New SQL Standard in Database Modeling

Eric Pardede
La Trobe University, Australia

J. Wenny Rahayu
La Trobe University, Australia

David Taniar
Monash University, Australia

INTRODUCTION

Relational Database (RDB) is arguably the most widely used repository for database applications. Since the 1970s, we have witnessed the relational data model, from which the RDB is originated, evolving. The progress aims to answer the increasing requirement in database applications. One of them is the requirement to deal with complex structure of real world problems. Unlike its Object-Oriented Database (OODB) counterpart, the RDB, for example, does not have facilities to store large structured objects, semi-structured data, and so forth.

The question might have been answered with the release of new version *Structured Query Language* (SQL) (Fortier, 1999; Melton, Simon, & Gray, 2001). The new SQL has provided many new features including many new data types. Since SQL is used for database definition and manipulation for relational model, the extension has enriched the modeling capability in RDB.

A problem then arises. With the existence of the new data types in the new SQL, the whole database modeling processes have changed. So far, database designers still use the conventional method for RDB design and implementation. This method can create inefficient and incorrect use of the new data types.

This work aims to show how new SQL data types affect the database modeling processes. It highlights new opportunities and new research challenges, brought by the new standard.

BACKGROUND: HISTORY OF SQL

SQL was introduced in 1970 and has emerged as the standard language for RDB (Melton, Simon, & Gray, 2001). The 1992 revision, SQL2, has been widely used by all Relational Database Management System (RDBMS) products. In 1993, an attempt to develop a new standard was started since the RDBMS vendors had enhanced their existing relational products with Object-Oriented (OO) features. The existing standard had become somewhat obsolete because it provided no support for OO features.

Many of the vendors created their own language extension of SQL to retrieve and manipulate data such as POSTQUEL (Stonebraker, 1986), Starburst (Lindsay & Haas, 1990), and so forth. These are vendor-specific languages and still there is no standard that can be used and is acceptable to all vendors. For standardization purposes, a new SQL 1999 was developed.

SQL 1999 has been characterized as "OO SQL" and it has become the foundation for several DBMS such as Oracle8. Ironically, many believe that SQL2 will still be used in the future (Elmasri & Navathe, 2002), since many researchers and practitioners still have unsettled arguments on many SQL 1999 issues.

Further, the standardization body ANSI/ISO has started to review SQL 1999 and aims to release a new SQL4 version in few more years. At the time of writing, this version is still an ongoing work and no release date has been announced (Melton, 2002). SQL4 adds some features to SQL 1999, and it also reviews its previous versions.

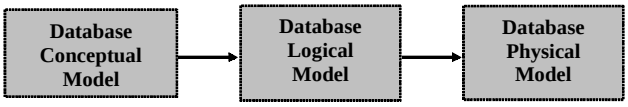
The new SQL has added some new data types to accommodate complex data structures. They are very useful to model real world problems. However, they have a big impact upon the database modeling and implementation.

DATABASE MODELING

Database modeling involves three main phases: conceptual model, logical model, and physical model (see Figure 1).

In the conceptual model level, the database designers capture the database user requirements. To model an RDB, the database designers can use many semantic data modelling such as ER, NIAM, EER, Functional Modeling, and so forth. With SQL4 data type extension, we cannot capture all features using traditional data modeling anymore.

Figure 1. Database modeling phases



The logical model links the conceptual model with the physical implementation. In RDB, this phase is started by transforming the conceptual model into logical design. It is followed by normalization, before we can come up with a set of relations that do not contain anomalies.

In traditional RDB, database designers are already familiar with the transformation methodology (Elmasri & Navathe, 2002), normalization rules (Codd, 1972), and so forth. However, this existing design was developed for simple data types. We do not know whether the old rules can be applicable to the new data types introduced in new SQL4 standard.

The physical model will be different based on how the database is implemented using DBMSs. In this phase, we are required to develop criteria in selecting DBMS. One of the most important criteria is the ability to implement all data type requirements.

For pure RDB that has simple and atomic data type, all DBMS products can be used for the implementation. These products are mainly developed based on SQL2 standard. With the existence of many new data types in SQL4, we need to search the newest DBMS version that can meet the requirement and design.

NEW SQL DATA TYPES

SQL4 classifies the data types into three main classes: predefined, constructed, and user-defined (Melton, 2002). It has few extensions from SQL 1999, but there are a large number of additional data types from SQL2, which is a pure relational model language. Figure 2 illustrates the complete data types supported by current SQL.

Predefined Data Types. Predefined or built-in data types are supported by original SQL. Even though predefined, sometimes the users are still required to determine certain parameters. They are atomic, and therefore very suitable for conventional relational model implementation.

New SQL added Boolean and large object type (BLOB and CLOB). *Boolean* data type enables us to represent the true or false values instead of using a character value with permissible values "T" or "F" (Melton, Simon, & Gray, 2001). *BLOB* can hold a very large binary piece of data such as the digital representation of one's signature. *CLOB* can hold a very large, variably sized, and usually non structured group of characters such as one's resume.

Constructed Data Types. There are two categories of constructed data types: atomic and composite (Melton, 2002; Melton, Simon, & Gray, 2001). From the latest development, SQL4 supports one atomic constructed type and three composite types.

The atomic constructed data type is *REF* type. Its value can be used to address a site holding another value. The site pointed to can be another constructed data type or user-defined type in a typed table.

The first composite data type is *Row*. It contains a sequence of attribute names and their data types. Since it is defined like a flat table, a row type inside a table resembles a nested table. The next data type is *array*, which can hold composite elements of similar type with ordering semantic. Finally, *multiset* contains composite elements that can be duplicated and do not need an

Figure 2. SQL4 data types classification