METADATA FOR BUSINESS RULES INTEGRATION WITH DATABASE SCHEMA

Rimantas Butleris
Department of Information Systems, Kaunas University of Technology
Studentu st. 50, LT-51368, Kaunas, Lithuania

Liudas Motiejunas
Department of Information Systems, Kaunas University of Technology
Studentu st. 50, LT-51368, Kaunas, Lithuania

ABSTRACT
In this article main business rules system components (business rules repository and business rules engine) are discussed. The business rules repository is a database that stores all the data about the business rules and all the necessary metadata about entities, attributes and relationships. This kind of the metadata is analysed. Two different types of business rules looking from the perspective of calling them from the business rules repository are discussed. Before evaluating business rules we must collect metadata about tables, columns and relationships that particular business rule is involved in. Steps of gathering such data are discussed and graphical schema is presented. Events that activate business rules and steps of their managing are discussed.

KEYWORDS
Business rules, business rules engine, metadata.

1. INTRODUCTION

There are several business rules based Information system (IS) architecture models (Halle, 2001, Molay, 2003). The number and purpose of the components can vary in different business rules systems, but there are some basic components that are vital for business rules system functionality.

The core component of the business rules system is the business rules repository. This is where business rules are kept. But it is not enough to keep only business rules there. For efficient management of the business rules there must be additional metadata about the logical database structure. This metadata is needed because the business rules engine must identify the tables, columns and relationships are involved in the business rule. Keeping information about the database structure in business rules repository lets the business rules engine navigate through database easily and efficiently.

The rest of the paper is organised as follows: section 2 overviews business rules engine and business rules repository of the business rules system; in section 3 the data stored in a business rules repository is discussed; in section 4 two different types of business rules looking from the perspective of calling them from the business rules repository are analysed; in section 5 steps for collecting metadata from business rules repository are discussed and finally in section 6 main steps of business rules execution are presented.

2. MAIN COMPONENTS OF THE BUSINESS RULES SYSTEM

2.1 Business rules repository

Physically, rule repository is an autonomic business rule collection which can be altered at any time using
relatively easy tools (Butleris & Kapocius, 2002). These are two solutions to storing rules:

1. Parameter driven approach. In this case rules are stored in the database where they are characterized by the values of various attributes. It has been shown by different researchers, that rule repository could be designed as an independent database (Plotkin, 1999) or as a part of the main logical model (Perkins, 2002). However, the first solution offers more flexibility and more options for the storage of complicated business rules.

2. Independent process-driven approach. This approach is similar to the traditional methodologies where rules are implemented directly in the program code, only in this case the code, representing rules, is stored independently from other layers of the IS and therefore rules are expressed only once in the system.

Business rules repository is a database that stores all the data about business rules and all the necessary metadata about entities, attributes and relationships that are included in the data model. Probably it is impossible to create an universally accepted business rules repository structure, because various organizations or business rules researchers define different types of business rules and ways that the rules are described (Kardasis & Loucopoulos, 2004; Business rules group, 1999; Ross, 1997). In our case we use the first solution to store business rules.

However, there are several attributes that must be stored in the business rules repository for quick and effective business rules execution. These attributes are described in section 3.

2.2 Business rules engine

The system-wide enforcement of stored rules is managed by the special rule interpretation mechanism called business rules engine. Such an engine is considered as a monolithic mechanism; however, as it has been shown in (Mariano et al, 2001), the task of enforcing or implementing rules can be carried out by more or less independent services.

The business rules engine calls business rules from business rules repository and performs actions described by the rule. Business rules engine can be implemented in various ways (Ross, 2003; Wilson, 2003), the same as business rules repository, because its architecture depends on business rules repository (the form that business rules are stored in it). In this paper we are talking about business rules engine that primarily deals with databases. Business rules engine “must know” what business data it has to deal with during a business rule execution. That is why we must keep additional data about tables, attributes and relationships existing in the database in the business rules repository.

It doesn’t matter in which way the business rules engine is implemented - it must ensure that business rules are performed correctly.

