TOWARDS A PATTERN-BASED APPROACH FOR ACHIEVING SEMANTIC INTEGRITY IN (OBJECT-)RELATIONAL DATABASES

Lance Ingram and Reinhardt Botha
Faculty of Computer Studies,
Port Elizabeth Technikon, Port Elizabeth
{lance,reinhard}@petech.ac.za

Abstract An aspect of information security is the information's integrity. An important aspect of integrity is that the information must retain its appropriate meaning. Semantic integrity rules specify requirements for ensuring that information maintains its meaning. It is believed that certain reoccuring themes are present in these semantic integrity rules. Pattern languages are often used to present solutions to such regularly occurring problems. This paper will therefore investigate semantic integrity from a pattern-based perspective. In doing so, the paper will argue that the implementation of certain semantic integrity rules can be aided through identifying patterns.

Keywords: Integrity, Semantic Integrity, Patterns

1. INTRODUCTION

An organization's security policy is expressed through business rules that dictates the behavior of users. It thus embodies the rules and regulations of the organization. These business rules express, among other requirements, the information integrity requirements of the organization. In particular, semantic integrity requirements ensure that changes to organizational information must happen according prespecified business rules. In other words, business rules are enforced to prevent the misuse of the organization's information.

The importance of information is emphasized by the change in organizations' methods of performing business. The proliferation of distributed environments, in particular, emphasized the need to protect information.

The field of information security has, therefore, received increased attention. The growth of the Internet and popularization of e-Commerce

has fueled these threats [9, 10]. Information security is concerned with ensuring that information stays confidential and available, while maintaining a state of integrity.

To be able to maintain the confidentiality, availability and integrity of information, five information security services should be implemented, namely: Authentication, Confidentiality, Integrity, Non-Repudiation and Access Control [2, 4]. These security services all collaborate to accomplish a common goal, which is to provide information availability, to maintain a state of integrity and confidentiality of information within an organization or institution. These goals have received increased attention as more and more organizations and institutions rely on technology to provide a more efficient way of doing business.

Organizations that make use of distributed technologies have the ability to interact and communicate information more effectively. An important element of communication between organizations is the electronic transfer of information. Although it is important that information is not tampered with in transit, it is also important that the information correctly reflects the state of the organization. Therefore information needs to have integrity. Information integrity has been chosen as this paper's primary focus. The next section expounds on integrity as a property of information.

1.1. INTEGRITY

Information has integrity when it reflects the real-world [2]. Therefore, information should be complete (it should be the whole truth) and valid (nothing but the whole truth) [7]. To achieve integrity three different types of measures must be considered [6]:

- Operational integrity measures deal with the synchronization of concurrent access to the data;
- Physical integrity measures protect against the loss of data from media failures;
- Semantic integrity measures ensure that the data reflects the true state of affairs.

The parameters in which a business operates is defined through the business rules. Business rules thus dictate what may and may not happen in the business. Semantic integrity is achieved when data complies with appropriate business rules.

According to Leymann and Roller [6] business rules can be categorized as being static, transitional and dynamic.

- Static business rules ensure the validity of each database state. For example, the total on an invoice must be equal to the sum of all the individual orders on that invoice.

- Transitional business rules ensure the validity of transition between two consecutive database states. For example, in financial accounting a double entry system is used. In such system for each debit there needs to be a credit. Another financial oriented example, is that you can only remove items from an order. Instead an explicit negation must be added.

- Dynamic business rules ensure the validity of corresponding transitions by refering to more than two database states. Each corresponding transition will be validated between the database during run-time of a certain event. For example, you cannot approve an order if the invoice for that specific order wasn't received yet.

When interpreted in the context of an information system, a business rule such as "an order may not be edited after it has been approved" aims to ensure that the system reflects the state of the physical "world" it represents, in that it restricts the actions that may be performed at a specific time. If a user could perform an edit on a specific order after it has been approved, semantic integrity would not be maintained since the business rule "an order may not be edited after it has been approved" would be violated. In essence, semantic integrity is maintained if information has a consistent and accurate meaning in the business.

Business rules are, therefore, enforced to prevent the misrepresentation of the organization's information. The proposed study is principally motivated by the realization that the information of organizations should reflect the true state of affairs, i.e. have semantic integrity.

2. INTEGRITY MECHANISMS IN (OBJECT-)RELATIONAL DATABASES

Current commercial database technology provides a variety of mechanisms to enforce semantic integrity constraints. The SQL:1999 standard [5] defines a language to interact with (object-)relational databases. Commercial products will aim to meet the standard requirements. The SQL:1999 standard has many features which can assist with semantic integrity. The following section will discuss the definition of both declarative and procedural integrity constraints that SQL:1999 supports.

