23

TOWARDS SEMANTIC INTEGRITY IN RELATIONAL DATABASES

Reinhardt A. Botha
Faculty of Computer Studies
Port Elizabeth Technikon
Port Elizabeth, South Africa
rbotha@computer.org

Abstract

The usefulness of data largely depends on its correctness, which is determined by the extent to which the data reflects the real-world and universe of discourse. Since the real world is constantly changing, it follows that data must constantly be changed. Since an integrity violation could occur when information is wrongfully changed, changes should only be entrusted to trustworthy users. The changes must, furthermore, occur according to business rules. This type of control falls within the domain of an access control service. This paper investigates activities involved to enforce semantic integrity in relational database environments, particularly those where access is controlled according to a role-based paradigm.

Keywords: Semantic integrity, role-based access control, business rules

1. INTRODUCTION

The proliferation of distributed environments, in particular the growth of the Internet and popularization of e-Commerce, emphasized the need to protect information [10]. This resulted in an increased awareness of information security.

Information security is concerned with ensuring that information stays both confidential and available, while also maintaining a state of integrity. To achieve this, the implementation of five information security services, namely authentication, confidentiality, integrity, non-repudiation and access control services, is suggested [3].

Several information security services may cooperate to provide suitable countermeasures against threats. Similarly, a single information

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35586-3_46

M. A. Ghonaimy et al. (eds.), *Security in the Information Society*

security service may contribute to the availability, confidentiality and integrity of the information. Consider, for example, the access control service.

The access control service supports *confidentiality* by allowing access only to authorized users. The access control service also ensures that information remains *available* for those who have the appropriate permissions. The access control service assists, furthermore, with upholding the *integrity* of information by ensuring that only specific users are allowed to alter the information.

For information to have integrity, its "soundness" needs to be above questioning. The information should be complete (it should be the whole truth) and valid (nothing but the whole truth) [7]. Leyman and Roller [5] identify three types of integrity:

- *Physical integrity* protects against the loss of data due to problems such as media failures and magnetic interference;
- *Operational integrity* is concerned with the synchronization of concurrent access to the data;
- *Semantic integrity* ensures that the data retains its meaning, i.e. that the data correctly reflects the real world that it models.

The following case study will be used in the ensuing discussion to illustrate the arguments.

1.1 Scenario

A simple ordering system for parts will be considered as a running scenario. There are suppliers who supply parts. Obviously not all suppliers supply all parts to the company. Parts are ordered from suppliers when the need arises. An order is sent to a specific supplier, but this order may simultaneously be for several parts.

When modelled as a relational database, this scenario can be depicted through the following simplified relations:

$$Supplier(Sno, Name)$$
$$Part(Pno, Description)$$
$$SupplierPart(Sno, Pno)$$
$$Order(Ono, Sno, Date, Approved, Submitter)$$
$$Orderline(Ono, Lineno, Qty, Pno)$$

This relational design embeds specific domain knowledge. This design, for example, captures details about the business objects. For instance,

by inspecting the Order and Orderline relations much information about "Orders" can be deduced.

However, business rules will specify when these objects will have integrity.

2. BUSINESS RULES SPECIFY CORRECTNESS

Data correctness requirements must be gathered from end-users and business regulations. These data correctness requirements are expressed as business rules. As such, they dictate the "what must be done", the "who may do it" and "under which circumstances" it should be done.

In the scenario, the following five business rules can be identified (among others):

- **BR1:** Order only from existing Suppliers.
- **BR2:** Order parts only from Suppliers who do supply the parts.
- **BR3:** Cannot change an order that is already approved.
- **BR4:** Auditors and Financial Managers should act independently.
- **BR5:** Financial Managers may not approve their own orders.

The business rules represent specific domain knowledge. Consider the specific types of domain knowledge that are embedded in these business rules.

Attribute domains. BR1 and BR2 specifically restrict the domain from which specific attributes may be selected. They, for example, specify that Order.sno may only have values that are represented in the Supplier domain.

Organizational roles and structure. BR4 and BR5 both contain references to specific job titles. BR5 also indicates specific responsibilities: a Financial Manager approves orders.

Business objects and associated operations. From BR3 we can say that "orders" can be "approved". BR5 implies that an order has an owner, which further investigation will show to be the person who submitted the order.

