2006-1613: CONCEPTUAL MODELING OF BUSINESS RULES

Reza Sanati, Utah Valley State College

REZA SANATI MEHRIZY is an associate professor of the Computing and Networking Sciences Dept. at Utah Valley State College, Orem, Utah. He received his MS and PhD in Computer Science from University of Oklahoma, Norman, Oklahoma. His research focuses on diverse areas such as: Database Design, Data Structures, Artificial Intelligence, Robotics, and Computer Integrated Manufacturing.

Curtis Welborn, Utah Valley State College

Afsaneh Minaie, Utah Valley State College

Conceptual Modeling of Business Rules

Abstract

The Entity Relationship (ER) and/or Enhanced Entity Relationship (EER) notation can be used to graphically represent action assertions. Action assertions are one of the three types of recognized business rules that control daily business operations. This graphical action assertion representation enforces constraints on the way data should be used by the organization and is implementation independent.

An organization may have many business rules that need to be implemented to correctly control the daily operation of the business. Traditionally these business rules have been encoded into application programs that have been virtually unrecognizable, unmanageable, and inconsistent[3]. Capturing business rules in such a way can lead to the business rules being inconsistent between various user applications. This approach places a heavy burden on the programmer, who must know all the constraints that an action may violate and must include checks for each of these constraints. An omission, misunderstanding, or error by the programmer will likely leave the database in an inconsistent state.

In this paper the ER/EER notation will be used to define business rules at the conceptual level in a relational data model. Placing the business rules at the same level as the data has a natural appeal because the business rules are always about the data. The business rules are thus used to enforce database consistency that goes beyond foreign key relationships and delves into how an organization should operate its business.

Introduction

Business rules are basic to what the business knows about itself. Rules must be explicit. No rule can ever be assumed about any concept or fact[7]. Business rules can be divided into the following three types:[1] structural assertions, action assertions, and derivations. Structural assertions are concerned with statements that express an aspect or relationship about the structure of a business. To define structural assertions an organization may also need to define business terms and facts. Business terms are the actual terms an organization uses to define how the business is to operate. A business term should have a specific meaning to an organization. Facts define relationships between terms. Derivations are concerned with statements that can be used to derive additional facts about the business. Derivations use facts known to the system to derive new facts based on some inference method. Action assertions are statements that control or limit the actions of the business. Action assertions focus on the dynamic behaviors of an organization by specifying what the organization should allow to happen and what the organization should not allow to happen.

This paper will focus on action assertions, which are constraints. In this way, business rules will be used to constrain different operating aspects of the business[2]. When the bulk of an organizations constraints are captured by implementing them in various user application

programs the constraints can be very difficult to manage. The difficultly arises because the form of the constraints can be obscured by procedural logic which is used to implement the constraint. This approach places a heavy burden on the programmer to know all the constraints that an action may violate, to implement them carefully, and to include a check for each of these constraints within each user application. Within a large organization this is not a reliable approach because an omission, misunderstanding, or error by the programmer will likely leave the database in an inconsistent state with respect to some business rule.

Defining business rules at a conceptual level without specifying how the constraint is to be implemented is a more modern and more reliable approach. The aim of this approach is to build the constraints into the system while reducing the chance of programming errors. The EER notation works well for specifying many constraints at the conceptual level. The EER notation has been augmented to allow business rules to be captured in a graphical form that is not supported by the simpler ER notation.[3] Capturing business rules and data relationships at the same level in a graphical form has a natural appeal because the business rules are all about defining constraints about how the data should and should not be used[4, 5].

The ER/EER notation will allow the business rules of the system to be captured at the conceptual level in a relational data model without specifying how the rule will be implemented. In this paper the conceptual representation will be used to enforce constraints in a relational data model specifically for the types of services a mechanic can provide for an airplane and for which pilots can fly an airplane. To capture these business rules we start the process with a simple example that will be improved gradually through the paper.