2.1. DECLARATIVE DEFINITION OF INTEGRITY CONSTRAINTS

SQL:1999 allows several language constructs that can be used to declaratively specify integrity constraints. Integrity constraints may be checked after each statement or only at the end of a transaction. These declaratively integrity constraints allow database application to enforce integrity upon each request. The constraints such as *check* defines an integrity constraint based on a search condition, while *not null* prevents a column from taking a null value. However not all constraints can be stated declaratively. Therefore certain procedural mechanisms exist.

2.2. PROCEDURAL DEFINITION OF INTEGRITY CONSTRAINTS BY TRIGGERS

Commercial databases implement the procedural definition of integrity constraints through the use of triggers. Triggers is based on the ECA rule in active databases [8]. Triggers fire on databases event, typically an insert, update or delete, occur. The condition is evaluated and the appropriate action is performed [1].

Although it can be seen that many mechanisms exist in commercial (object-)relational database, developers sometimes also have difficulty in choosing an appropriate mechanism. This paper suggest that certain patterns in business rules can be identified, which would aid developers with choosing an appropriate implementation mechanism.

To demonstrate this point we look at a specific pattern for a specific business rule and discuss the various implementation strategies associated with the pattern.

3. SCENARIO

A pattern is a regular occurring form, that documents the solution to a common problem [3]. Within information systems the requirement for something to exist before we could do the next step is quite common. Often that requirement is necessary to ensure that the information have the appropriate meaning. For example, it would not make sense to store detail of an order which does not exist, nor would it be sensible to order from suppliers which does not supply specific products.

We can therefore identify the “Pre-existing values” pattern as below. Note that the pattern has several possible implementations.

Name: “Pre-existing values”

Requirements: That certain values may only be used if they exist in another place.
Implementation of the pattern: There are numerous ways to solve the problem that relates to the chosen pattern namely:

- Primary-foreign keys
- Triggers
- Encapsulation

Each of these implementation options will now be discussed in more detail.

4. IMPLEMENTATION OF "PRE-EXISTING VALUES" PATTERN

The "pre-existing values" pattern will be evaluated and discussed in each of the following sections. Each section will consist of a description that will explain the concept of each solution. It will also consist of an example business rule. ORACLE code shows an implementation of the proposed solution to the "pre-existing values" pattern. Each solution will be commented on to indicate certain requirements or downfalls.

4.1. PRIMARY-FOREIGN KEYS

Description In (object-)relational databases primary and foreign keys are used to ensure that integrity is maintained. By making use of primary and foreign keys, a relationship between two entities is created. If one entity consist of information that does not exist in the other entity integrity would have been violated.
Example The Order table contain the **SupplierNo** which is related to the Supplier table **SupplierNo**. The Supplier need to be a valid Supplier for an Order to be approved. When a relationship between two entities is established integrity is enforced by the relationships between the entities.
Solution

```
CREATE TABLE ORDER
( OrderNo NUMBER(5,0) NOT NULL,
  SupplierNo NUMBER(5,0),
  CONSTRAINT Order_pk PRIMARY KEY (OrderNo),
  CONSTRAINT Supplier_fk FOREIGN KEY (SupplierNo) REFERENCES SUPPLIER(SupplierNo));

CREATE TABLE SUPPLIER
(SupplierNo NUMBER(5,0) NOT NULL,
 CONSTRAINT Supplier_pk PRIMARY KEY(SupplierNo));
```

Comments Normalization often causes a single object in the real world to be represented by multiple entities in the database. For example, the Order and Orderline table both contributes to the real world "invoice" object. Normalization has well-known advantages as far as maintenance (and therefore integrity) is concerned. However, primary-foreign key constructs only is applicable where only two tables are involved.

4.2. TRIGGERS

Description Certain integrity requirements require synchronization of data between multiple tables. The use of triggers allows the (object-)relational database to do synchronizations between multiple tables.

