction is affected by that transaction. Provenance query answering is performed by evaluating all the transactions and schema changes happening in the valid period of queried values.

Table 2: An example of AM&PM provenance database

(a) SMO

SID	SMO_Text	Source	Target	Timestamp
s1	smo_1	V_1	V_2	2006-11-20 12:38:44
s2	smo_2	V_1	V_2	2006-11-20 12:42:07

(b) ICMO

ICID	ICMO_Text	Source	Target	Timestamp
i1	*icmo*	V_2	V_3	2007-02-01 13:04:46

(c) Transaction

PID	DB_User	Schema	Timestamp
p1	Guest1	V_1	2006-09-15 11:09:12
p2	Guest1	V_1	2006-10-16 17:26:09

(d) Transaction_Text

PID	Transaction_Text
p1	$tran_1$
p2	$tran_2$

(e) Customer_timestamp

Tuple_ID	Attribute	Timestamp
100	*id*	2006-09-15 11:09:12
100	*name*	2006-09-15 11:09:12
100	*age*	2006-09-15 11:09:12
100	*city*	2006-10-16 17:26:09
100	*birth*	2006-11-20 12:38:44

smo_1: "ADD COLUMN *birth* DATETIME INTO *customer*"
smo_2: "RENAME COLUMN *location* IN *customer* TO *city* "
icmo: "ALTER TABLE *customer* ADD FOREIGN KEY pk1(*id*) REFERENCE *customer_payment* (*customer_id*)"
$tran_1$: "INSERT INTO *customer* VALUES (100, 'Sam', 30, 'San Francisco')"
$tran_2$: "UPDATE *customer* SET *location* = 'Los Angeles' WHERE *name* = 'Sam' "

Table 2 illustrates how to query provenance under schema evolution. Suppose that there is one table *customer* in our database. The table definition is the same as the example in Figure 1. There is a set of transactions and schema evolution operations. The query is: find the how-provenance of "Los Angeles" in the attribute *city* of table *customer*. The alive period of this value is from "2006-10-16 17:26:09" to now. So we search all the transactions and schema evolution operators whose timestamps overlap this period. Then we get *icmo*, smo_1, smo_2, and $tran_2$. After evaluation, only $tran_2$ and smo_2 affect this value. We return timestamped $tran_2$ and smo_2 as the how-provenance of "Los Angeles".

4 The AM&PM System

Figure 2 depicts the architecture of AM&PM system. An advantage of AM&PM is that the provenance database is constructed on relational database, and thus suitable for any commercial RDBMS. AM&PM consists of two main components: *provenance database* and *query parser*. Provenance database is a combined database of *schema evolution tables* and *data provenance tables*. Schema evolution tables include 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. Query parser rewrites XQuery to SQL and checks the syntax of the provenance queries.

The provenance database is automatically constructed. Schema evolution tables are filled by parsing the past schemas of the database and the schema evolution scripts. Schema evolution script is the set of SMOs and ICMOs used to upgrade database as defined in Section 2. In AM&PM, schema evolution script is produced by a module called *PatchGenerator*. The basic idea of *PatchGenerator* is to compare different schemas and express the changes using schema evolution language. The content of data provenance tables is extracted from the instance of input database and the database history log. The instance of input database provides current data and schema. The database history log offers the source values and relevant transactions.

Once provenance database is filled, users can issue provenance queries expressed in SQL or XQuery. XQuery is introduced to facilitate the expression of complex temporal queries[20]. Provenance queries written in SQL are directly executed in our provenance database. For queries expressed in XQuery, they are rewritten to SQL statements by AM&PM parser. Many proposals have been made to tackle the problem of translating XQuery to SQL. We exploit the algorithm presented in [20], based on the SQL/XML standard [16] which supports the storage of XML documents in relational database.

Extended Functionality Examples of meta-level functions being considered for AM&PM include:

Provenance Query Rewriting Suppose we have a provenance query set, then some changes happen to the schema of current database. Can the current provenance query set be automatically upgraded to work on the new schema? We will take advantage of query rewriting technique in PRISM [9] to solve this problem.

Provenance Propagation If a value in source database is changed, how to propagate this change to current database? This problem is studied in [13] in data warehousing. We plan to extend the algorithm in [13] to schema evolution environment in AM&PM.

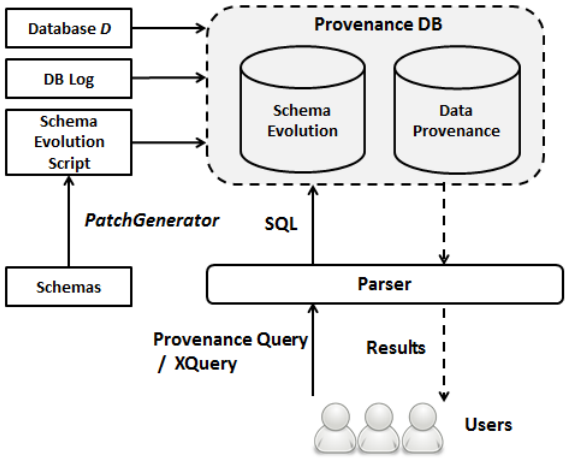

Figure 2: Architecture of AM&PM System

5 Related Work

An overview of database provenance research is presented in [6, 19]. The data provenance in relational views was first studied in [8]. The authors proposed an algorithm to capture the provenance by analyzing the syntax of queries. In [5], the where- and why- provenance are discussed for the data derived by database queries. These works didn't propose a general model for the provenance of data and underlying metadata.

