Universal Data Models and Patterns

By Len Silverston

According to Webster's, the term *universal* can be defined as "generally applicable" as well as "applying to the whole." There are some very common patterns that can be generally applied in data models to develop higher quality and more holistic designs. Additionally, reusing patterns can save time and effort by providing model structures that have been proven to work well, flexible alternatives that can handle additional future needs, and consistent models and structures that facilitate data sharing.

This article will describe a few common patterns that can be used to build "universal" data models, namely, patterns for roles, statuses, categorizations.

For each pattern, I will show specific and abstract ways to model these types of constructs. Which one is right? I believe that there are usually not right and wrong data models, but pros and cons of various ways to model data. More specific data models have the advantage of being more easily understood, and they show the enforcement of business rules. More abstract data models have the advantage of flexibility and maintainability because the model is much more adaptable to change. Later in this article, I will discuss when to use a specific versus an abstract pattern.

Roles

Roles represent the part that a party plays, or the "hat" that it wears. There are two types of roles: declarative and contextual. Declarative roles "declare" that a party is playing a particular role, such as a party that is identified as and declared an "employee." Contextual roles show how a party acts within the context of another entity. For example, a party may play a role of product manager within the context of the entity PRODUCT, or a party may play the role of quality assurance manager within the entity PROJECT.

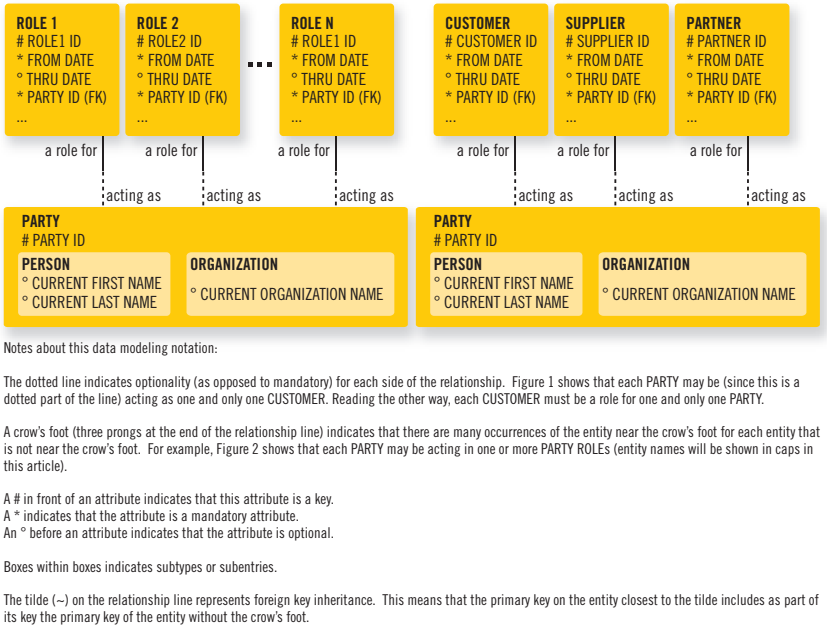

Figure 1: Specific Declarative Roles Pattern

Declarative Roles

Most enterprises maintain information about key entities such as customer, employee, partner, supplier and other roles that are played by people and/or organizations. However, people do not often realize that the role in data models is not the same as the party involved, and there may be information that is related to a party independent of their role.

When we look at ABC Corporation as only a customer, we are sometimes not seeing the whole picture. A party such as ABC Corporation may be a customer of the enterprise, and it may also be a partner and a supplier as well as play many other roles. When we confuse this and just show the party as a customer, its information may show up in many places – one for each role it plays. By showing that there is a PARTY entity (which may be either a person or an organization), which is distinguished from the party's various roles, we create models that more closely represent reality and, therefore, are more stable.

The models in Figure 1 show a spe-

cific pattern for modeling roles and parties as well as an example using this pattern. Each role may be represented by an entity and associated to a PARTY entity. For example, there may be CUSTOMER, SUPPLIER and PARTNER role entities and each are related to a PARTY – so that if the same party is a customer, supplier and a partner, their information (such as their name and contact information) is maintained once.

This specific model is relatively easy to understand, but one of the issues in this model is that one may not be able to identify all the roles that will exist over time. Therefore, as new roles are identified, additional entities are needed for the model. Furthermore, there may be common information for all roles, such as a need to track relationships that exist between any type of roles.

