Modeling Relationships

Welcome to Lecture on Modeling Relationships in the course on Healthcare Databases. In this lecture we are going to cover two types of relationships, namely, the subtype and the many to many relationships. We have already referred to these modeling techniques in previous sessions, here we review in more depth.

A few definitions could help. A hierarchy is a collection of super and sub entities. A super entity is the broadest definition of several sub entities. We sometimes refer to this as super type, others have also referred to this as super data class. A sub entity is an entity that inherits its relationship from another entity. It may be referred to as sub type and sub data class.

Relationship describes the link between two entities; it is usually described as the shared attribute among two entities. Sometimes the word linkage is substituted for relationship. Thus one might say a data class linkage when one intends to say the relationship of an entity. These plethora of terms are needed to distinguish the database design at the conceptual level (e.g. entity and their relationships) from the physical level (tables, data classes and linkages).

In general, the relationship between two entities can be identified through constructing a sentence containing the names of the two entities as subject and object and a verb phrase in between. Examples of verb phrases are: "has," "contains," "visits," "prescribes," "travels with," etc. The two entities "Patient" and "Clinician" can be made into a sentence such as "A patient visits a clinician." When such sentences can be constructed, then the two entities have a relationship described by the verb-phrase in the sentence. Finding logical sentences that connect two entities together is an easy way of identifying relationships.

There are also other more specific ways of checking on possible relationship between two entities. One could possibly ask the following questions:

1. Is one of the entities a look up table used to provide menu items for an attribute of the other entity? For example, the entity Gender is a look up table for the Gender attribute in the entity Patient. It is also a look up table for the entity Clinician.
2. Is one of the entities a subcategory to another? For example, the entity Patient is a subcategory of the entity Person. Clinician is a subcategory of Person. This type of relationships may also be identified with the verb phrase "is a". Thus, we might say, a Patient "is-a" Person.
3. Do two entities share an attribute? For example the entity Visit includes an attribute for the patient's identity. So does the entity Patient. Then it is likely that the two entities are related. Another way to ask this question is whether the identity of one entity is needed as an attribute of another entity? In our previous example, we need the patient's identity to describe who has made the visit. The information is essential to the description of the visit. Therefore, the two entities are related.

A relationship is documented by listing the following information:

1. The name of first entity
2. The name of second entity
3. The verb phrase describing the relationship
4. The cardinality of the relationship (one to one, one to many or many to many). The cardinality of relationships was discussed in an earlier lecture.

Most entities have relationships that require no additional modeling. But some entities have relationships that reveal the need for a third entity. There are two types of relationships that require more modeling: hierarchical and many to many relationships.

Hierarchical Relationships

In our daily life we are confronted with a myriad of objects. When we refer to these objects we most likely use generic terms. We speak of tables, vehicles, clothing, etc. These terms are abstractions of the actual instances we see and handle or use, and the reason we can use a generic label for some particular object without getting confused is because we know which characteristics the object must have in order for it to be referred to as a "table" or as a "vehicle", etc. Also from our daily experience, we know that these generic terms can be made more specific. For example we can narrow the scope of the term "table" by saying "dinner table", or by saying "sedan" or "pick-up truck" or "SUV" when we need to further characterize the kind of "vehicle" we have in mind.

The process of abstracting from narrowly scoped terms into terms of broader scope is a generalization process. The general term contains only those characteristics that are common to all the more specialized terms. Conversely, the specialized terms contain characteristics that do not apply across the board, but specifically to that subset. This process of starting with a general term and narrowing its scope is a specialization process.