3. DATA STORED IN A BUSINESS RULES REPOSITORY

Business rules are based on business data that is directly used in business operations and would be used even in the absence of computerized system. Metadata is additional data that describes what these computerized systems contain and how they work, or describes the business data, such as definitions of business terms. A large component of the metadata that system designers work with is the metadata that describes the structure of an application database. This kind of data is captured in data models. We can divide the business rules repository in two major parts – metadata about database and business rules themselves. This separation doesn’t mean that metadata about the database and business rules are kept separately; it only shows that there are two major parts that are different according their meaning. In some DBMS (database management system) there is already metadata about entities, attributes, relationships, etc., but in different DBMS there is different metadata or at least it has a different structure. Therefore we need to extract that metadata from the original database and keep it in the business rules repository in a common form.

3.1 Metadata about database tables

The database is made from tables that contain columns and relationships between them. We have to keep some information about its structure, because the business rules engine has to identify what table’s attributes it has to deal with.
Business rules repository is a database itself. At first, there must be a table in the business rules repository in which we keep data associated with the database tables. This table is named Table. Each table in the application’s database has its own unique physical name that should be kept in the business repository because the business rules engine is primarily working with database tables. As this name is unique it can be used as the primary key in the business rules repository’s entity Table. That name we store in the attribute named Table_name. The second attribute is Entity_definition, which is a business definition of the entity. The entity definitions are important for analysts, programmers and business users for understanding what this entity is really for. Structure of the entity Table is shown in figure 1.

We do not need to enter metadata into the table manually. We can extract metadata from the data-modelling tool. There are many different data modelling tools and any of them can be a source for the data which has to be entered into the table.

3.2 Metadata about columns

In logical database schema tables represent the entities and columns represent the attributes. The business rules repository must be able to contain metadata about columns that form tables in the database. For that purpose we must design a table in the business rules repository which will hold that metadata. This table is named Column. Further, attributes that are kept in this table are discussed.

The first one is Column_ID. Sometimes there can be attributes with the same name, but with different meaning in different database tables. For example, there can be the Customer table with the attribute Last_name and the Employee table with the same attribute. Because of that we need to have a unique column identifier, which can be a generated code.

It is very important for the business rules engine to identify what physical columns it has to operate with. Physical column names are then stored in attribute named Column_name.

The same as with metadata about tables, there must be an attribute which contains attribute definition (attribute_definition). It can help users or developers to understand the meaning of a particular attribute.

The business rules engine must identify which columns of a table are primary keys. For example, this is needed when data is moved between tables. This information is kept in the attribute Column_PK. It is the Boolean value that is TRUE if a column is a primary key and is FALSE if it isn’t.

The business rules engine also must identify whether a column is a foreign key. This information is kept in the attribute Column_FK and it is Boolean value that is TRUE if a column is a foreign key and is FALSE if it isn’t.

Every column is found in one table. It is necessary to identify this table. This we can do by adding Table_name from an entity Table. Structure of the Column entity is shown in figure 2.
The same like with metadata about tables, we can extract metadata about columns from the data-modelling tool.

### 3.3 Metadata about relationships

Relationships are the third component of data model that the business rules engine has to deal with. Entity about relationships has to be added to the business rules repository static structure model. There can be a relationship between two and only two entities – a parent entity and a child entity. However, there can be more than one relationship between the same two entities. Because of that we need to have a unique key that identifies a relationship. For that purpose the `Relationship_ID` attribute was created.

Of course we must have parent and child tables names of the relationship. We store this information in the two attributes `Parent_table` and `Child_table`.

For better understanding of the relationship we add an attribute named `Relation_definition`.

When the primary key of the parent table has migrated into the child entity, this attribute does not always have the same name. The business rules engine has to know how these attributes are named in parent and child entities. To manage this problem there is a new entity called `Relationship_column` in which we store this information. This entity has a primary key of `Relationship_ID` and `Parent_column_ID`. `Parent_column_ID` is the `Column_ID` of the parent column that migrated from the parent entity to the child entity.

Attribute `Child_column_ID` is the `Column_ID` of the corresponding column in the child table. Structure of the `Relationship` and the `Relationship_column` entities is shown in figure 3.

