ISSN No. 0976-5697

Volume 3, No. 3, May-June 2012

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Business Rules in DBMS

Abhijeet Raipurkar*
Computer Science & Engineering Department
PRMIT&R Badnera,India
abhi741@gmail.com

Gajanan Deokate
Computer Science & Engineering Department
PRMIT&R Badnera,India
deokate_gajanan@rediffmail.com

Abstract: Business rule is very important to the modeling and definition of information systems. Business rules are implemented as triggers in relational databases. The rule expresses a policy that how organization carries out a task. Business rules represent information about real world and database is collection of related information. Also people are interested in the set of rules that determines operation of business. Data models in DBMS give the structure of data; business rules are sometimes used to tell how the data can and should be used. There are various types data models supported in DBMS like network model, Hierarchical model and relational model. There are various business rules in DBMS like domain rules, integrity rules and triggering operations. In relational database business rules can be implemented with the help of checks, assertions and triggers. Eliciting, expressing, and capturing business rules is most important in DBMS which makes live databases more strong.

Keywords: Network model, Hierarchical Model, Trigger, Assertion, Domain rule, Integrity Rule

I. BACKGROUND OF PRESENT SYSTEM

The term *business rule* is an informal, intuitive one. It is used with different (but overlapping) meanings in different contexts. GUIDE (2000) defined business rule as follows: "A business rule is a statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behavior of the business." Thus, the rule expresses a policy (explicitly stated or implicitly followed) about some aspect of how the organization carries out its tasks [1]. This definition is very open and includes things outside the realm of information systems (e.g., a policy on smoking). a business rule in this sense usually specifies issues concerning some particular data item (e.g., what fact it is supposed to express, the way it is handled, its relationship to other data items).

In any case, the business rule represents information about the real world, and that information makes sense from a business point of view. That is, people are interested in "the set of rules that determine how a business operates". Note that while data models give the structure of data, business rules are sometimes used to tell how the data can and should be used. That is, the data model tends to be static, and rules tend to be dynamic [2]. Business rules are very versatile; they allow expression of many different types of actions to be taken. Because some types of actions are very frequent and make sense in many environments, they are routinely expressed in rules.

II. INTRODUCTION

Though informal, the concept of business rule is very important to the modeling and definition of information systems. Business rules are used to express many different aspects of the representation, manipulation and processing of data. However, perhaps due to its informal nature, business rules have been the subject of a limited body of research in academia.

There is little agreement on the exact definition of business rule, on how to capture business rules in requirements specification (the most common conceptual models, entity relationship and UML [3], have no proviso for capturing business rules), and, if captured at all, on how to express rules in database systems. Usually, business rules [1] are implemented as triggers in relational databases. However, the concept of business rule is more versatile and may require the use of other tools.

III. TYPES OF BUSINESS RULES IN DBMS

Domain Rules:

Rules that constrain the values that an attribute may take. Note that domain rules, although seemingly trivial, are extremely important. Domain rules allows

a. Verification that values for an attribute make "business sense"
b. Decision about whether two occurrences of the same value in different attributes denote the same real-world value
c. Decision about whether comparison of two values makes sense for joins and other operations.

For instance, EMPLOYEE.AGE and PAYROLL. CLERK-NUMBER may both be integer but they are unrelated. However, EMPLOYEE.HIRE-DATE and PAYROLL. DATEPAID are both dates and can be compared to make sure employees do not receive checks prior to their hire date.

A. Integrity Rules:

Rules that bind together the existence or nonexistence of values in attributes. This comes down to the normal referential integrity of the relational model and specification of primary keys and foreign keys.

CONFERENCE PAPER
National Conference on Information and Communication Technology for Development
Organized by PRMITR, Amravati (MS) India
http://mitra.ac.in/forthcoming.html

B. Triggering Operations:

Rules that govern the effects of insert, delete, update operations in different attributes or different entities. This division is tightly tied to the relational data model, which limits its utility in a general setting.

C. Structural Assertions:

Something relevant to the business exists or is in relation to something else. Usually expressed by terms in a language, they refer to facts (things that are known, or how known things relate to each other). A term is a phrase that references a specific business concept. Business terms are independent of representation; however, the same term may have different meanings in different contexts (e.g., different industries, organizations, line of business). Facts, on the other hand, state relationships between concepts or attributes (properties) of a concept

(Attribute facts), specify that an object falls within the scope of another concept (generalization facts), or describe an interaction between concepts (participation facts). Facts should be built from the available terms. There are two main types of structural assertions:

a. Definition of business terms:

The terms appearing on a rule are common terms and business terms. The most important difference between the two is that a business term can be defined using facts and other terms, and the common term can be considered understood. Hence, a business rule captures the business term's specific meaning of a context.

b. Facts relating term to each other:

Facts are relationships among terms; some of the relationships among terms expressed are (a) being an attribute of, (b) being a subterm of, (c) being an aggregation of, (d) playing the role of, or (e) a having a semantic relationship. Terms may appear in more than one fact; they play a role in each fact.

c. Constraints (also called action assertions):

Constraints are used to express dynamic aspects. Because they impose constraints, they usually involve directives such as "must (not)" or "should (not)." An action assertion has an anchor object (which may be any business rule), of which it is a property. They may express conditions, integrity constraints, or authorizations.

d. Derivations:

Derivations express procedures to compute or infer new facts from known facts. They can express either a mathematical calculation or a logical derivation. There are other classifications of business rules [3], but the ones shown give a good idea of the general flavor of such classifications.

