Set Valued Attributes

Karthikeyan Ramasamy  
Juniper Networks, USA

Prasad M. Deshpande  
IBM Almaden Research Center, USA

INTRODUCTION

About three decades ago, when Codd (1970) invented the relational database model, it took the database world by storm. The enterprises that adapted it early won a large competitive edge. The past two decades have witnessed tremendous growth of relational database systems, and today the relational model is by far the dominant data model and is the foundation for leading DBMS products, including IBM DB2, Informix, Oracle, Sybase, and Microsoft SQL server. Relational databases have become a multibillion-dollar industry.

However, as these databases grew so did the complexity of the data being stored in them with the emergence of a new class of applications. It quickly became apparent that relational databases suffer from various deficiencies and limitations. Relational database systems support a small, fixed collection of data types (e.g., integers, dates, strings) that has been proven to be adequate for traditional applications. With a new class of applications, more complex data needs to be handled. These complex data include hierarchical data of computer-aided design and modeling (CAD/CAM), multi-media data, and documents. Support for this kind of data requires the database to incorporate abstract data types and type constructors based on object-oriented concepts. This leads to the development of object database systems along two distinct paths:

- **Object-Oriented Database Systems**: These systems were developed with the goal of adding persistence to object-oriented languages that support complex types.
- **Object-Relational Database Systems**: These systems are an attempt to extend the relational database systems with the functionality needed to support complex types.

Object relational systems are characterized by:

- **Abstract Data Types**: They represent the ability to add a new data type into the system that is seamlessly treated as equivalent to built-in types.

The definition of a new data type describes the data fields and the methods that operate on these fields.

- **Type Constructors**: They are used to construct new types by composing base or abstract data types. The major classes of type constructors are composites (records), collections, and references. The class of collections can be further divided into sets, bags, arrays, and lists.
- **Inheritance**: It allows the creation of new data types by derivation from existing types.

This article focuses on the set type constructor popularly referred to as set-valued attributes. Set-valued attributes represent a collection of elements of the same type with uniqueness constraint. To illustrate the usefulness of set-valued attributes, consider the following schema where a product item and the availability of its colors need to be described. In a standard relational schema, we need two tables as follows:

```
PRODUCT(Id: integer, Name: string, Manufacturer: string)
COLORS(Id: integer, ColorName: string)
```

This schema is complex and similarly the queries will also be more complex (since it requires a join for the relating product and its colors). An instance of these tables is shown in Figure 1.

In an object-relational database system that supports set-valued attributes, we can describe the same by a single table:

```
PRODUCT(Id: integer, Name: string, Manufacturer: string, Colors: set (string))
```

where the construct set indicates that the Colors attribute is a set of strings. This schema is more intuitive and concise from a data modeling and querying perspective. An instance of this table is shown in Figure 2.

This article examines set-valued attributes, their background, and how they are supported in object-relational database systems.
Set Valued Attributes

Figure 1. An instance of relational schema: PRODUCT and COLORS

<table>
<thead>
<tr>
<th>PRODUCT</th>
<th>COLORS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>Name</td>
</tr>
<tr>
<td>1001</td>
<td>Vacuum Cleaner</td>
</tr>
<tr>
<td>1002</td>
<td>Food Processor</td>
</tr>
<tr>
<td>1003</td>
<td>Juice Blender</td>
</tr>
<tr>
<td>1003</td>
<td></td>
</tr>
<tr>
<td>1004</td>
<td></td>
</tr>
<tr>
<td>1004</td>
<td></td>
</tr>
</tbody>
</table>

BACKGROUND

Research in set-valued attributes has been conducted from different perspectives: data modeling, nested relational databases, object-oriented databases, and object-relational databases.

Data Modeling

Set-valued attributes have long been studied under the context of data modeling, nested relational databases, object-oriented databases, and object-relational databases. Some of the earlier semantic data models (SDM) incorporate sets and collection of entities (Hammer & McLeod, 1981). A further discussion on data modeling using sets can be found in Brodie (1981, 1984). These studies describe how sets can describe the semantic notions of the real world with ease.

Nested Relational Databases

The nested relational model relaxes the assumption that relational attributes are atomic. In order to extend the power of the relational model, Zaniolo (1983) proposes a query language called GEM. GEM adds sets as a data type. Here the sets are viewed as a logical collection, and the set operations of equality and containment are defined as a part of the query language. It also notes that set operations are very expensive to support in standard relational systems. Extension of relational models with set-valued attributes with reference to statistical databases (SDB) has been studied in detail in Ozsoyoglu, Ozsoyoglu, and Matos (1987). The DASDB projects at the Technical University of Darmstadt (Schek & Scholl, 1989) supported the nested relational algebra with nest and unnest operations. The nest operation is used to break up the set instances into individual tuples duplicating other attributes for each set element. The unnest operation combines multiple tuples on a given attribute to form a set when the rest of the attributes values in the tuple match.

OBJECT-ORIENTED DATABASES

From the object-oriented database system perspective, there were two different attempts: One adds persistence to object-oriented languages and the other combines the features of a database system with those of an object-oriented language. Such systems supported many collection types, including sets. Three early projects laid the foundation in this area—Gemstone (Copeland & Maier, 1984) was based on Smalltalk, Vbase (Andrews & Harris, 1987) was based on CLU-like language, and Orion (Banneree et al., 1987) was based on Common LISP Object System (CLOS). New SQL-like languages were designed to support powerful querying for these systems. These query languages allowed nested queries and universal and existential quantification queries.

Object-Relational Databases

Object-relational systems typically start from a relational model and its SQL language and build from there. Early systems supported row types and collection types like sets. The best-known research implementations of object-relational database systems are POSTGRES (Stonebraker, 1997; Stonebraker & Kemnitz, 1991) from the University of California, Berkeley and Paradise (Patel et al., 1997) from the University of Wisconsin. POSTGRES supported the dynamic addition of new types, support for complex objects including sets, inheritance, and rules support. Paradise departs from POSTGRES in that it is a parallel object-relational database system. The main contribution is to explore the parallelization of object-relational features in a shared-nothing environment.

SET-VALUED ATTRIBUTES

In order to incorporate full support for set-valued attributes into object-relational database systems, the following issues must be addressed:
Related Content

An Integrated Query Relaxation Approach Adopting Data Abstraction and Fuzzy Relation
www.igi-global.com/article/integrated-query-relaxation-approach-adopting/47419?camid=4v1a

Discovering Quality Knowledge from Relational Databases
M. Mehdi Owrang O. (2009). Selected Readings on Database Technologies and Applications (pp. 95-111).
www.igi-global.com/chapter/discovering-quality-knowledge-relational-databases/28548?camid=4v1a

Fuzzy Information Modeling with the UML
www.igi-global.com/chapter/fuzzy-information-modeling-uml/4810?camid=4v1a

Cost and Service Capability Considerations on the Intention to Adopt Application Service Provision Services
www.igi-global.com/chapter/cost-service-capability-considerations-intention/63671?camid=4v1a