![Figure 3. Structure of the Relationship and Relationship_column entities](image)

The same like with metadata about tables and columns, we can extract metadata about columns from the data-modelling tool. The unique attribute `Relationship_ID` can be generated automatically.

In previous chapters we have discussed metadata about applications database that must be kept in business rules repository. Another component of the business rules repository is the business rules themselves. There are a lot of different ways of structuring the business rules, but that is out of scope of this article. Further we discuss general business rules management ideas based on the business rules firing.
4. EXAMPLE OF THE BUSINESS RULE METADATA

The following example shows simple business rule and related metadata that is kept in business rules repository. Business rule: Each customer must have address. 
Customer.Address <> Null.

One table is involved into this business rule. Metadata about this table is shown in table 1.

Table 1. Metadata about Customer table

<table>
<thead>
<tr>
<th>Table name</th>
<th>Entity description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer</td>
<td>This table contains data about customers</td>
</tr>
</tbody>
</table>

Also one attribute is involved in this business rule. Metadata about attribute address is shown in table 2.

Table 2. Metadata about attribute Address

<table>
<thead>
<tr>
<th>Column_ID</th>
<th>Column_name</th>
<th>Attribute description</th>
<th>Column_PK</th>
<th>Column_FK</th>
<th>Table_name</th>
<th>FK_parent_table</th>
<th>FK_parent_column_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addr1</td>
<td>Address</td>
<td>…</td>
<td>FALSE</td>
<td>FALSE</td>
<td>Customer</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

With this metadata about database structure business rules engine can correctly interpret business rule.

5. CALLING BUSINESS RULES

As business rules are kept in the business rules repository, separate from the program code, they are declarative and implicate no control logic. They have to be called and executed by a special component – the business rules engine. Every rule rejects, produces or projects some type of actions or data (Ross, 2003).

Also, each rule is associated with a particular data. Until the user does not take any action, business rules are not called from the business rules repository. But when some action occurs the business rules engine must verify the business rules and evaluate that action.

By the nature of the business rules calling from the business rules repository we can divide them in the two major types: data control business rules and business rules for the processing of specific events (Motiejunas&Butleris, 2004). These two types of the business rules can be classified further, but we do not discuss that, because these subtypes don’t make difference for business rules activating.

Generally, the business rules engine starts working on the three basic events – INSERT, DELETE and UPDATE. The business rules that respond these events are similar to data base triggers, but the business rules are much more than just triggers (Date, 2000). Getting the trigger code correct can be very difficult, and implementation limitations often get in the way, too – the fact remains that in general triggers still mean that someone, somewhere, is writing a lot of procedural code. Either way when the user makes an attempt to insert, delete or update data, the business rules engine must fire business rules that are associated with the data that the user wants to change. It is like a monitoring process – the business rules engine is inactive and comes to action only when some changes in data are noticed. Business rules of this type must be fired at least on two separate events (Ross, 2003). This doesn’t mean that business rules must be fired when two events occur at the same time. This means that business rule must be fired when one event occurs, but the same rule must be fired in another situation considering actions that the user has made. For example, let us take the following business rule: Order must have an assigned customer. When the user attempts to insert new Order record, this rule must be validated to ensure that the Order has assigned Customer. But there is another situation when the same rule must be fired again: when the user tries to delete Customer record, the business rule engine must validate if none of the Order records is violated. So the same business rule is fired on two separate events. A general schema of the business rule, which is called on INSERT, DELETE and UPDATE events, is shown in figure 4.
As we can see from Figure 4 each rule must be fired at least at two events; of course there can be more events that can violate the rule.

Business rules that do not control data are fired in other situations, not only when the user attempts to change data. This kind of business rules can create data themselves. As an example, the business rules of DERIVATION or CALCULATION type could be pointed out. The result of such business rules is the derivative data that can be stored in a file, shown on the screen or printed in a report. These rules usually are called by specific events, which can depend on user's actions (button click) or simply on the timer (the last day of the month). The business rules of the second type do not have to be fired at least on two events, because they do not ensure the consistency of data, they can create data themselves (Figure 5).