Capturing Business Rules

Figure 1 is an Entity Relationship Diagram that depicts the following information about airplane mechanics. A mechanic is an individual with skills that allows him to maintain airplanes. A mechanic must receive specific types of training related to maintaining airplanes. There are many different types of training that a mechanic can receive for maintaining airplanes, such as training on landing gear, training on engines, training on electronics, and so on. In turn, the types of training that a mechanic receives are used to determine the types of maintenance services that the mechanic can perform on an airplane. A specific maintenance service may require that a mechanic receive more than one type of training. Yet, a specific type of training may be useful in providing more than a single maintenance service.

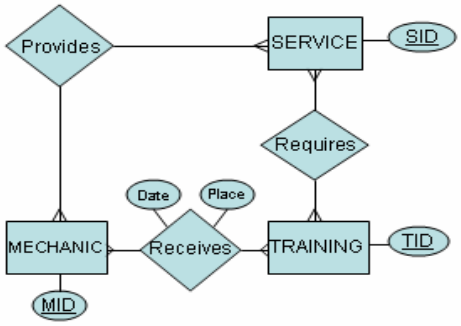

Figure 1: Mechanics

While the entities (Mechanic, Training, Service) and relationships (Received, Requires, Service_Provided) of Figure 1 depict the information discussed about mechanics, training and services, there is an implied constraint upon this information that is not depicted in Figure 1. The constraint could be stated as "a mechanic can only provide a maintenance service if he has received all the training required for that service." The ER model presented in Figure 1 indicates that a mechanic may receive many different types of training and provide many different services. It does not guarantee that the mechanic will receive the required type of training for the type of service that he provides.

To make sure that the mechanic has received the required type of training for the type of service he provides, we must make sure that for each row that appears in the Provides table with the attributes MIDi and SIDj, there is a row in the Receives table with the same attributes MIDi for all TIDs required by the service identified by SIDj. The modified ER diagram represented in Figure 2 shows the same Entity Relationship Diagram as Figure 1 with the addition of this constraint. It enforces this constraint (business rule) explicitly. The arrows mean the relationship "Provides" enforces the relationships "Receives" and "Requires"[3].

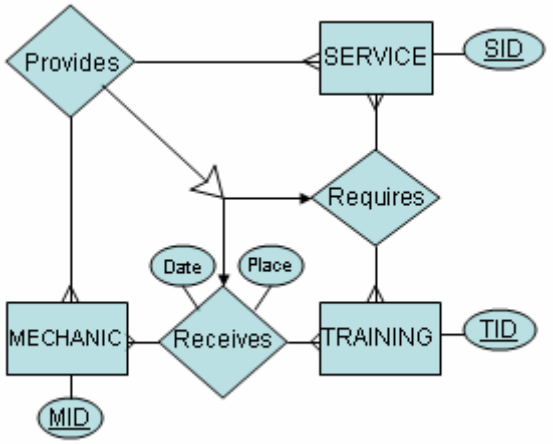

Figure 2: Mechanics + Training Constraint

Figure 3 is the schema for the ER diagram represented in Figure 2.

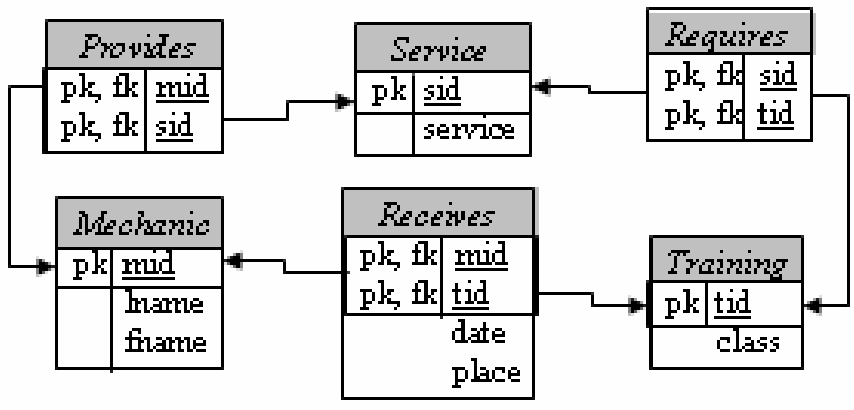

Figure 3: Schema

The following tables show some entities for the tables created from Figure 2 with the addition of some new attributes for Service, Mechanic and Training.