Business process information. BR3 dictates a specific order of events. These kind of business rules will provide additional knowledge about the internal workings of business objects (in this case an order has a state) or the business processes that are followed (changing happens before approving). Similarly BR5 implies that orders are approved after being submitted.

Control principles. Business rules could stipulate the use of well-known control principles such as separation of duties. BR4 specifies a static separation of duty constraint, i.e. it should never happen that a person is both a Financial Manager and an Auditor. BR5 specifies a dynamic separation of duty constraint, i.e. enforcement changes depending on knowledge about the submitter of an order.

Several business rules will have an effect on the functioning of the access control service. The next section will argue that some of the domain knowledge required to achieve semantic integrity is embedded in access control information. Particular focus falls on role-based access control.

3. ROLE-BASED ACCESS CONTROL

Role-Based Access Control (RBAC) is used widely within organizations. RBAC is policy neutral in that it provides a means for articulating policy rather than embodying a particular security policy [9].

In order to show that some domain knowledge will be modelled through role based access control, consider the main concepts within RBAC as depicted in figure 1. The concepts presented here are based on the well-known RBAC96 model [9].

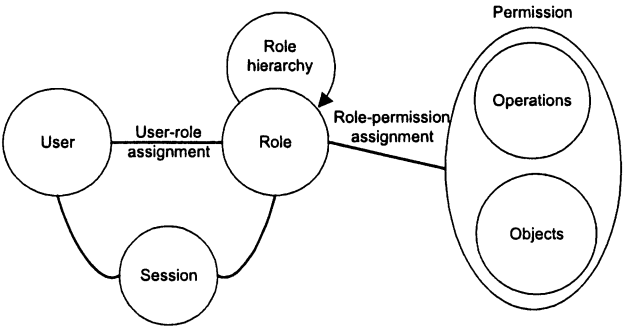

Figure 1. Role-based Access Control Model

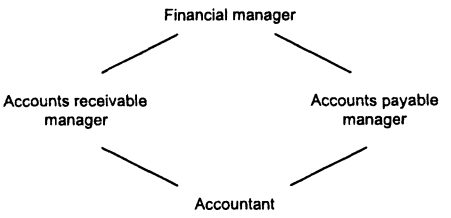

Figure 2. An example role hierarchy

The notion of a role is central to RBAC. **Users**, being human beings, programs or processes, are uniquely identified. A **role** often names a function or a job title within the organization. A user can have many roles, and a role, in turn, can have many users, depicted by the **user-role assignment** in figure 1. A **permission** constitutes the ability to do something. A role can have many permissions, and the same permission can be assigned to many roles according to the **role-permission assignment** relation in figure 1.

Roles may, furthermore, be related to other roles according to a role hierarchy. A **role hierarchy** is typically structured to reflect the organization's line of authority and responsibility [9]. Figure 2 depicts a typical role hierarchy. It shows that a "Financial Manager" role is senior to both the "Accounts Receivable Manager" and the "Account Payable Manager" roles. The last named two roles, in turn, are senior to the "Accountant" role. Users who are associated with the "Financial Manager" role will inherit the roles of "Account Receivable Manager", "Account Payable Manager" and "Accountant".

A user that is associated with a superior role may also choose to only activate a subset of the roles inferior to the superior role. A user who is associated with the "Financial Manager" role in figure 2, may choose to only assume the role of "Accountant". The **session** entity, as depicted in figure 1, provides the run-time association between a user and the roles that he/she wishes to activate.

In a generalized model, permissions are a set of uninterpreted symbols. When implemented, a permission would relate to the ability to perform a specific operation on an object. Permissions are thus typically described in terms of the underlying semantics of the system. In an order processing system, a permission may be the ability to "approve an order", while from the operating system perspective "create directory" may be an example of a permission.

Roles and the relationship between roles provide another type of domain knowledge. The relation to job titles and responsibilities in the

organization represents the ability to capture knowledge about the organization.

Business rules present us with knowledge about how the business ideally should operate. The information systems that is used to assist the business must therefore enforce these business rules. Semantic integrity constraints that model these business rules can thus be developed.

