MMS: Using Queries As Data Values for Metadata Management

Divesh Srivastava
AT&T Labs–Research
divesh@research.att.com

Yannis Velegrakis
University of Trento
velgias@dit.unitn.it

Abstract

We demonstrate MMS, a system for storing and managing a variety of metadata in a simple, elegant and uniform way. The system is based on two observations. First, that the relational model augmented with queries as data values is a natural way to uniformly model data, arbitrary metadata and their association. Second, that relational queries with a join mechanism augmented to permit matching of query result relations, instead of only atomic values, is an elegant way to uniformly query across data and metadata.

1. The Problem of Metadata Management

Databases are becoming increasingly complex, both in their internal structure (e.g., thousands of tables) and in their interactions with other databases and applications (e.g., mediators and workflows). In successfully understanding, maintaining, querying, integrating and evolving these databases, metadata plays an important role. Metadata is data about data, a secondary piece of information that is separate in some way from the primary piece of information to which it refers. Metadata examples include schema, integrity constraints, comments about the data, ontologies, quality parameters, annotations, provenance information, security policies, or statistical data characteristics. Each such metadata has different structure and semantics. The majority of the proposals that have been made over the years for a metadata management system are mostly extensions of a data model and are tailored to a specific kind of metadata. A simple elegant uniform approach has been elusive.

We demonstrate MMS, a system that allows the storage and querying of different forms of metadata. The demonstration intends to communicate to the database audience a number of important messages [2]. The first is that the relational model is adequate to manage both data and metadata, if it becomes free of specific metadata semantics. No specialized data models are needed, and everything can be modeled through relations. This is the main principle on which MMS has been built. The philosophy is not different from the one followed by the relational model, where at the conceptual level there may be a distinction between entities and relationships, but at the database level, for the purpose of management, everything is represented through relations. The second message the demonstration seeks to communicate is that queries stored as data values [3, 1] in relation attributes is an elegant way to associate data with metadata. Our studies have shown that the main relational model association mechanism, i.e., the join on an atomic value, is limited. Thus, MMS uses queries as data values, referred to as *q-type values* or simply *q-values*. Queries stored in tuple attributes provide an intensional description of a set of records, i.e., the data in the result of the evaluation of the query. This virtual relation, can in turn be used to implement "joins" with other tuples in the database. We refer to this new "join" mechanism as a *q-join*. In MMS, q-join serves as the main linking mechanism between data and its metadata.

2 Demonstration Overview

The top four tables of Figure 1 illustrate a fraction of the database used in the demonstration. The demonstration steps seek to emphasize the numerous advantages that can be gained with the use of queries as data values. In particular:

Metadata Recording with Intensional Associations Assume that a sales analyst runs some data mining tools on the data of the database in order to discover customer trends. In MMS, the analyst would like to annotate the data with the results of the analysis performed by the tools and some comments/observations she makes. For instance, she may want to make the statement that New Jersey customers, i.e., those with loc='NJ', should enjoy a 10% discount in order to boost the sales in New Jersey. The schema does not facilitate the recording of such a statement because this type of information is not part of the main application and alteration of the Customer table may be either not permitted, or may have undesired consequences. The analyst can instead create a separate table Comment (shown in Figure 1) that records her statements and associates entries in that table to the data through q-values, i.e., queries stored as values in a specific column. As another example, assume that the entries in table Item are collected from various data sources and it is important to know the source from where each entry originates.

Customer

cid	cname	loc	tn
1	John	NJ	x7214
2	Nick	NY	x7314
3	Mary	NJ	x6214
4	Kathy	NY	x7994

Order

oid	cid	odate
A	1	09/15
B	2	01/16

Config

oid	iid
A	3
A	6
A	8
B	9

Item

iid	iname	price	warranty
3	CPU	40	N
6	HD	20	Y
8	HD	60	Y
9	HD	20	Y

Comment

qref	txt
select * from Customer where loc='NJ'	apply 10% discount
select * from Customer where cid='1'	high profile customer

Provenance

forD	db	ip
select iname, price from Item where iname='HD'	NJDB	147.52.1.3
select price from Item where iname='CPU'	GDB	211.1.11.73

QualityParams

forP	lastUpd	freq
select ip from Provenance where db='NJDB'	07/06	hourly
select ip from Provenance where db='GDB'	3/29	monthly

Figure 1. A database instance with q-values.

This is achieved by annotating the entries with their provenance information (another form of metadata). This information is recorded in table Provenance as Figure 1 indicates, and associated to the tuples in Item through the q-values of attribute forD.

Assigning Metadata to Value Blocks An important feature the demo will highlight is the ability to assign metadata not only to single values but also to a set of values, even if these values do not constitute a whole tuple. Using a key/foreign key mechanism to associate the provenance information with the entries in the Item table, one can associate a whole Provenance tuple with only a whole Item tuple. MMS is more expressive and this is achieved through the attributes specified in the select clause of the q-values. For instance, the fact that the select clause of the q-value of the second tuple of table Provenance has only the price attribute, it means that the 'GDB' database contributed to the Item table only the price values for the 'CPU' items.

Defining Metadata over Metadata In some cases metadata may have to be defined over existing metadata. We will demonstrate how easily MMS allows the modeling of such situations. The prices of the items change frequently. For that reason, in order to assess the quality of the data, it is important to know how accurate is a price that appears in Item. For that, a data administrator has decided to associate to each data source some quality parameters, such as how frequently it is updated and the time the most recent update has taken place. The same mechanism MMS uses to model metadata on data can also be used to model metadata on metadata, as Figure 1 illustrates through the table QualityParams.

Querying Data and Metadata MMS uses an extension of SQL that allows querying of data and metadata. The extension provides a new powerful operator $\doteq$ that is used to specify conditions on the virtual relations described by the q-values. For example, assume that a user would like to know the conclusions that the analyst has reached regarding the customers in New York and what statements she has made. The following query will return that information:

select p.txt from Comment p
where p.qref$[loc] \doteq [\prime NY\prime]$

Querying Metadata Independently of the Data Many metadata management tools consider the metadata as an integral part of the data, which means that metadata cannot be retrieved without retrieving also the data with which it is associated. This is not the case for MMS. We will show that storing the metadata in independent tables, associated to the data through the q-values, allows them to be queried and retrieved independently. For instance, if a user would like to know the sources that have been used to collect the existing prices of the items, she can simply query the Provenance table alone.

Recording Data Transformations An interesting application of MMS we will demonstrate is its capability to record data transformation processes. Data is usually retrieved from databases, analyzed, processed and then stored back in the same or a different database. During this process there are two issues of great importance. The first is to be able to understand the process through which the data has been generated, and the second is to be able to trace back and find the original data on which the process was applied (provenance). MMS provides a way to automatically record this information and then allow users to visually observe the transformation flow and helps them understand the whole process. This is achieved by recording the transformation information and associating it to the source and transformed data through queries that are stored as data values.

References

[1] D. Gawlick, D. Lenkov, A. Yalamanchi, and L. Chernobrod. Applications for Expression Data in Relational Database System. In *ICDE*, pages 609–620, 2004.

[2] D. Srivastava and Y. Velegrakis. Using Queries to Associate Metadata with Data. In *ICDE*, 2007.

[3] M. Stonebraker, J. Anton, and E. N. Hanson. Extending a Database System with Procedures. *ACM TODS*, 12(3):350–376, 1987.