Provides

mid	sid
100	1010
100	1020
101	1005
102	1005
102	1022
102	1032

Requires

sid	tid
1000	10
1005	15
1010	10
1020	10
1022	10
1022	15,
1022	21
1032	10
1032	15
1032	21
1032	30

Service

sid	service
1005	engine repair
1010	repair skin
1020	repair landing gear
1022	approve alteration
1032	taxi jet

Receives

mid	tid	date	place
100	10	10-23-2003	dia
101	10	07-11-2002	dia
101	15	09-18-2005	dia
102	10	09-07-2000	pa
102	15	03-01-2002	pa
102	21	11-27-2004	pa
102	30	01-05-2006	dia

Mechanic

mid	lname	fname
100	Smith	Mark
101	Harris	Ken
102	Mast	Cory

Training

tid	class
10	airframes
15	power plant
21	IA
30	Citation 3 - Init

Figure 4: Sample Data

Continuing on with our mechanic example, we learn that one or more specific tools are often required to perform a maintenance service. Figure 5 depicts the mechanic example with the addition of TOOL entity type.

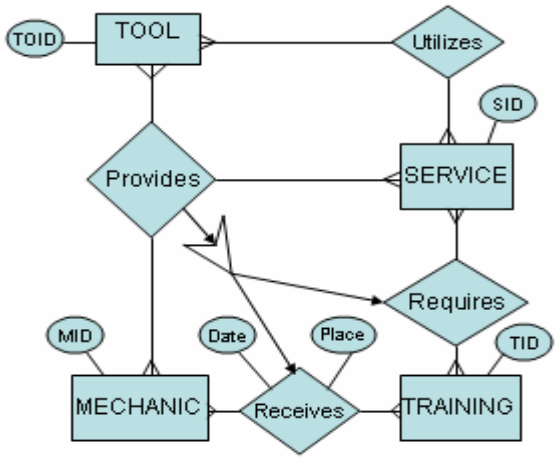

Figure 5: Mechanic + Tools

With the addition of TOOL entity type, our original constraint can also be expanded. While the example still has the constraint that "a mechanic can only provide a maintenance service if he has received all the training required for that service," in addition the constraint should include that "a maintenance service must be done using required tools." This requirement (constraint) can be enforced as in Figure 6.

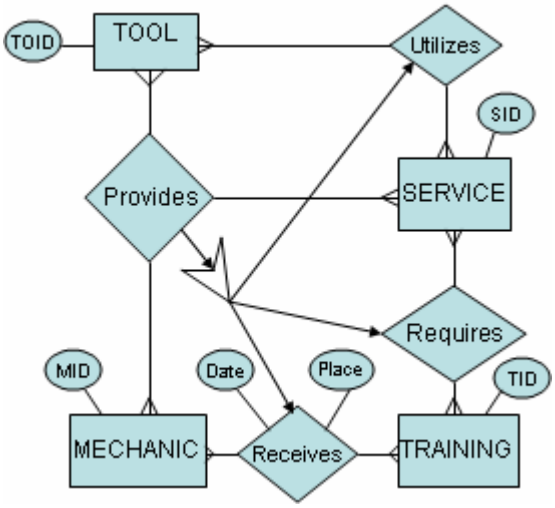

Figure 6: Mechanic + Tool Constraint

By looking at our example a little more, we realize we do not want mechanics and tools just to exist. They should be associated with airplane hangers where the maintenance work is provided. An individual hanger can have more than one mechanic working in it, and a hanger would have many tools. Within this example we will limit a tool to being in only one hanger, while a mechanic can work in more than a single hanger. Adding these entities and the relationships to our example results in Figure 7.