4. FROM BUSINESS RULES TO SEMANTIC INTEGRITY CONSTRAINTS

Semantic integrity constraints specify conditions on the database objects that must be satisfied for data to be a correct reflection of the real world. Different mechanisms are provided by database management systems to implement such constraints.

It is important to realize that semantic integrity constraints only concentrate on ensuring that information is valid, i.e. that the information is the truth [7]. Semantic integrity constraints cannot ensure that the information is complete, i.e. that it is the whole truth.

Consider, for example, an order. Semantic integrity constraints can enforce that an order is placed at a valid supplier and that for each orderline the supplier supplies the ordered part. However, a semantic integrity constraint cannot be formulated to ensure that all the order lines for an order are captured.

It is safe, however, to assert that semantic integrity constraints will contribute to ensuring semantic integrity. The next section will evaluate how semantic integrity constraints can be built into modern relational databases.

5. BUILDING INTEGRITY CONSTRAINTS IN RELATIONAL DATABASES

Commercial relational database technology provides a variety of mechanisms to enforce semantic integrity constraints. Interaction with relational databases is typically based on the well-known SQL language. The new SQL standard [4], known as SQL:1999 provides several features that are geared towards providing semantic integrity support.

This section will consider how some of these features, in combination with appropriate design activities, could be utilized to achieve semantic integrity in relational databases. The section commences by discussing how to appropriately abstract permissions.

5.1 Abstracting Permissions

During design-time business artifacts will be modelled as objects in the system. These objects represent specific domain knowledge and have application specific semantics. This allows for the modeling of permissions that are specific to our domain of discourse. In a general model permissions might be seen as a set of uninterpreted symbols [9]. However, when the model is implemented within a specific domain the notion of permissions has specific semantics.

Different objects allow different operations on them. For instance, in the example of section 1.1 a permission might be the ability to "approve" (operation) an "order" (object).

During design-time it is important to identify possible permissions. Particular attention must be paid to those permissions that may threaten the integrity of the data. Some permissions, for example the ability to "view an order", do not have integrity repercussions.

In a relational database environment the permissions of concern are those that perform insert, update or delete operations on the underlying tables. Note however that an abstract permission may relate to more than one database operation. Table 1 shows how the "edit an order" permission relates to possibly five database operations that should be allowed.

Table 1. Mapping the abstract permission "edit an order" to operations on relation database tables

Operation	Database table	Specific columns
INSERT	Order	-
UPDATE	Order	Sno
INSERT	Orderline	-
DELETE	Orderline	-
UPDATE	Orderline	Qty, Pno

Designing permissions that are of an appropriate granularity is important. On the one hand permissions must be of a fine enough granularity so that proper control can be exercised. On the other hand, permissions must be of coarse enough granularity to allow for ease of administration.

Within a role-based access control environment it is further necessary to consider carefully how role hierarchies are designed.

5.2 Design of Role Hierarchies

Permissions associated with roles are inherited according to the specification of role hierarchies. It is thus important to ensure that role hierarchies are properly designed.

Moffett [6] argues that various different hierarchies exist within an organization. There are, for example, the organizational structure (e.g. departments and business units), generalization hierarchies (is-a relations – a nurse "is-a" healthcare provider), aggregation hierarchies (financial control consists of Financial Forecasting and Financial Accounts) and supervision hierarchies (managers and supervisors). Moffett [6] shows that careful consideration must be given to the inheritance of access permissions when an existing hierarchy is used as the role hierarchy. If users inherit unnecessary permissions it may hold a serious threat to data integrity.

However, not all permissions are inherited, some are directly given. Careful attention must be paid to the administrative activities.

5.3 Administrative Activities

In a RBAC environment, some business rules may be expressed as constraints on the associations between entities. For example, business rule BR4 states that "Auditors and Financial Managers should act independently". This rule, an example of a static separation of duty constraint, can be enforced by ensuring that a user may not be associated with both the "Financial Manager" and "Auditor" roles [1].

This can be modelled as a constraint on the role-permission assignment relation. Administration tools should be sensitized to such constraints in order to prevent administrators to inadvertently make associations that are against organizational policy.

Whereas some business rules purely have an access control flavor, others resort fully within the application domain. In particular, integrity constraints must be defined on the application database.