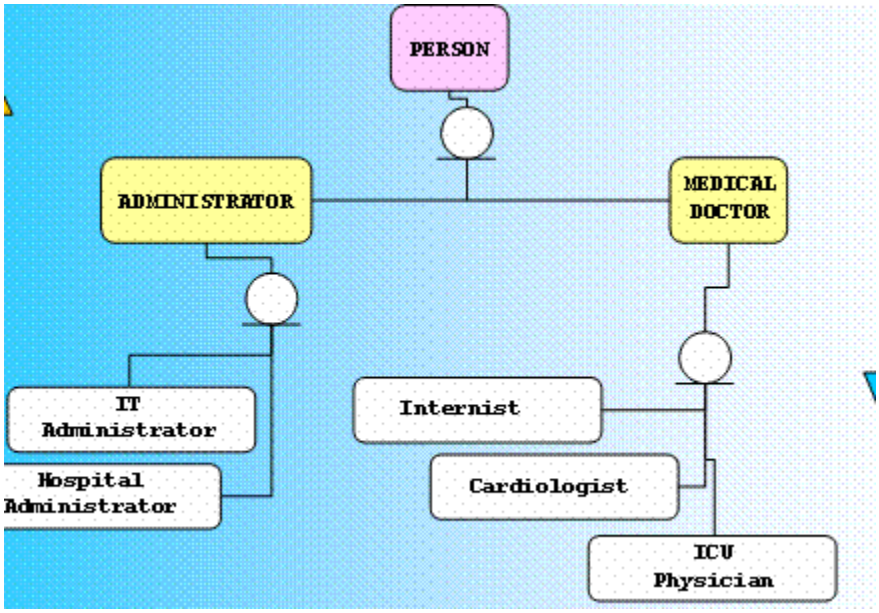

Figure 1. Generalizations and Specializations in an Information Model

There can be multiple levels of generalization. If we are in a health care environment we may be dealing with physicians that work in the intensive care unit, as well as those that handle heart problems or specialize on problems related to the internal organs. Figure 1 shows that medical doctor has three subtypes and the administrator has two subtypes. In an information model each of these subtypes will have characteristics that apply only to the type, as well as characteristics that are shared across all types. An Intensive Care Unit physician has features that only matter for this type of physician as well as features shared by all medical doctors.

Similarly, in a hospital we may have individuals who administer the IT resources whereas others may be in charge of the day-to-day operations of the hospital. As data classes we could say that the IT Administrator and the Hospital Administrator are both specializations of the the more generic data class of Administrator.

And as we said a moment ago, we could further abstract both the data class for Administrator and for Medical Doctor and view them as specializations of the more general data class Person, which, for example, the Human Resources department handles within their system.

In a database, we may have both general data classes as well as specialized ones. When this is the case, we may ask ourselves whether it make sense to bring them under a consistent subtype hierarchy. Oftentimes, introducing a subtype hierarchy can simplify the information model. Here are the rules for creating super types:

- Super-type should have all the attributes that are common to all the subtypes
- There should be a discriminator attribute for navigating to sub-types
- When possible entity relations should be made at super-type

The super-type, i.e. the broadest entity, should have all the attributes that are shared across the sub-entities. Each sub-entity should have an attribute that makes it different from other entities. The attributes maintained in the sub-entities should be mutually exclusive. In other words, the attributes that describe a physician should be kept at the super-entity, perhaps called "Physician." The attributes that describe an intensive care physician will be kept in the sub-entity called "ICU Physician." The attributes that describe a cardiologist will be kept in the sub-entity called "Cardiologist." In a proper hierarchical relationship the ICU entity and the Cardiologist entity share no attributes, if they did the shared attribute would be kept in the super-entity called Physician.

The relationship between a super-entity and a sub-entity is always represented by the verb phrase "is-a". Thus the diagram in Figure 1can be read as stating that "An Administrator is-a Person" and "A medical doctor is-a Person". Similarly, we can read that "An Internist is-a Medical doctor" and "A Cardiologist is-a Medical doctor". Conversely, our information model says that in this business domain "A person is-a Medical doctor or is-a Administrator". All hierarchical relations convey the meaning implied by the verb phrase "is a."

In some business domains it may not be possible to guarantee that the instances we are dealing with can unambiguously reside in one and only one of the subtypes specified in the hierarchy. If that's the case we need to consider whether a further refinement of the subtype hierarchy can

eliminate the ambiguity. Otherwise, it may not be prudent to insert a subtype hierarchy. In other words, the introduction of a subtype hierarchy in an information model is something that needs to be weighed carefully. It is not absolutely required that every information model contain such hierarchies.