The pattern and example in Figure 2 shows a more flexible data model for declarative roles. The PARTY ROLE entity is an associative entity between a PARTY and a ROLE TYPE. The PARTY ROLE entity is a supertype of the various roles that could occur. It effectively "tags" a party as having one or more role. The ROLE TYPE maintains information on the possible roles that exist, for example, customer, supplier, partner, worker and so on. The left side of the model shows the pattern that any enterprise may use and the right side of this diagram shows an example that may be applicable to some types of enterprises, for example, a manufacturer of products.

The advantage to this model is that as new roles are discovered over time, the model can more easily accommodate these with the addition of another ROLE TYPE. There may also be information related to a ROLE TYPE, such as product pricing, dependent on the role that one plays or authorizations based on the role. Please note that the style of modeling shows subtypes as well as a ROLE TYPE entity because some information may be associated with a specific role (e.g., salary for an employee), while other information may be related to a role type in general (pricing discounts or authorizations related to ROLE TYPE).

Contextual Roles

Contextual roles define how a party is (or was) involved within the context of another entity. For example, there could be several roles that people or organizations play within the context of a project. A person or organization may be the sponsor, worker, manager, lead, advisor or quality assurance manager for a project. This may apply to any other entity that has roles such as order, contract, shipment, payment or product. For example, a shipment could have roles of receiver, sender, carrier, approver, entered by and so on.

Figure 3 shows a pattern and an example for modeling contextual roles using a specific style of modeling. The pattern shows that there may be any number of roles related to an entity and each role may be related using either a M:M association or a 1:M relationship. For example, there may be SPONSORs, WORKERs, a PROJECT MANAGER and a PROJECT LEAD for a specific PROJECT. These roles may have been first declared using the previous pattern or they may just be roles that are only contextual, without the need to declare them. The benefits of using this pattern are that it is relatively easy to understand, and it can enforce specific business rules that may apply, for

I believe that there are usually not right and wrong data models, but pros and cons of various ways to model data.