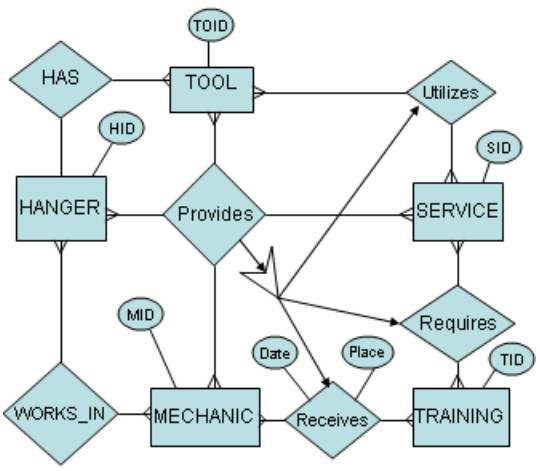

Figure 7: Mechanic + Hanger

Our constraint can now be expanded to include the fact that "a maintenance service is only provided in a hanger." The complete constraint would be that "a hanger can only provide a maintenance service if there is a mechanic that works in the hanger that has received all the training required for that service and the hanger has all the tools needed and the mechanic is using the required tool for that service." The new constraint is show in Figure 8.

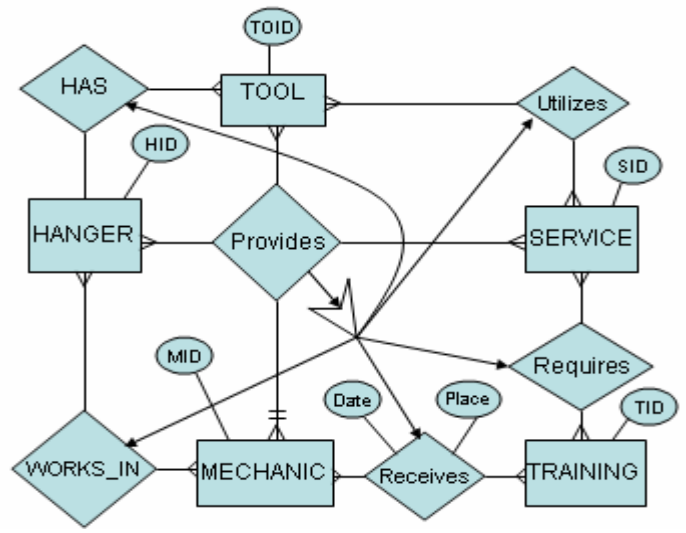

Figure 8: Mechanic + Hanger Constraint

A constraint now can be expressed as: "a service can only be provided by a hanger if there are two mechanics that can provide the same service." The justification for the additional constraint is that one mechanic performs the maintenance service while the other mechanic reviews the work that is performed. This additional condition is represented in Figure 8 by the cardinality symbol (⫤⋖) next to the MECHANIC entity type.

Extending the Domain

To make the problem more realistic, it will be extended into the domain of Pilots, Airplanes and Airports. Rather than combining the Pilot, Airplane and Airport information directly into our example, we will first develop these relationships independent of the original example and then merge the two examples to create a more complete system of relationships and constraints.

A trained pilot would be able to fly multiple different airplanes. The airplanes a pilot can fly may all be of one type or the pilot could be trained to fly various types of airplanes. A single airplane can be flown by many different pilots. Prior to a pilot flying an airplane, the pilot would have to be at a particular airport. Many different pilots could be at the same airport at any point in time. The above information for pilots, airplanes and airports is depicted in Figure 9.

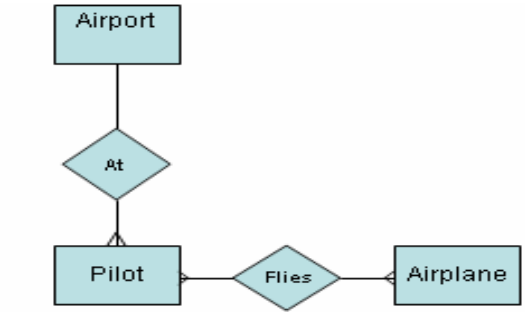

Figure 9: Pilot, Airport and Airplane

Two additional relationships can be specified by merging the information in Figure 8 with the information in Figure 9. The first relationship is that an airport can have multiple hangers, yet a hanger can be at only one airport. The second relationship is that an airplane is stored in a single hanger even though the hanger may have multiple airplanes stored in it. The merged information is shown in Figure 10.