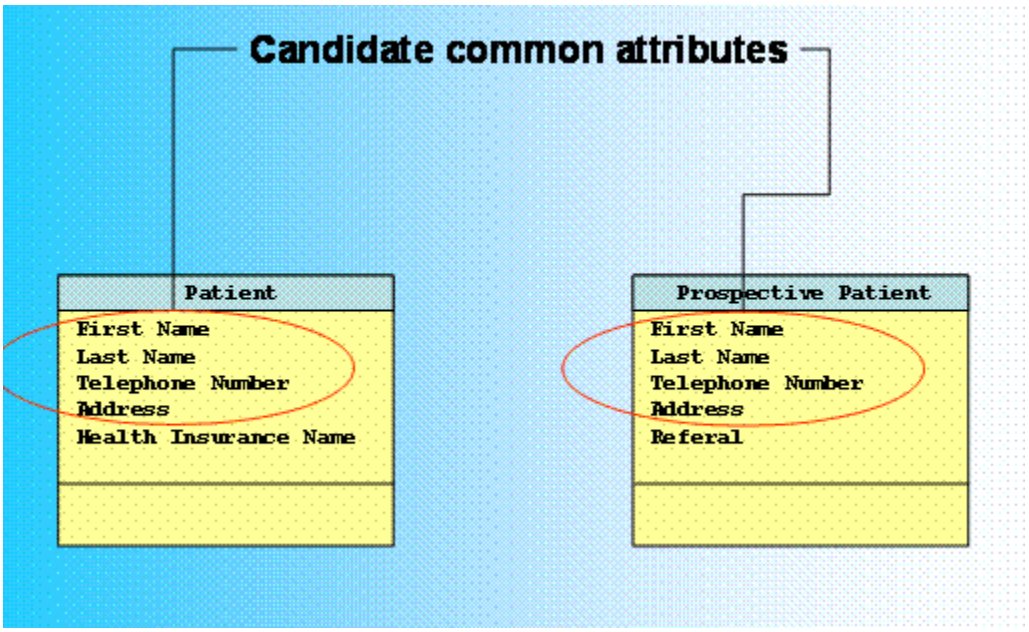

Figure 2: Guidelines for Building a Subtype Hierarchy

Let's assume for now that we have good reasons for using subtypes hierarchies. If we have been modeling our information flows according to the procedures explained in the preceding lectures we are likely to have specified many entities. When we review these entitles our analysis may indicate that some of them are good candidates for a subtype hierarchy. Figure 2 shows the steps that we should take to generate a super-entity from the multiple specialized entitles. We should migrate to the super-class all the attributes that are common among the data sub-entities.

The next step is that the proposed super-entity should have an extra attribute that allows us to navigate from the super-entity to any one of the sub entities. There should be an attribute in the Person super entity that points to either Medical doctor or Administrator, so that when we have an instance, let's say "John Doe" we can figure out whether he is an Administrator or a Medical doctor. Similarly, in the Administrator entity there should be another discriminator attribute that allows us to indicate that the administrator named "John Doe" is an IT Administrator.

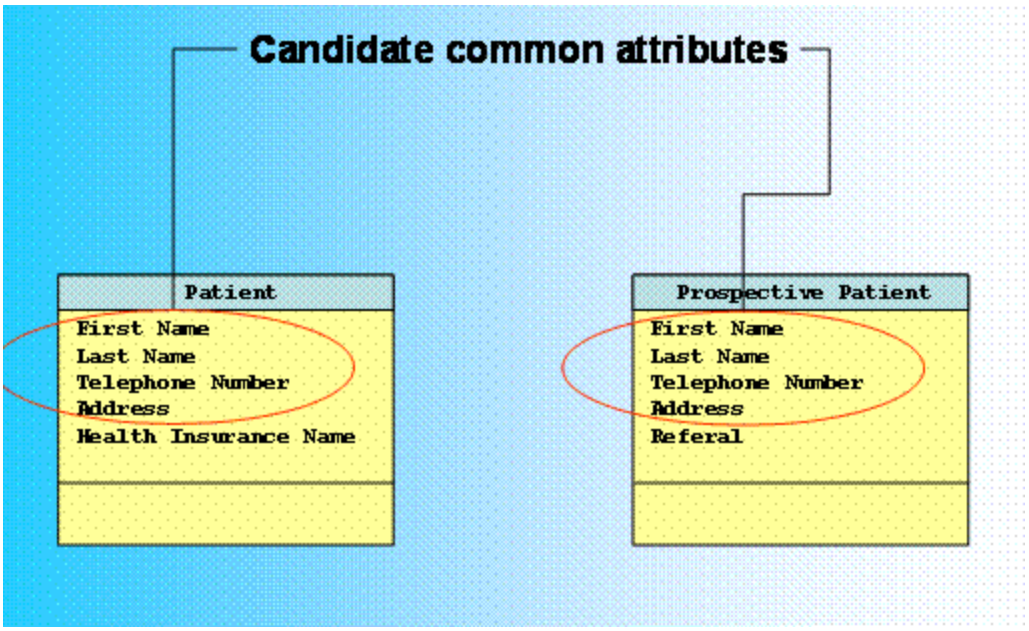

Figure 3: An Example of Two Types of Patients

So let's look at a concrete example. Suppose we need to distinguish between two types of patients: actual and prospective patients. Figure 3 shows the two entities with their corresponding set of proposed attributes. If we wanted to bring these two entities under a subtype hierarchy, the first thing we would have to do is to compare the definitions for each one of their attributes. The semantics of the attributes First Name, Last Name, Telephone Number and Address, i.e., the definitions we have documented in our information model—indicate that in fact they are the same for both the Patient and the Prospective patient entities. Since they are common to both entities we can consider them as candidate attributes that could go into a super-entity.