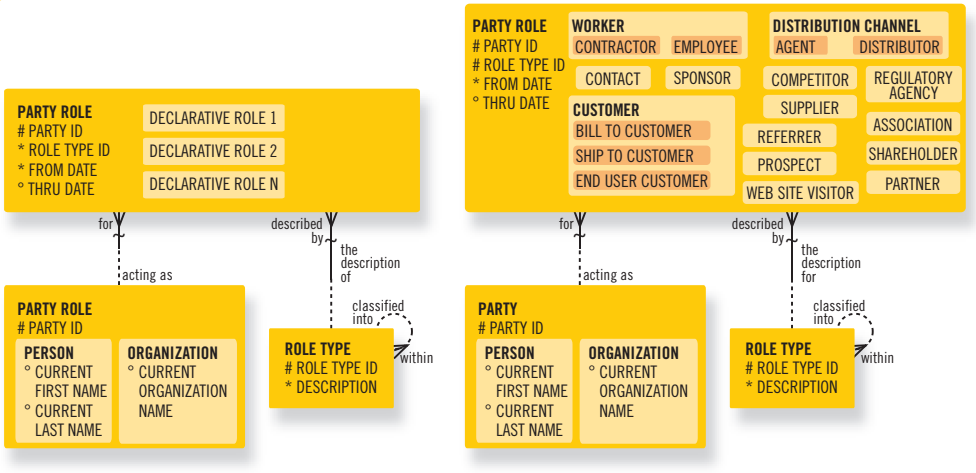

Figure 2: Flexible Declarative Roles Pattern

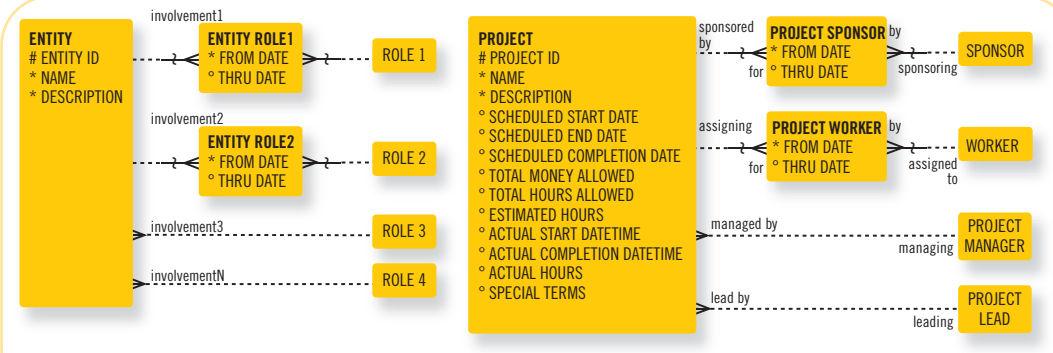

Figure 3: Specific Contextual Role Pattern

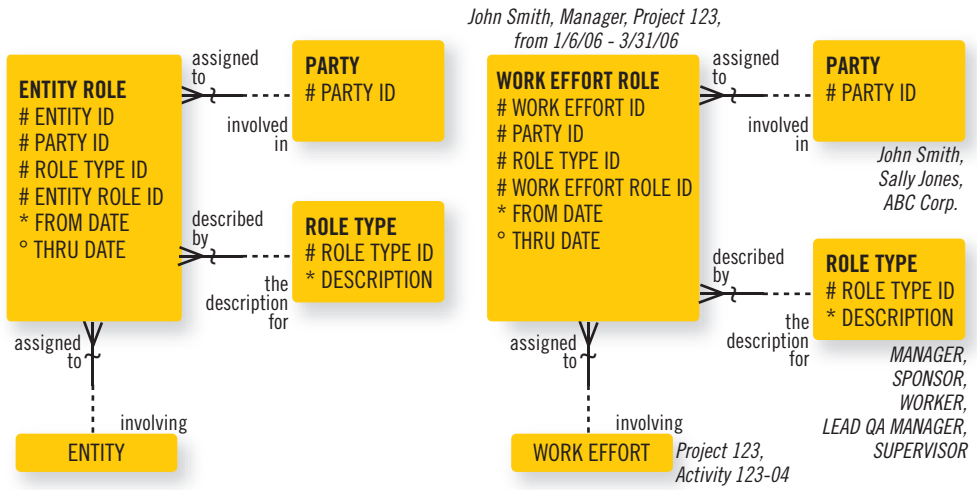

Figure 4: Flexible Contextual Role Pattern

example, that there may be only one project manager and only one project lead for a project.

The disadvantage to using this pattern is that business rules and/or processes may change over time, and the data model (and underlying database) may need to be changed. For example, what if a new role was added later for a quality assurance manager, or what if the business rule changed that allowed several project managers on the same project (at the same time or over time)?

Therefore, Figure 4 shows a more flexible pattern that allows an entity to have any number of PARTYs involved in any ROLE TYPE for any number of time frames. This pattern accommodates almost any type of new modeling requirements that could occur, including new roles that could be added over time, changes to business rules and maintaining history over time.

However, the drawbacks of using this pattern are that it is much more abstract, and therefore more difficult to understand. Also, this model doesn't enforce specific business rules, for example, if a rule was needed that there can only be a single project manager for a project. However, if we are seeking a stable, solid foundation for our databases so that new business rules do not result in changes to the underlying data structures, then this pattern may be ideal.

Statuses

Any time a date attribute or a datetime attribute is needed in a model, the modeler may want to use the patterns in Figures 5 and 6.

Figure 5 shows a pattern for modeling statuses using a specific style of modeling as well as an example that shows an application of this pattern. The model shows that an ORDER (which could be a purchase order or a sales order or both, depending on the information requirements) could include attributes for all the types of statuses that are to be tracked. However, what if a new status is needed, such as the date that a purchase order was delayed and when the new expected shipment date is expected? There also may be a need to maintain the history of these statuses, for example, to record the first expected shipment date and then the new expected shipment date.

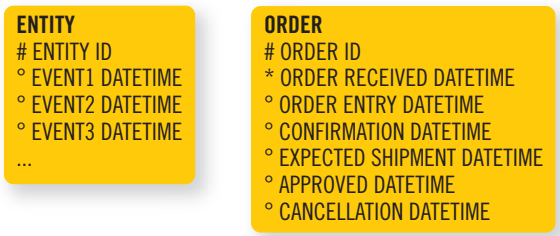

Figure 5: Specific Status Pattern

Thus Figure 6 shows a more flexible pattern and an example for modeling statuses that allows any number of statuses and accommodates the history of statuses over time. This same pattern could be applied to other entities, such as the statuses of requirements, quotes, invoices, products or parties over time.

Categories

Another common pattern that frequently occurs in data modeling deals with the classification of entities. Data models often have various classifications, types, groups, families and other ways to categorize entities. For example, there may be product groups, product types, product families, customer types, facility types, industry classifications and many other categorizations of entities. Often these categorizations are modeled independently, without a firm understanding of common ways to model these categorizations.