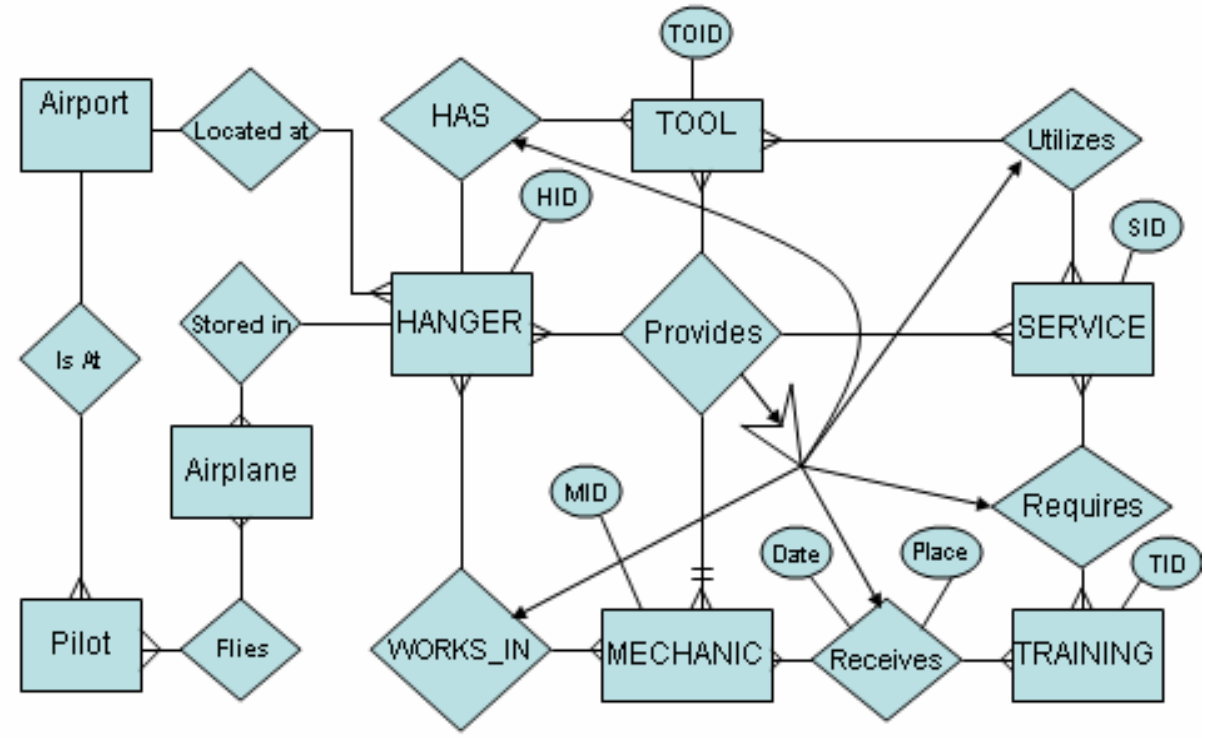

Figure 10: Mechanics + Pilots

Using the merged information from our mechanics and pilots examples, a new constraint can be added to our example to make it more complete. A pilot can only fly an airplane if the airplane and the pilot are located at the same airport. The constraint is developed from the Flies relationship and takes into account what airport the pilot is currently at, what hanger an airplane is currently stored in, and in which airport a hanger is located. The new constraint is shown in Figure 11. The ER/EER diagram now contains a rather complex system which shows not only the relationships between mechanics, training, services, tools, hangers, pilots, airports and airplanes, but also the business rules which capture at the conceptual level who can perform maintenance services and who can fly an airplane.