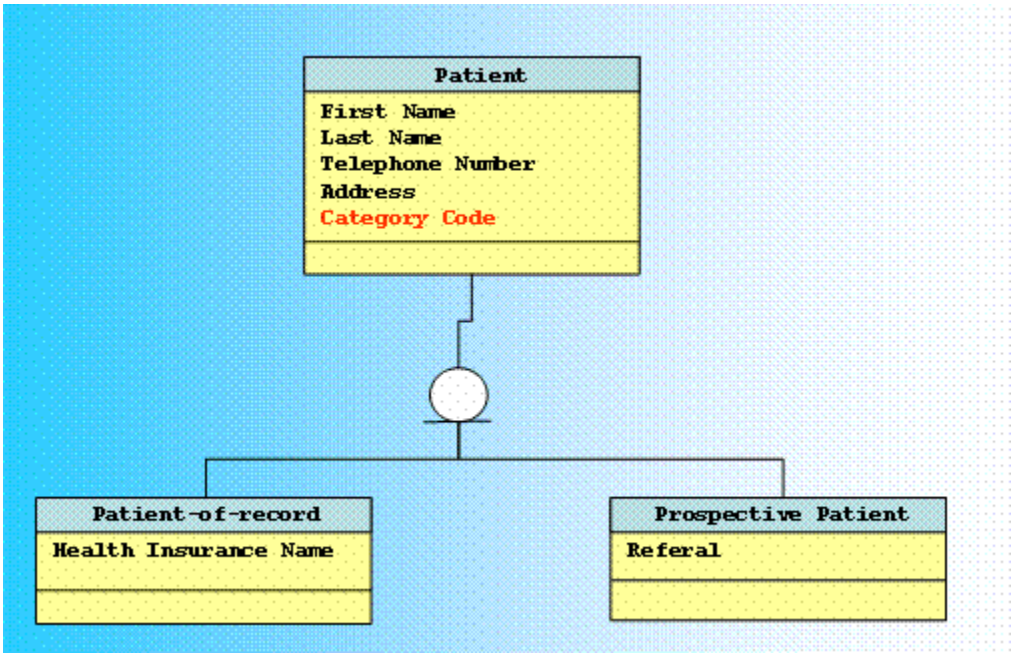

Figure 4: Revision of the Two Types of Patients to Show Super Entity

In an information model super-entities can have linkages to other entities in addition to the linkages they have to their sub entities. Of course, so can the sub entity. However, if the linkage that a subclass has to another entity could equally apply to the super-entity, then there is a big advantage in establishing the linkage at the super-entity level because it automatically gets inherited by all the sub entities in the hierarchy. This makes the information model simpler to read and also to implement later on in a relational data base. If one does not establish the linkage at the super-entity level then one could potentially end up having to establish explicit linkages for each of the sub entities, a lot of unnecessary work.

Why should we care about having subtype hierarchies? Hierarchies make introduction of new entities easier. In a highly linked information model, any new entity could require establishing dozens of new links. However, if the new entity can be brought under existing subtype hierarchies, and if we have established the linkages at the super-type level, then the impact on the overall information model is likely to be minimal. That's one of the great advantages of using subtype hierarchies: they stabilize the overall model with respect to new requirements.

Let's suppose that after the initial analysis we present our draft information model to the customer and we are told that sometimes patients of record become inactive but that the system does not purge them immediately but keeps them in the system for two years, and that, therefore, our proposed information model should support this feature as well.

Since we already have a subtype hierarchy for dealing with patients, all this means is that we need to insert a new entity in the hierarchy, include the new entity in the discriminator attribute, the Category code in the example, check that all sub entities are appropriately named and we are done. Now instead of Patient Of Record we have an Active Patient Of Record, as well as an Inactive Patient Of Record.

Figure 5 shows the new situation. Note that as we indicated before, not only have we migrated the common attributes up to the super-entity, but we have also added the discriminator attribute that would allow us to navigate to the corresponding sub entities in the proposed hierarchy. Also note that we have borrowed the name of one of the original entities and used it for the super-entity. Therefore, we have modified the name of the original sub entity to state clearly the new meaning of the class within the hierarchy. Our former Patient entity is now named Patient Of Record to indicate clearly what it means in the modified information model.