Figure 7 shows a pattern that an entity may simply be classified or typed using a 1:M relationship from a "type" entity to the entity. The example on the

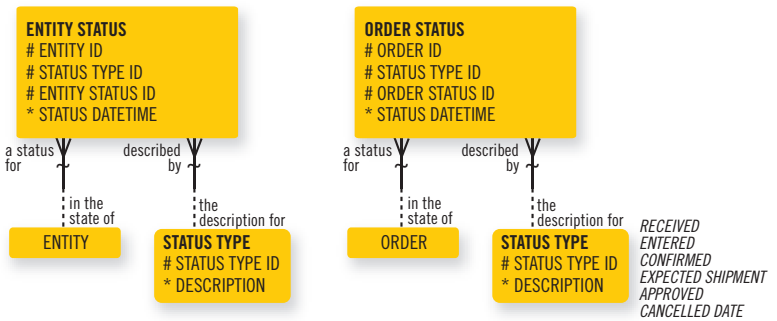

Figure 6: Flexible Status Pattern

right of the diagram shows that a product may be classified via a PRODUCT TYPE (such as a service, good or solution), PRODUCT FAMILY (such as the household products, car products or office products), or PRODUCT LINE (such as commercial or residential lines of products). Additionally, there may be types of types, as shown in the recursive relationships around PRODUCT FAMILY. For example, there may be a family of office products that is further classified as computer products, office supply products and so on.

While this pattern provides a clear understanding of the various categorizations of products, there are disadvantages of using this style of modeling. What if new types of product categories are added over time, such as the types of materials used in the products, the industry for which the product is intended or any other categorization that a creative sales or marketing person deems is needed?

Figure 8 shows a more flexible pattern and example of categorizing entities that allows for any number of classifications and any number of levels of classifications. For example, if an entity such as PRODUCT seems to have many different ways to classify it, with many different levels of classifications, then this flexible model may be appropriate. The PRODUCT CATEGORY entity could maintain all types of categories at various levels and the PRODUCT CATEGORY ROLLUP could maintain which product categorizations roll up to which higher level of categorization. For example, in Table 1 the product ID 2222 "Johnson bond paper 2000" may be classified as both an office supply product, a recurring product and a product type of "good" (as opposed to service). The PRODUCT CATEGORY CLASSIFICATION would relate this product to the lowest level category that applies (that Johnson bond paper is an office supply), and then the PRODUCT CATEGORY ROLLUP could maintain that the product category of office supplies is within the higher-level category of "office products" and that the "household products" category is within the "product family" category.

Specific versus Abstract Models

By now you may be seeing a pattern within these patterns. There are usually multiple ways to model any construct. This article covers only a couple of ways to model a few examples of patterns; however, there are many more patterns and variations that exist. One of the major variations in modeling stems from whether the model is using a specific or abstract style of modeling.

How can one know whether to use a specific pattern or an abstract pattern? One could first ask the question, "What is the purpose of a data model?" I believe that there are two key purposes to a data model: 1) To illustrate and communicate information requirements and 2) To model a sound foundation for a database design. These purposes can be at odds with each other. If the purpose is to illustrate and communicate information requirements, then the modeler will most probably develop a more specific model showing the specific needs of the business representative. For instance, the model may show the specific roles involved in a project such as the sponsors, workers, project manager and project lead with the specific cardinalities as shown in Figure 3.

If the purpose is to model a sound foundation for a database design, then the modeler may incorporate flexibility and use a pattern such as the one in Figure 4, where there can be any number of parties and roles over any time period, and thus the model is very stable and very unlikely to need changes based upon new business processes or rules. With this model, many of the rules that are in the specific model may be documented in some other format as an adjunct to the data model.

If you are an advocate of the Zachman Framework, you may recognize that there may be different audiences for a data model which results in the need for two models: a model for the owner/business representative and a model for the designer/architect. The model that one develops for an owner or business representative would most likely be a specific model, and therefore one could use the specific patterns to illustrate and