IV. BUSINESS RULES AND DATA MODELS

There are two main problems in capturing and expressing business rules in database systems. First, most conceptual models do not have means to capture many of them. Second, even if they are somehow captured and given to a database designer, most data models are simply not able to handle business rules. Both problems are discussed in the following sections. Focusing on the relational model, it is clear that domain constraints and referential integrity constraints can be expressed in the model. However, other rules (expressing actions, etc.) are highly problematic. The tools that the relational model offers for the task are checks, assertions, and triggers.

V. BUSINESS RULES IN DBMS

Checks are constraints over a single attribute on a table. Checks are written by giving a condition, which is similar to a condition in a WHERE clause of an SQL query.

As such, the mechanism is very powerful, as the condition can contain subqueries and therefore mention other attributes in the same or different tables. However, not all systems fully implement the CHECK mechanism; many limit the condition to refer only to the table in which the attribute resides. Checks are normally used in the context of domains (Melton and Simon, 2002); in this context, a check is used to constrain the values of an attribute.

For instance,

CREATE DOMAIN age int CHECK (Value >= 18 and Value <= 65) Creates a domain named age, expressed as an integer; only values between 18 and 65 are allowed. Checks can be used inside a CREATE TABLE statement, too. Checks are tested by the system when a tuple is inserted into a table (if the check was given within a CREATE TABLE Statement, or if the check was given within a CREATE DOMAIN and the table schema uses the domain), or when a tuple is updated in the table. If the condition in the check is not satisfied, the insertion or update is rejected. However, it is very important to realize that when check conditions involve attributes in other tables, changes to those attributes do not make the system test the check; therefore, such checks can be violated. For instance, the following declaration

```
CREATE TABLE dept (
Dno int,
Mgrname VARCHAR(50),
Check (Mgrname NOT IN (Select name from employee
where salary < 50,000))
```

would block insertions into table dept (or updates of existing tuples) if someone introduced in the tuple a manager that made less than 50,000. However, if someone updated the table emp to change a manager's salary to less than 50,000, the update would not be rejected.

Assertions are constraints associated with a table as a whole or with two or more tables. Like checks, assertions use conditions similar to conditions in WHERE clauses; but the condition may now refer to several existing tables in the database. Note that, unlike checks, which are associated with domains or attributes in tables, assertions exist at the database level and are independent of any table. Assertions are checked by the system whenever there is an insertion or update in any of the tables mentioned in the condition. Any transaction must keep the assertion true; otherwise, it is rejected. This makes assertions quite powerful;

CONFERENCE PAPER
National Conference on Information and Communication Technology for Development
Organized by PRMITR, Amravati (MS) India
http://mitra.ac.in/forthcoming.html

unfortunately, assertions are not fully or partially supported by many commercial systems. Triggers are expressions of the form E-C-A, where E is called an event, C is called a condition, and A is called an action. An event is an occurrence of a situation. An event is instantaneous and atomic (i.e., it cannot be further subdivided, and it happens at once or does not happen at all). The typical events considered by active rules are primitive for database state changes (e.g., an insert, delete, or update). A condition is a database predicate; the result of such a condition is a Boolean (TRUE/FALSE). Action is a data manipulation program, including insertions, deletions, updates, transactional commands, or external procedures. In order to allow the action to process data according to the event,most systems use the variables OLD and NEW to refer to data in the database before and after the event takes place. If a row is updated, both OLD and NEW variables have values. If a row is inserted, then only the NEW variable has a value; if a row is deleted, then only the OLD variable has a value. As a final note, it is necessary to point out that many database developers avoid the use of triggers, if at all possible, for two main reasons: (a) most trigger systems add considerable overhead, and (b) triggers bring in issues of complexity usually associated with programming (i.e., it is hard to ensure that the right semantics are being implemented). However, triggers are very powerful tools and sometimes have no substitute to express business logic.

VI. CAPTURING BUSINESS RULES

A. Eliciting and Representing Business Rules in Conceptual Models:

Because business rules sometimes represent an implicit type of knowledge, elicitation of business rules share problems and challenges with knowledge elicitation in general. In particular, complex managerial and organizational issues must be addressed; at the minimum, a managerial action must be taken that encourages the sharing of knowledge by human agents. I will not address elicitation issues here, as they are outside the scope of this article.

Clearly, business rules should be captured in the requirement specification phase of a software project. It is during this phase that real-world information is analyzed and modeled and the scope of the system decided . However, there is no single technique that will guarantee capturing all business rules that an organization may need.