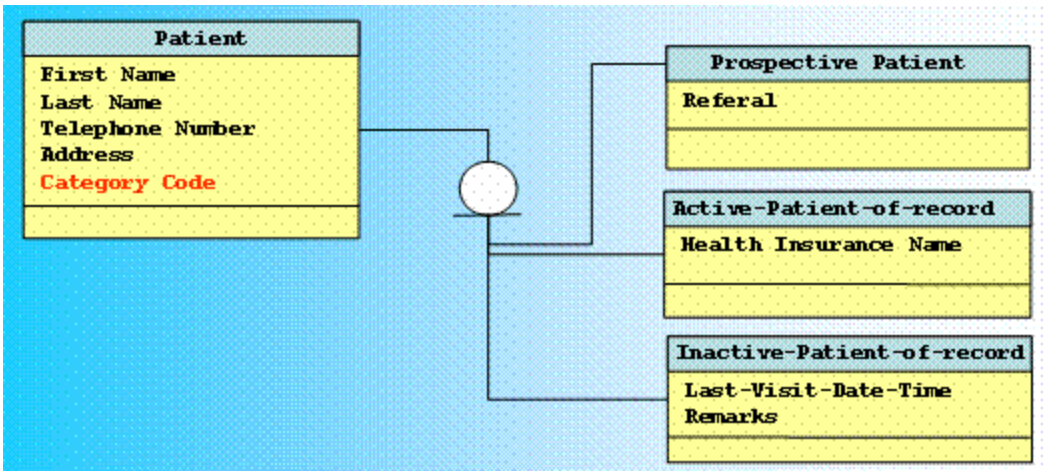

Figure 5: Distinguishing between inactive and active patients

When inserting a new sub-type we should pay attention to the same issues that were important in defining the hierarchy. Let's go over these issues and see how it applies to the Inactive Patient Of Record sub entity. The first issue is whether the sub entity is distinct form other entities. Clearly, we can say that an individual cannot be at the same time a Prospective Patient and a Patient Of Record. Nor can we say that an active patient of record is in the same type as in active patient of record. We know this because the Use Case would tell us that there is an approval process. A prospective patient would be either accepted or rejected. In addition, the Use Case may define a patient not seen for a certain amount of time as an inactive patient.

The second issue is whether the attributes in the super entity apply equally well to the new sub entity. Here again the attributes seem to fit the new sub entity, name, phone number and address are needed to define inactive patient just as much as active patients. Please note that sometimes one or more attributes may only apply to a few of the sub entities but not all of them. It is a matter of taste whether one should leave them in the corresponding sub entity, or whether one should elevate them to the super entity. If you are a purist you should leave them in the sub entities.

The third issue is whether the discriminator attribute in the super-entity applies to all sub types. Clearly it needs a new value for the new sub entity; but once this value is provided it does apply to the new entity.

The discussion of hierarchies should make the importance of good entity and attribute definitions clear. Many a headache can be avoided by spending time up-front generating good definitions, and basing all modeling decisions on them rather than on more superficial kinds of analysis.

Modeling Many to Many Relations

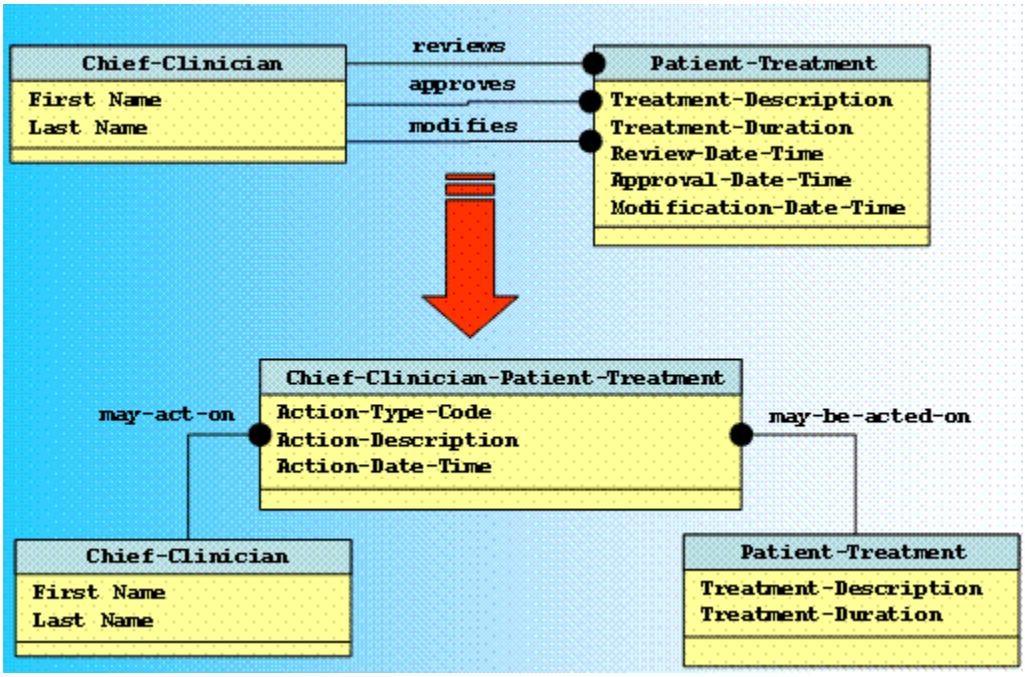

Figure 6: Resolving Many-to-Many Links

Many-to-many relationships between two entities suggest the need for a third entity not yet specified. In Figure 6, a clinician provides many different types of treatments and a particular treatment can be offered by many clinicians. The two entities have many to many relationship. Sometimes, it is not enough to say that two entities have many to many relationships but it is necessary to name and distinguish these relationships. In other words, the requirement may be not only that one entity is related to another, but in what way. We want to capture in the database more information about the relationship and not just the existence of the relationship. For example, it may not be enough to know that a clinician has provided a particular treatment; we may need to know when the treatment was offered. In these circumstances, we need an entity to track the various relationships and to provide additional information about them.