If data, created by such a rule, is inserted in a database table, then the rule of the first type must activate and verify whether those data meet database requirements.

6. COLLECTING RELATED METADATA

As it was mentioned above, business rules can be of two general types – those, which secure the integrity of data and those, which are not directly associated with the data control and can create derivative data. Before executing business rules of any type, the business rules engine has to collect metadata from the business rules repository about tables, columns and relationships that a particular business rule is involved to. Main schema for collecting metadata is shown in Figure 6.
Every business rule is related with at least one table from the database. The first step is to collect information about these tables. When we have information about tables the business rules engine can collect data about columns that are related to a business rule.

At first the business rules engine must gather information stored in Column_ID and Column_name attributes. Then it must check if the column is a primary key, a foreign key or none. This can be done by checking the values of the attributes Column_PK and Column_FK. If a column is a foreign key then the business rules engine must know what the parent table is the foreign key has came from. This information is kept in the attribute named FK_parent_table. If a column is a foreign key, then there must be a corresponding column in the parent table. This information is kept in the attribute FK_parent_column_ID.

Collecting information about relationships is optional because one business rule can deal with columns only in one table and there is no relationship with other tables.

When the business rules engine has all information related to the business rule, business rule can be executed. Steps for managing the business rule are presented in following section.

7. EXECUTION OF THE BUSINESS RULES

As it was mentioned above, business rules can be of two general types – those which secure the integrity of data and those which are not directly associated with the data control and can create derivative data. Rules of the first type can be violated and must be fired when the user attempts to change data. Then the business rules engine has to take appropriate actions and to ensure the consistency and the integrity of the data. Actions are as follows (Motiejunas & Butleris, 2003):

1) When an appropriate event DELETE or UPDATE occurs, then the business rules engine makes a copy of data (if an event INSERT has occurred a copy of the data is not made) that the user tries to change (it can be a single field, a record or some other data structure).
2) Data changes that were indicated by the user are performed (for example, the user has updated one field in a record).
3) The business rules engine fires the rule that is associated with that field and performs a test, restriction or the same action as it is defined in that rule.
4) Depending on whether the rule was violated or not, the following actions are initiated:
   a) If conditions described in the rule were satisfied, then the actions defined by the user are accepted (in this case the updated field is accepted) and no message is shown.
   b) If conditions described in the rule were violated, then the actions defined by the user are rejected; the initial data, using a copy that was made in step 1, is restored and an error message is shown to the user.

The business rules of the second type are managed much more simply. Because they create new data by data kept in the database or perform selection of data and are not related to the data control, managing them does not require a lot of actions. When some specific action is performed or appropriate circumstances occur, then the business rule is simply fired (as it is shown in figure 7) and actions described in the business rule are performed. However, rules of the first and the second type can be associated with the same data. For that reason, we can use specific labels to indicate the type a rule belongs to and there is no a reason to perform rules of the second type, when the data are changed.

8. CONCLUSION

In this paper the main metadata about tables, columns and relationships that the business rules repository must contain is presented. In the business rules repository this kind of data is essential because the business rules engine must identify with what business data it must deal during a business rule execution. Every business rule is related to data that is stored in a database. When metadata about database structure is stored in the business rules repository closely with business rules, it is easier to execute the business rules and perform actions described by the rules. For that purpose Table, Column, Relationship, Relationship_column entities have been created in order to store the necessary information. At first the business rules engine must collect metadata about tables and columns that a particular business rule is involved to. Collecting
information about relationships is optional because one business rule can deal with columns only in one table and there is no relationship with other tables.

In this paper business rules are divided in two major types (data control business rules and business rules for specific events processing) by their activating from the business rules repository. This separation can make business rules managing easier, because the second rule type is processed very simply.

If the rule was related to the data, the business rules engine must make a copy of the data (if an event INSERT has occurred, a copy of the data is not made), perform data changes and fire the rule. If the rule was violated, then the action is rejected and the initial data are restored. In the other case data changes are accepted.

If the rule is activated on specific events, it is simply fired and actions described in the business rule are performed.

On the basis of this article the next step is to create the business rules repository architecture for storing business rules and metadata together.

REFERENCES