B. Expressing Business Rules in Data Models:

Even if business rules are somehow captured at requirement specification time, the current process for implementing, business rules (usually, to give some specification to a programmer so that an application can be written enforcing the rules) has several drawbacks: The process of creating and codifying a rule is long, labor-intensive, and involves several groups of people, creating a risk for miscommunication; assessing impact of rules (the quality and adequacy of the data produced by its application) is extremely difficult; and changing the rules is extremely difficult. To decide how to express a given business rule, the most promising approach is to classify the rule according to some characteristic that will allow one to determine an efficient implementation.

There are some obvious options, such as classifying rules as static or dynamic or classifying rules according to the data they affect. The first classification makes sense because some rules are more intimately bound to change than others. For instance, a rule stating that salary increases for employees cannot exceed 10% is clearly dependent on a change. Therefore, I define dynamic rules as those that are directly related to change, and static rules as those that are not. Even though this definition is somewhat ambiguous, it can be made more precise with a test. To specify the rule, is there a need to refer to the before and-after of a certain action? If one applies the test to the examples, the rule about employees' salaries not being higher than the managers can be seen as static, while the rules about employees salaries not increasing by more than 10% can be seen as dynamic. As for classifying rules according to the data they affect, to make such a distinction meaningful in the context of a database the classification must be based on the conceptual model used to design the database, one can then talk about rules affecting a single attribute, several attributes within a single entity, several attributes in several entities, and relationships, which implicitly means the related entities.

VII. IMPLEMENTING BUSINESS RULES IN DBMS

A. Rules that affect a single Attribute:

These rules should be implemented with a check. As an example, consider the previous rule about employees' ages. Implementing such rules as checks is advantageous because it is simple (i.e., checks are easy to write), natural (i.e., the check can be attached to the domain of the attribute in question or to its table), and, in the case where no other attributes are present, guaranteed to hold.

B. Rules that affect several attributes in one entity:

These may include some domain rules and some integrity rules and some facts and action. These rules should be implemented with an assertion, because even though checks can mention attributes in other tables, they are not guaranteed to hold.

C. Rules that affect several attributes in one entity:

These rules should be implemented with an assertion, for the same reasons as in the previous case. Assume, for instance, that there are ranks in the organization and each employee has a rank. Associated with each rank there is some information; for instance, the maximum and minimum salary for that rank. An obvious rule is that employees' salaries remain within what is dictated by each employee's rank. Then the assertion would have to mention both tables:

```
CREATE ASSERTION salary-rank
(NOT EXISTS (SELECT * FROM EMPLOYEE, RANK
WHERE Employee.rankid
=RANK.rankid AND
(Employee.salary < rank.min_salary OR
```

Employee.salary > rank.max_salary)))

VIII. FUTURE TRENDS

To implement business rules in a relational database, it is necessary not only to find some way to capture and express the rule in the requirement specification, perhaps using a combination of techniques (and surely going beyond what usual conceptual models have to offer), but also to find some way of implementing the rule in the database, a process that might get complicated.

It is clear that checks, assertions, and triggers together manage to cover most of the usual business rules; however, it is difficult to decide which tool to use for a particular rule. It therefore would be desirable to develop either a new construct specifically tailored to business rules or to extend the existing rules to make the task more manageable. From the analysis used here, some obvious avenues of development can be pointed out. For instance, extending checks and assertions with the ability to specify what to do in case a condition is violated or ensuring that a check is always guaranteed to hold would allow many static rules to be expressed without having to resort to triggers, which could then be used only if truly necessary.

IX. CONCLUSION

Although business rules are a very important part of information systems, there is little academic research on them. Some practitioners have contributed to the understanding of the subject, but there is no underlying framework for its study. Also, support for business rules in conceptual models and database systems leaves much to be desired. As a result, eliciting, expressing, and capturing business rules remain more of an art than a science. However, given their practical importance, more research in the subject is expected in the near future.

X. REFERENCES

[1]. Badia, A. (2004, March). Entity-relationship modeling revisited. ACM SIGMOD Record, 33(1), 77-82.

[2]. Chen, P. (1976). The entity-relationship model: Toward a unified view of data. ACM Transactions on Database Systems, 1(1), 9-36

[3]. Date, C. (2000). An introduction to database systems (7th Ed.). Reading, MA: Addison-Wesley.

[4]. Davis, F. (1993). Requirement specification: Objects, functions and states. Upper Saddle River, NJ: PrenticeHall.

[5]. Ibrahim, H., Gray, W.A. and Fiddian, N.J. SICSDD – A Semantic Integrity Constraint Subsystem for a Distributed Database. Proceedings of the 1998 International Conference on Parallel and Distributed Processing Techniques and Applications, USA, 1998, pp. 1575-1582.

[6]. Ibrahim, H., Gray, W.A. and Fiddian, N.J. SICSDD– A Semantic Integrity Constraint Subsystem for a Distributed Database. Proceedings of the 1998 International Conference on Parallel and Distributed Processing Techniques and Applications, USA, 1998, pp.1575-1582.

[7]. Ibrahim, H., Gray, W.A. and Fiddian, N.J. The Subsystem for a Distributed Database (SICSDD). *Databases*, UK, 1996, pp. 74-91.

CONFERENCE PAPER
National Conference on Information and Communication Technology for Development
Organized by PRMITR, Amravati (MS) India
http://mitra.ac.in/forthcoming.html