For another example, a clinician may be in various types of relationships with a given patient. The clinician may be the primary health care provider, or he may be only a back-up in case the primary clinician is not available due to illness or some other reason. If we only have a many-to-many linkage between the Patient and the Clinician entities, it is not possible to capture this type of information. Clearly, it does not make sense to add an attribute in the Clinician entity to indicate the fact that he is a primary clinician because the clinician may not be a primary clinician for all patients. He may be the back-up clinician for another patient. One could decompose the many relationship between Clinician and Patient by naming the various possible kinds of relationships. Instead of having a single relationship between a Clinician and a Patient one would have three or more linkages with the corresponding verb phrases "is-primary" "is-back-up", etc. This is done by creating an association entity or association class. In it we will name the various relationships.

Figure 7 shows how the many-to-many linkage between the Patient and Clinician entities can be replaced by two new linkages and a new entity, namely, the Patient Clinician Association entity. Now we can easily state the fact that a Patient may have multiple Clinicians, each with its specific type of association, as well as the date and time when such association became effective.

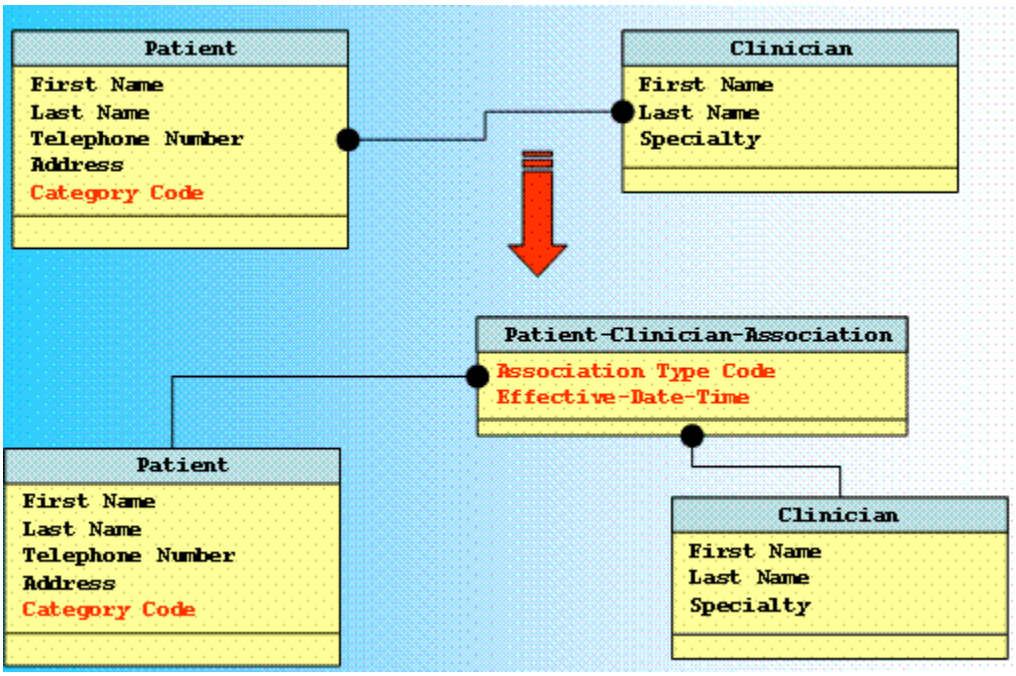

Figure 7: Relationship between patients and clinicians represented in a separate entity

Generally speaking, it is always a good practice to resolve all many-to-many linkages between data classes prior to moving to the physical implementation of an information model because relational data bases do not support the many-to-many linkage in its original form.

In some information models the number of roles can be literally in the hundreds. This would mean that we would need to establish as many linkages between entities as there are roles. Clearly this is not a very efficient and robust form of information modeling. Creating an Association Class is a simpler way of capturing and tracking many relationships.

Special Case: Linking a Class to Itself

If two data classes can be linked to each other via an association class it does not do too much violence to our understanding of information modeling if we think that a data class may be linked to itself, after all, individuals are related to other individuals, organizations are related to other organizations, electronic components are related to other electronic components, etc. In an information model the fact that "John Doe" is related to "Mary Doe" via a father-daughter relationship would be expressed by stating that the data class Person can be linked to itself. As we saw a moment ago, however, establishing a linkage for every conceivable role is something to be avoided. Instead, introducing an association class, namely, Person-Association and two linkages between it and the data class Person could give us all the functionality we need.