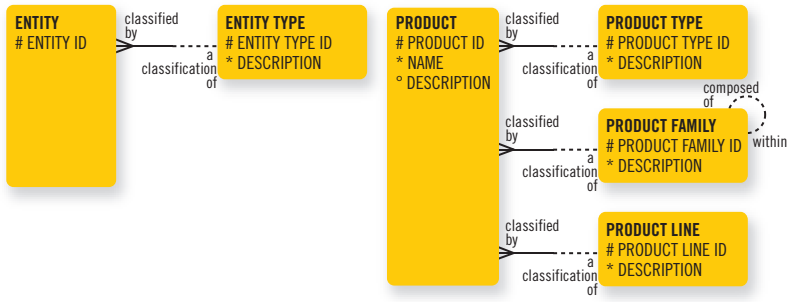

Figure 7: Specific Classification Pattern

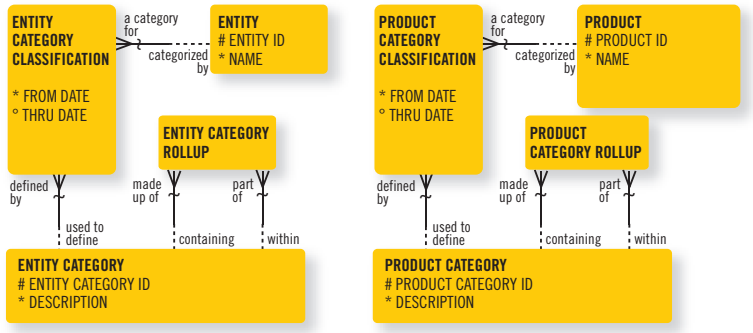

Figure 8: Flexible Classification Pattern

communicate their needs. The model for a designer or architect would most likely be designed for flexibility and adaptability to change, reducing maintenance costs. Thus, the more abstract and flexible patterns could be used to develop this type of model.

To maintain a "business data model" and an "architecture data model" and to map and cross-reference them can be quite a bit of work. A possible solution to this is to incorporate both specific and abstract patterns into the same model as shown in Figure 9. Views can be created to show specific aspects of the model as well as abstract aspects of the model. For example, one could show a view of the specific relationships of various roles to work effort, namely, sponsors, workers, project managers and project leads in order to validate requirements. Then one could show an architectural view of the model showing that a work effort (which could be a project, activity, task or any other unit of work) may have any number of parties with any number of roles associated with it over time.

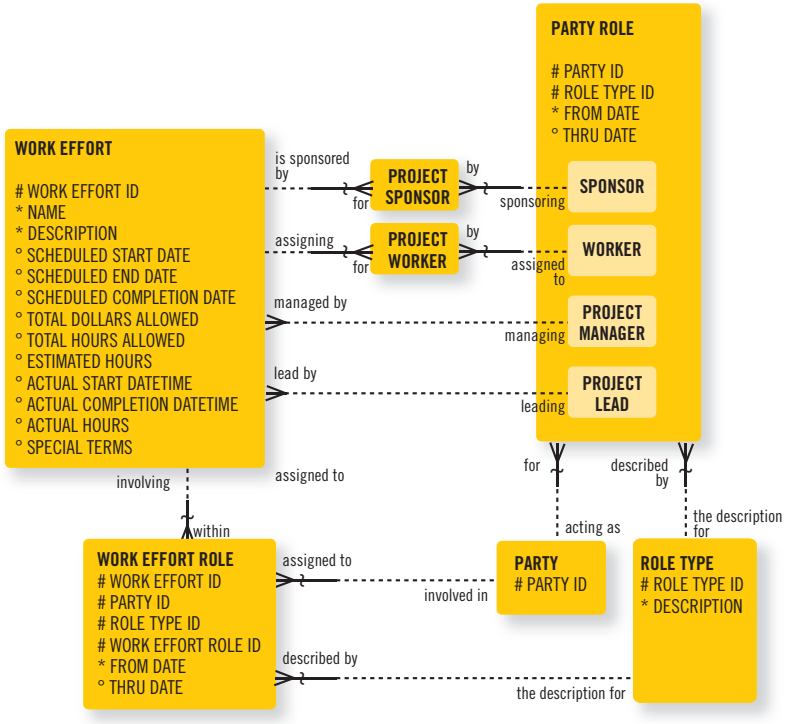

Figure 9: Specific and Flexible Contextual Role Pattern Together

A Universal Approach

Most data models have a need to maintain very common types of data such as statuses, categorizations and roles that various parties play. There are several ways to model these recurring data constructs, and the modeling method often varies by how specific or abstract one chooses to model the data. By providing several effective alternatives for modeling each pattern (including specific and abstract styles) and criteria for making choices between these alternatives, we can develop more integrated and much higher quality data models in shorter time frames.

Len Silverston is the best-selling author of **The Data Model Resource Book** *series, which describe more than 230 reusable data models and which was rated #12 on the Computer Literacy Best Seller List. He is an author, consultant and speaker with more than 24 years of experience helping organizations integrate their data and systems. Silverston has been a contributor for DM Review and a frequently invited speaker at many international conferences. He is the winner of the DAMA International Professional Achievement Award for 2004. Silverston's company Universal Data Models provides consulting, training and software to jump-start data modeling and data warehouse design efforts while increasing design quality.*