Enumerated Attributes for Relational Databases

Phyllis Jones and Joel Jones
Department of Computer Science
University of Alabama
pecj@cs.ua.edu jones@cs.ua.edu

Aliases

Add-a-bead

Problem

How do we enforce that relational database attribute values are consistent, limited, and extensible?

Context

The relational database design contains attributes with a limited subset of values. Furthermore, this subset may expand over time.

Forces

- It is desirable to enforce consistent entries.
- It is desirable to enforce only a limited choice of values.
- It should be easy to add new values to the limited choice list.
- It is desirable that permissible values can be added without changing code.
- We want to keep our relational database design as normalized as possible.

Solution

Create a table for each set of permissible attribute values. This table should have a primary key which is an integer or other short encoding. It should also have a column for the full description. Create a foreign key constraint on every table that contains an attribute that references the extensible attribute [extent table].

Extend the set of permissible values by adding a row to the extent table.

Resulting Context

The application of the add-a-bead pattern implies that more attention must be placed on performance when writing queries and reports. Anything requiring the full description necessitates joining the enumeration table to the primary table. Reports require more complex joins to retrieve the detailed values, but are a reasonable tradeoff for conformity of values. With the use of foreign key indexes, large caches, and faster processors this should be efficient even in large databases.

Rationale

When writing reports and other automated queries, without this pattern it is difficult to get accurate statistics because of value misspellings and inconsistencies. User interfaces written without this pattern, but still requiring validation, must have the limited values coded into them. Then, whenever the enumerated values must be extended, the source code must be modified and redeployed. Also, if database triggers are used for validation, these also must change. By placing the enumerated values in one place, the enumerated values table, extending the enumerated values requires change in only one location.

Alternatively, in a single table design, a user interface may extract the values from the table and present them in a drop-down list with an additional free text entry alternative. This single table design necessitates an extraction of distinct values over a large table, while the add-a-bead pattern allows a simple query over a much smaller separate table. Further, presentation ordering can be dictated in the separate table, improving usability.

For applications that do not present a user interface, the foreign key constraints between the main table and the enumeration table also enforce consistency. The primary key of the enumeration table is a numeric index or short character abbreviation.

There are several other advantages to this pattern. Join queries using foreign key constraints are more efficient than joining lengthy description fields. The performance enhancement depends on the number of rows in the adjoining database table. From a program understanding view point, developers familiar with this pattern can quickly categorize tables as enumeration tables or major tables. The enumeration tables contain detailed data values of interest for field-level entry. The dominant tables contain the main data of interest, non-categorical data, and foreign key values.

Known Uses

This pattern is used in most normalized relational databases with lengthy text description fields. Additionally, it is used where data entry applications and their back-end database are closely correlated such that the drop-down boxes for the data entry screen are tables in the database. We rediscovered it in state crash report data while converting

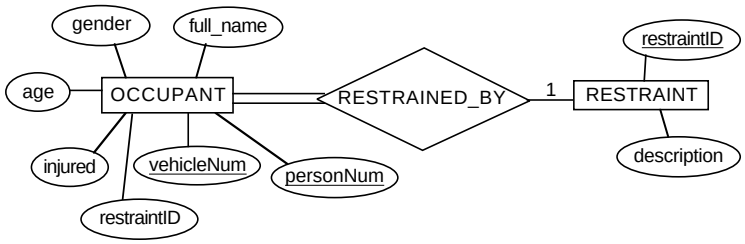

Figure 1: ER diagram

it to CARE's data format. CARE is a data-analysis tool of the University of Alabama's CRDL laboratory [3].

Related Patterns

The Extensible Attributes [1] pattern of Kent Beck has a similar intent. In his pattern, during prototyping, objects keep their attributes in a Dictionary and provide functions for accessing them. This enables the code to rapidly evolve until the design becomes more stable. The surviving attributes are then changed to reside in instance variables, rather than in a Dictionary. The intent of add-a-bead is slightly different, as it is intended that items can only be added to the set of attributes. Otherwise, legacy data would lose its internal consistency.

For other patterns related to data verification, see Ward Cunningham's Checks pattern language [2].

Sketch

In Table 1, we see a relational database table to which this pattern could usefully be applied. This is derived from a database used to contain information about automobile accidents.

An ER diagram can be seen in Figure 1. The entity OCCUPANT has total participation in the RESTRAINED_BY relationship, as every occupant is restrained, even if by the restraint "none." The entity RESTRAINT does not have total participation in the RESTRAINED_BY relationship, as new restraint devices (e.g. "booster seats") may be added before any accident report has been filed with an occupant restrained by that device. Every occupant is restrained by only one restraining device. The occupants are uniquely identified by the vehicle number and person number in that vehicle.

When an accident report is taken, one piece of information that many states gather is what kind of restraint device was being used by the drivers and passengers of the automobiles involved in the accident. The restraint column records this information, but because the add-a-bead pattern wasn't used, there are many inconsistencies. These incon-

Table 1: OCCUPANT_FLATTENED

vehicleNum	personNum	full_name	gender	age	injured	restraint
1	1	John Smith	M	35	N	seat belt and shoulder belt
1	2	Jane Doe	F	33	Y	shoulder harness
1	3	Bob Smith	M	8	Y	seat belt
1	4	Don Smith	M	3	N	car seat
2	1	Mary Miller	F	19	Y	none
2	2	Ann Miller	F	15	Y	seet belt
3	1	Tom Jones	M	28	N	sholder belt
3	2	Mark Ford	M	27	Y	seat and sholdar belt
4	1	Julie Johnson	F	27	N	belt
4	2	Joe Johnson	M	10	Y	nun
4	3	Jack Johnson	M	11	Y	no

Table 2: RESTRAINT

restraintID	description
1	none
2	lap belt
3	lap and shoulder belt
4	shoulder belt
5	car seat

sistencies include differences in terminology, such as “shoulder harness” vs. “shoulder belt.”and misspelling, such as “seet belt.”

In Tables 2 and 3, we see the application of this pattern to a similar design. There are several things to note. First, the add-a-bead pattern is applied to the restraint column by introducing a numeric foreign key between Table 3 and Table 2. Second, we note that the ability to extend the set of valid restraint devices to include “booster seats” is easy and can be accomplished by adding a row to Table 2. Note that the key of any existing row should not be changed.

Third, the pattern was not applied to either the gender column, the age column, or the injured column. The pattern is not appropriate here, as the encodings used are small and are very unlikely to change. Additionally, an age constraint such as age is between zero and one hundred can easily be logically enforced in the program or as a trigger in the database.

To mimic a simple dump of the flattened table, use the following SQL query:

```
SELECT vehicleNum, full_name, gender, age, injured, r.description
  FROM OCCUPANT o, RESTRAINT r
 WHERE o.restraintID = r.restraintID
```

Table 3: OCCUPANT

vehicleNum	personNum	full_name	gender	age	injured	restraintID
1	1	John Smith	M	35	N	3
1	2	Jane Doe	F	33	Y	4
1	3	Bob Smith	M	8	Y	2
1	4	Don Smith	M	3	N	5
2	1	Mary Miller	F	19	Y	1
2	2	Ann Miller	F	15	Y	2

References

[1] Kent Beck. *Smalltalk Best Practice Patterns*. Prentice Hall, 1997.

[2] Ward Cunningham. The CHECKS pattern language of information integrity. In James O. Coplien and Douglas C. Schmidt, editors, *Pattern Languages of Program Design*, pages 145–156. Addison-Wesley, 1995.

[3] *CARE* Research and Development Laboratory. *CARE* traffic safety data analysis page. http://care.cs.ua.edu/, 2003.