A self referring relationship is found in the same way as all relationships are found by making a a sentence containing the name of the entity and a verb phrase. For example, the sentence "A person is a father to another person," suggests that the entity Person may be related to itself. The verb phrase for this relationship is "father to." Of course other verb phrases is also needed, for example, a person may be a sister to another person. One could imagine an associative entity that describes the relationship between the entity Person and itself.

Use of Combined Association and Sub Types

There is also another scenario where the use of an association class to link a data class to itself can simplify an information model and make it more robust. Figure 8 shows a portion of an information model. Let's assume for the sake of argument that the model has been built to help a hospital administrator manage all the medical facilities that the hospitals under his supervision either own, use or have access to.

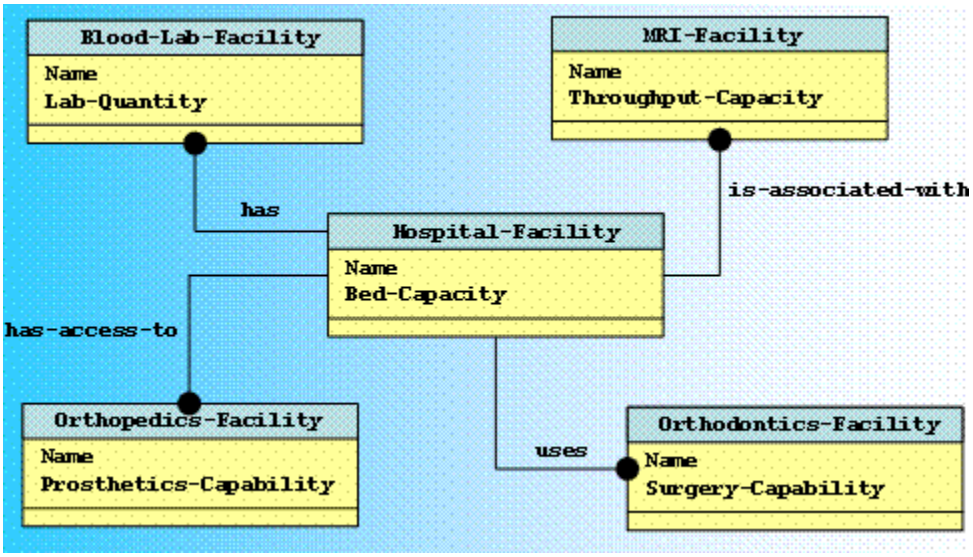

Figure 8: Portion of Entity Relationship Diagram

Without any other requirement this type of model could easily fit the bill. It is clear in terms of its meaning and correctly reflects the fact that a given hospital can have one or more blood labs, MRI facilities, Orthopedics facilities and Orthodontics facilities. But let's suppose that after the original information model is reviewed the customer indicates that not all the medical facilities that a hospital owns, uses or has access to are co-located, and that, furthermore, some of the facilities may have multiple postal addresses.

One way to incorporate the new requirement would be to add multiple attributes to each one of the entities. If the number of possible postal addresses for a facility is five, then we would need 5 attributes in each entity. If we wanted to keep track of the street name, street number, city, state and ZIP code separately, then each new data class would have to have 25 new attributes, the majority of which would never be used, since potentially only a few of the facilities would actually have five different postal addresses, most of them having only one or two at the most.

Here the use of a subtype hierarchy and an association class to relate the super-class to itself, could, quite easily and elegantly solve the problem.