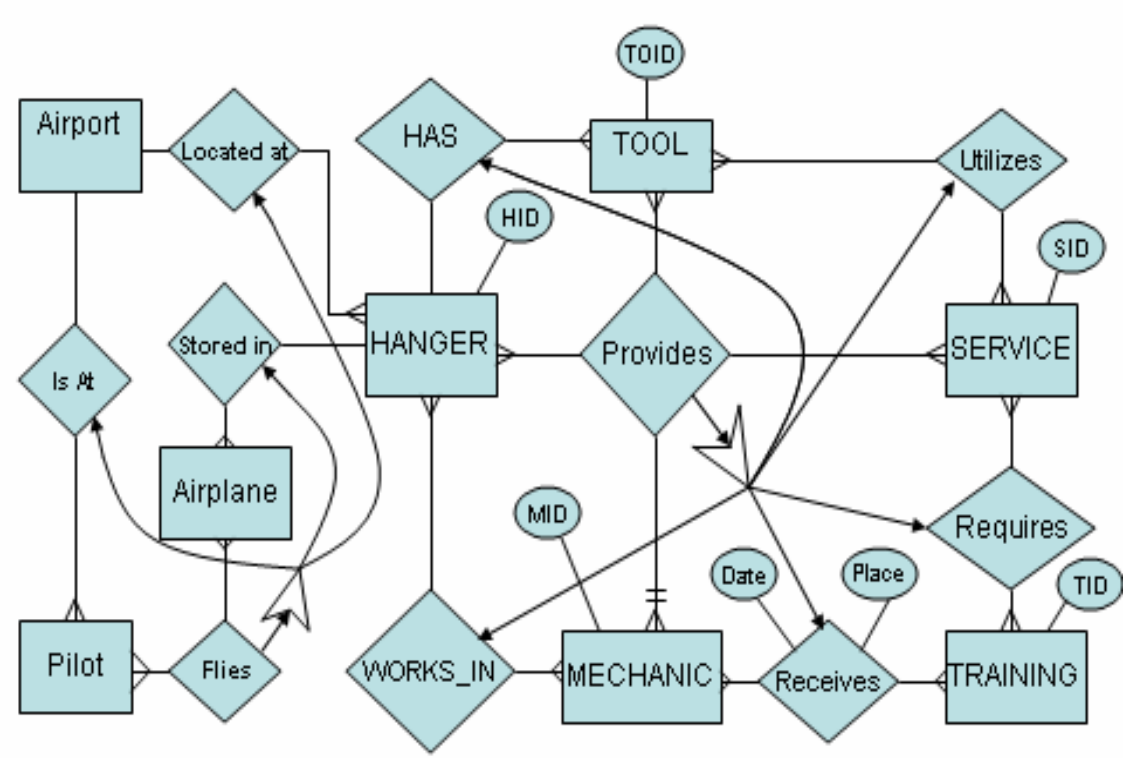

Figure 11: Mechanics + Pilots + Constraints

Conclusion

Business rules should be captured in a conceptual manner such as the ER/EER notation rather than capturing the rule in the procedural logic of a user application. Constraints need to be fully integrated with existing systems and databases[6]. This will be accomplished by the developer who implements the constraint using the conceptual model. In this paper the ER/EER notation has been used to define business rules at the conceptual level in a relational data model without specifying how the rule will be implemented. This conceptual representation is independent of the implementation of the rule. Business rules are not limited to only those constraints that have been represented in this paper. The constraints have been discussed in this paper have so far focused on the relationships between entities and could be characterized as existence constraints. The future of this work will be to examine other types of constraints such as cardinality constraints, attribute value constraints, polymorphic constraints and temporal constraints.

References

[1] The Business Rules Group, "Defining Business Rules – What Are They Really?", February, 2006, http:www.BusinessRulesGroup.org /first_paper/br01c0.htm

[2] Perkins, "Business Rules = Meta Data", The proceedings of the: Technology of Object-Oriented Languages and Systems, IEEE, 2000.

[3] J. A. Hoffer, M. B. Prescott and F. R. McFadden, "Modern Database Management", Seventh Edition, Prentice Hall, 2005.

[4] G. Ronald Ross, "Business Rule Concepts", Business Rule Solutions Inc., 1998.

[5] B. von Halle, "Building a Business Rule System, Part 1", Data Management Review, Faulkner & Gray, January 2001.
[6] http://www.transparentlogic.com/campaign/decisionenginelanding/
[7] http://www.businessrulesgroup.org/brmanifesto.htm