5.4 Declarative Definition of Integrity Constraints

SQL:1999 allows for several language constructs that can be used to declaratively define integrity constraints (summarized in table 2). Integrity constraints may be checked after each statement or only at the end of a transaction.

A constraint such as BR1, for example, can be implemented by specifying that Order.Sno is a foreign key that references Supplier.Sno. Other

Table 2. Declarative integrity constraints in SQL:1999

Language element	Description
NOT NULL	prevents a column from taking a null value
DEFAULT	sets the default value of a column
UNIQUE	defines that a column(s) must have unique values within a table
PRIMARY KEY	defines the primary key of a table
FOREIGN KEY (REFERENCES)	specifies a foreign key whose values must match that of a unique/primary key
CHECK	defines an integrity constraint based on a search condition
DOMAIN	creates a restricted column domain
ASSERTION	defines a table-independent integrity constraint based on a search condition

constraints, such as BR2, cannot be defined through a declarative integrity specification. To defines such constraints, we turn to a procedural definition.

5.5 Procedural Definition: Creating Triggers

Integrity constraints that depend on application semantics are prime candidates for being implemented through a procedural definition. Commercial databases implement the procedural definition of integrity constraints through the concept of triggers. Triggers are similar to ECA rules in active databases [8].

Triggers fire when a database event, typically an insert, update or delete, occurs. A condition is evaluated and an appropriate action performed.

BR2 would, for example, require a trigger to fire when an insert into or update on the Orderline relation is done. Checking the condition involves interrogating the Order relation to determine who the supplier for the order under scrutiny is, and to interrogate the SuppPart relation to determine whether that supplier is a valid supplier of the part being ordered.

Triggers are often required where the business rule is highly reliant on specific application semantics. Consider the business rule "an order may not be edited after it has been approved". This business rule relies on the semantics of the application. It requires, for example, that the notion of the "state of an order" and the permission "approve an order" are defined within the application domain. Triggers that relate to the underlying tables involved and events generated when the permission "approve an

order" is exercised will have to be developed. These triggers will have to interrogate the application tables to determine what the state of the order is. Based on this information, an applicable decision, in line with the business rule, will be made.

Due to the overhead involved with triggers, they should not be used to enforce integrity constraints that can be stated declaratively. Current implementation of triggers in commercial databases is the first step in the evolution of active rules in database systems [2]. Enhancements to triggers will include better control over the way in which events are processed, for example, by introducing the ability to defer execution of triggers.

6. CONCLUSION

Several mechanisms exist to assist with achieving semantic integrity. However, a lack of application of these techniques in software development points to interesting and urgent research to be conducted.

In particular the extraction of business rules during requirements gathering and the translation of those business rules to integrity constraints require attention. Future research will address the lack of design methods and suitable design tools. The time for rigorous methodologies and techniques to ensure semantic integrity is ripe.

REFERENCES

[1] Reinhardt A. Botha and Jan H.P. Eloff. Separation of duties for access control enforcement in workflow environments. *IBM Systems Journal*, 40(3):666–682, 2001.

[2] Stefano Ceri and Raghu Ramakrishnan. Rules in database systems. *ACM Computing Surveys*, 28(1):109–111, March 1996.

[3] ISO 7498-2: Information Processing Systems — Open System Interconnection — Basic Reference Model – Part 2: Security Architecture, 1989.

[4] ANSI/ISO/IEC 7075-2:99 ISO International Standard: Database Language SQL - Part 2: Foundation (SQL/Foundation), September 1999.

[5] F. Leyman and D. Roller. *Production Workflow: Concepts and Techniques.* Prentice–Hall, 2000.

[6] J.D. Moffett. Control principles and role hierarchies. In *Proceedings of 3rd ACM Workshop on Role-based Access Control*, pages 63–69, 22–23 Oct 1998.

[7] Amihai Motro. Integrity = validity + completeness. *ACM Transactions on Database Systems*, 14(4):480–502, December 1989.

[8] N. W. Paton and O. Díaz. Active database systems. *ACM Computing Surveys*, 31(1):63 – 103, 1999.

[9] R. S. Sandhu, E. J. Coyne, H. L. Fenstein, and C. E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, Feb 1996.

[10] L. Weinstein and P. Neumann. Inside risks: Internet risks. *Communication of ACM*, 43(5):144, March 2000.