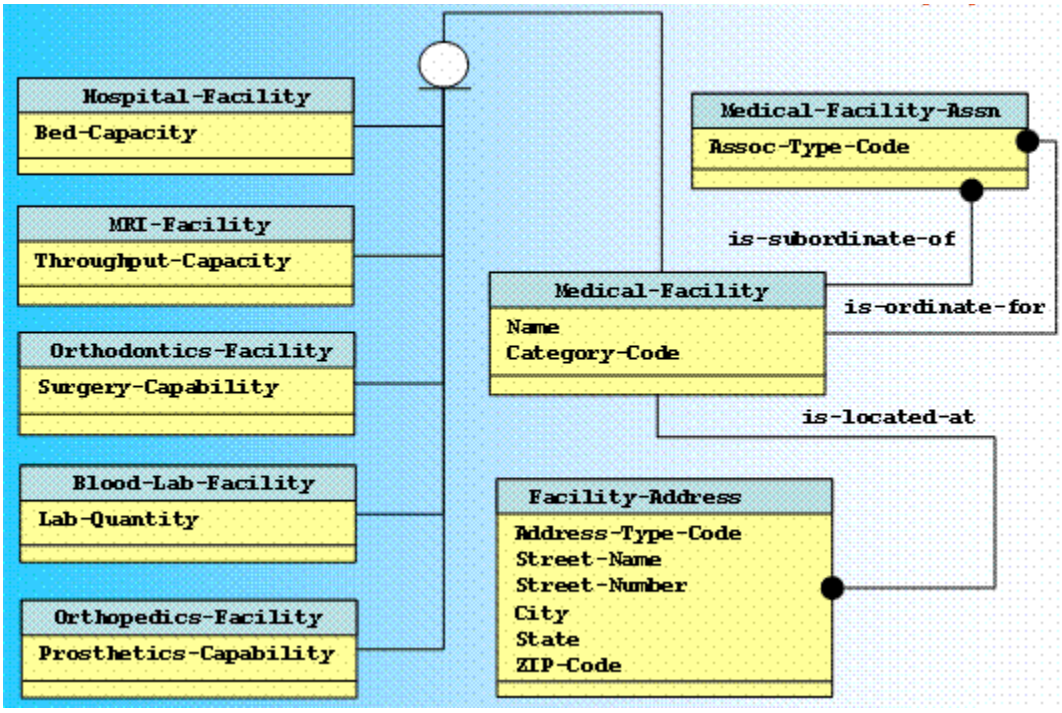

Figure 9: Example of How to Handle New Requirements

Figure 9 shows the new model recast as a subtype hierarchy for Medical Facility, with the original data classes now as its subtypes. In addition the association class Medical Facility Association now allows us to express the same semantics as before, namely that a Hospital can have one or more blood labs, MRI facilities, Orthopedics facilities and Orthodontics facilities simply by making an instance of Hospital-Facility the ordinate and relating it to the appropriate instances of the other kinds of facilities as required. Then, all we now require to be able to assign addresses to any medical facility, be it a hospital, a blood lab, etc., is to link the Medical Facility entity to the Facility Address entity through a one-to-many linkage.

Summary

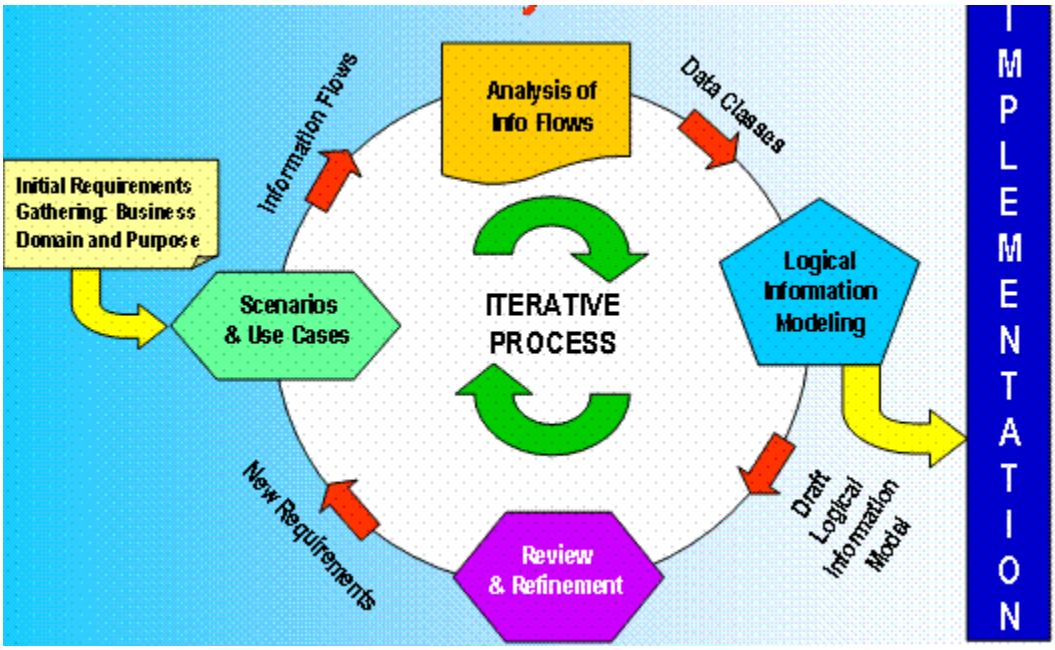

Figure 10: The Iterative Process of Information Modeling

We have seen how it all starts with the initial gathering of requirements. Then scenarios focus on decisions and contain use cases which reveal the information exchange. Use cases set the fields that should be included in the database. Entities are chosen to group fields in logical sets. Relationships are identified by making sentences using the names of two entities

The whole process is an iterative one. The first round of analysis will produce a draft logical model, which should be vetted by the customer, and reviewed by the subject matter experts. This review process is crucial. Requirements may be better formulated after the customer sees the information model and begins to understand how the expected functionality of the overall system will be implemented. The customer should be an active participant, a stake-holder, in the information modeling process.

Eventually, all requirements will be correctly reflected in the information model and then the next phase, namely the implementation of the logical information model into a physical data model can begin.