MMS [17] proposes a relational model which uniformly models data and metadata in database. However, the evolution of metadata is not discussed. SPIDER [3, 7] comes closer to our proposal. SPIDER uses nested relational model and mapping language to compute provenance over schema mapping. Our work is different from SPIDER since we archive provenance and schema evolution in a simple relational model and use SQL as query language (also XQuery on temporal views of archived information).

6 Conclusion

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.

The implementation of AM&PM is in progress. We have constructed a provenance database on Mediawiki data set with 323 schema versions. The SQL script for building provenance database and the schema evolution scripts for Mediawiki schemas are available in [2].

References

[1] Pantha rhei benchmark datasets. http://yellowstone.cs.ucla.edu/schema-evolution/index.php/Benchmark_home.

[2] Schema evolution script for the database of mediawki. http://yellowstone.cs.ucla.edu/provenance/.

[3] ALEXE, B., CHITICARIU, L., AND TAN, W.-C. Spider: a schema mapping debugger. In *Proceedings of the 32nd international conference on Very large data bases* (2006), VLDB '06, VLDB Endowment, pp. 1179–1182.

[4] BOSE, R., AND FREW, J. Lineage retrieval for scientific data processing: a survey. *ACM Comput. Surv. 37*, 1 (Mar. 2005), 1–28.

[5] BUNEMAN, P., KHANNA, S., AND TAN, W. C. Why and where: A characterization of data provenance. In *ICDT '01: Proceedings of the 8th International Conference on Database Theory* (London, UK, 2001), Springer-Verlag, pp. 316–330.

[6] BUNEMAN, P., AND TAN, W.-C. Provenance in databases. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data* (New York, NY, USA, 2007), SIGMOD '07, ACM, pp. 1171–1173.

[7] CHITICARIU, L., AND TAN, W.-C. Debugging schema mappings with routes. In *Proceedings of the 32nd international conference on Very large data bases* (2006), VLDB '06, VLDB Endowment, pp. 79–90.

[8] CUI, Y., WIDOM, J., AND WIENER, J. L. Tracing the lineage of view data in a warehousing environment. *ACM Trans. Database Syst. 25*, 2 (June 2000), 179–227.

[9] CURINO, C. A., MOON, H. J., DEUTSCH, A., AND ZANIOLO, C. Update rewriting and integrity constraint maintenance in a schema evolution support system: Prism++. *Proc. VLDB Endow. 4*, 2 (Nov. 2010), 117–128.

[10] CURINO, C. A., MOON, H. J., AND ZANIOLO, C. Graceful database schema evolution: the prism workbench. *Proc. VLDB Endow. 1*, 1 (Aug. 2008), 761–772.

[11] EISENBERG, A., MELTON, J., KULKARNI, K. G., MICHELS, J.-E., AND ZEMKE, F. Sql: 2003 has been published. *SIGMOD Record 33*, 1 (2004), 119–126.

[12] GREEN, T. J., KARVOUNARAKIS, G., AND TANNEN, V. Provenance semirings. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems* (New York, NY, USA, 2007), PODS '07, ACM, pp. 31–40.

[13] IKEDA, R., SALIHOGLU, S., AND WIDOM, J. Provenance-based refresh in data-oriented workflows. In *Proceedings of the 20th ACM international conference on Information and knowledge management* (New York, NY, USA, 2011), CIKM '11, ACM, pp. 1659–1668.

[14] OZSOYOGLU, G., AND SNODGRASS, R. T. Temporal and real-time databases: A survey. *IEEE Transactions on Knowledge and Data Engineering 7* (1995), 513–532.

[15] SIMMHAN, Y. L., PLALE, B., AND GANNON, D. A survey of data provenance in e-science. *SIGMOD Rec. 34*, 3 (Sept. 2005), 31–36.

[16] SQL/XML. http://www.sqlx.org/.

[17] SRIVASTAVA, D., AND VELEGRAKIS, Y. Intensional associations between data and metadata. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data* (New York, NY, USA, 2007), SIGMOD '07, ACM, pp. 401–412.

[18] STONEBRAKER, M. The design of the postgres storage system. In *Proceedings of the 13th International Conference on Very Large Data Bases* (San Francisco, CA, USA, 1987), VLDB '87, Morgan Kaufmann Publishers Inc., pp. 289–300.

[19] TAN, W. C. Provenance in databases: Past, current, and future. *IEEE Data Eng. Bull. 30*, 4 (2007), 3–12.

[20] WANG, F., ZANIOLO, C., AND ZHOU, X. Archis: an xml-based approach to transaction-time temporal database systems. *The VLDB Journal 17*, 6 (Nov. 2008), 1445–1463.

[21] ZANIOLO, C., CERI, S., FALOUTSOS, C., SNODGRASS, R. T., SUBRAHMANIAN, V. S., AND ZICARI, R. *Advanced database systems*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.