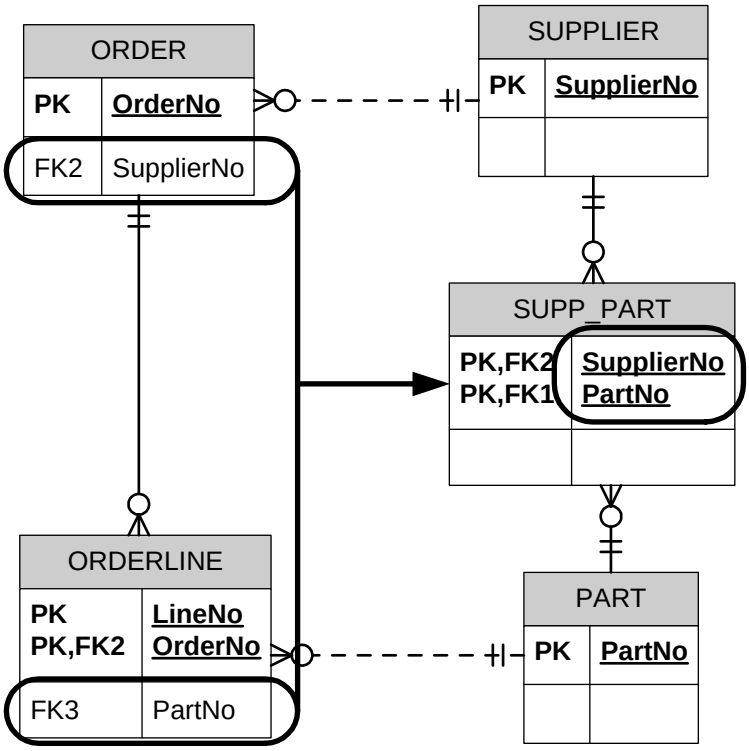

Figure 1 Implementation of Trigger

Example Figure 1 illustrates the requirement that a product can only be ordered from a supplier who actually supplies that product. As such the combination of SupplierNo in the Order table and the corresponding PartNo in the Orderline table must exist as a combination in the Supp_part table.

Solution

```
CREATE OR REPLACE TRIGGER ins_order_line
BEFORE INSERT ON ORDER_LINE
FOR EACH ROW
BEGIN
  SELECT ORDER.OrderNo
  into v_order
  FROM
  ORDER,
  SUPP_PART
    WHERE ORDER.Sno = SUPP_PART.Sno
    AND :new.Pno = SUPP_PART.Pno
    AND ORDER.Ono = :new.Ono;
EXCEPTION
WHEN NO_DATA_FOUND THEN
  RAISE_APPLICATION_ERROR(-20010,'Violation of rules');
END;
/
```

Comments Triggers can only be based upon tables and react on events such as insert, update and delete. These are however the only events that may cause integrity problems. An aspect to consider is that the trigger may under certain circumstances perform corrective action if an event violates integrity.

Triggers have the further disadvantage that they are relatively expensive and as such should not be used when another implementation mechanism (such as primary-foreign keys) are available.

It is also interesting to note that many of the situations where triggers are required actually originate from the process of normalization. In particular this is the case when a physical object is represented in multiple tables, but the combination of values between the various tables are still of importance.

4.3. ENCAPSULATION

Description In Object-Orientation the child object will inherit features and functionality from the base object. Thus allowing the child object to act as the base object, this also indicates that the child do not require to reinvent the wheel, but access the features that already exist within base class. This allows reusability of features and functionality that already exist.

Example The Order table can be created as an object that allows to SupplierNo in the Order table reference Supplier object that already exist. The Supplier table can be created as a object that makes the

Supplier and feature of the Supplier object that can be reference by the child object in the Order table.

Solution

```
CREATE OR REPLACE TYPE SUPPLIER_T AS OBJECT
( SupplierNo NUMBER(5,0) NOT NULL
  //and other fields
CONSTRAINT Supplier_pk PRIMARY KEY(SupplierNo));

CREATE OR REPLACE TYPE ORDER_T AS OBJECT
( OrderNo NUMBER(5,0) NOT NULL,
  SupplierNo REF SUPPLIER_T
  //and other fields
  CONSTRAINT Order_pk PRIMARY KEY(OrderNo));
```

Comments There is an increase in complexity for developers due to the fact that they need to change their mindset to a more object-orientated approach. The fact that object-orientation allows the reusability of table, function and triggers is by itself a great advancement in databases. The encapsulation of objects in other objects intuitively model the relationship between those objects. Techniques such as information hiding also adds additional integrity features.

5. CONCLUSION

In this paper semantic integrity rules were evaluated to identify a regularly occurring pattern. The pattern that was identified were discussed to indicate the various implementation solutions available for the specific pattern. In doing so the paper has illustrated that the implementation of certain semantic integrity rules can be aided through identifying patterns.

It is foreseen that complex business rules could be built from using these reoccuring patterns in different combinations. We will therefore expand on the ideas presented in this paper through the identification of additional patterns for semantic integrity requirements. Each pattern will be studied in detail and implemented in a database such as Microsoft SQL Server and Oracle. In this process metrics will be created and arguments for choosing the appropriate implementation option for a database pattern. The combination of different patterns to construct more complex business rules will also receive attention.

References

[1] Reinhardt Botha. Towards semantic integrity in relational databases. In *IFIP TCII Eighteenth Annual Working Conference on Information Security*, Cairo, Egypt, May 2002.

[2] G. Bruce and R. Dempsey. *Security in Distributed Computing*. Prentice Hall, 1997.

[3] Ward Cunningham. The check pattern language of information integrity. In James O.Coplien and Douglas C. Schmidt, editors, *Pattern Languages of Programs design*. Addison Wesley, 1995.

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

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

[6] F. Leymann and D. Roller. *Production Workflow Concepts and Techniques*. Prentice Hall, 2000.

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

[8] N. Paton and O. Diaz. Active database systems. *ACM Computing Survey*, 31(1):63 – 98, March 1999.

[9] R. Von Solms. Information security management (1): why information security is so important. *Information Management and Computer Security*, 7(1):174 – 177, April 1998.

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