Data Normalization: A Primer

by Dennis Santoro

Revised October 16, 2002

This paper discusses the data normalization process in database design from a practical and theoretical perspective. Examples are presented to illustrate multiple approaches to the process.

Modeling the enterprise

It is usually important, especially for new computerization, to model the existing workflow as much as is practical. While this phase is an opportunity to improve workflow in the enterprise the advantages of this must be balanced against the disadvantages of the effects any workflow modifications will have on staff. A part of modeling the enterprise is to identify the data elements involved in the work and to organize it in a way that facilitates workflow while as much as possible modeling the data relationships that actually exist in the organization’s environment.

Normalization

Prior to actual system design, when developing databases, you must derive the data elements which are needed. If the organization has done strategic data planning this process will mostly involve selecting data elements from the strategic database's data dictionary. If not then it is important to identify any shared data resources which will be used and then define additional data items which will be required.

Data dictionaries (not technical but descriptive ones) should be developed at this stage if they do not exist. Although they will most likely change during normalization it is important to have a clear data dictionary of the original system to begin the normalization process. The development of the data dictionary will start with identifying relevant data elements and creating clear, unambiguous definitions of those elements. It is essential that the definitions be clear and consistent and that the organization uses them consistently. If a single data element is defined in more than one way the data can become essentially meaningless. This is particularly true of systems which will be used for important analyses.

As an example, suppose we have a contact management system. One of the contact types we would have is phone calls. On the surface, this seems simple. However, do we track a phone call that is attempted but not completed (busy signal)? Tracking the attempt may be important. Do we track a call which we complete and where we did not get the party we wanted but rather left a

message? Is it different if we leave a voice mail or a message with a secretary? Obviously all these are different from a call we complete where we talk with the intended party. And what about incoming calls as compared to outgoing? So categorizing each type of phone call becomes important and clear definitions of what constitutes each type of call becomes essential.
I will use this contact management example throughout this document. So let's begin by identifying data elements. We will

We Deliver Contact Sheet

Name: Althea Genofski, Director
Organization: Springfield AIDS Prevention Project
Address: 43 State Street
City: Springfield
State: MA
Zip: 01107

Phones: 413 293-8456 Office, 413 293-7748 fax, 413 293-5542 beeper

Contact description: Private non-profit community agency. Called on 7/1/2002 for info.

Called on 8/5/2002 to order TLC kits in Spanish and English and the AIDS info in both languages too

Order info: 8/5/2002: 10 Spanish TLC kits (NC. $3 S&H ea) 10 English TLC kits (NC. $3 S&H ea) and 10 Aids kits (NC. $1 S&H ea)

Payment received on: 9/10/2002 - 70.00

Date Last Modified: 9/10/2002

Form 1: Contact sheet

assume this system tracks contacts for sales and the orders the sales contacts generate. We will also manage the shipping data here so that sales folks using the system can tell if the sales have been shipped. (Note: This system will be simplified so as to provide good examples but a real contact management system which does this would tend to be more complex. I will focus on the contact rather than the order side of the application for this paper. Also, spaces, _,-,# should not generally be used in field and table names, especially in Paradox, but are used here for

readability.)

Let us suppose the organization currently tracks contacts on paper (or in a non database format such as word processing) using a contact form such as that in Form 1 above.

In thinking through a contact system that meets the needs above we have to think about the nature of contacts, the nature of orders, and our business model. We can then begin listing characteristics of those that we need to track. Based on the contact sheet above some of the items we would need include:

Name
Title
Address
City
State
Zip
Organization
Organization Address
Organization City
Organization State
Organization Zip
Phones # (1..n)
Phone # Description(1..n)
Category of person/organization Description(1..n)
Type of contact Description(1..n)
Dates of contact(1..n)
Order Date(1..n)
Quantity(1..n)
Item(1..n)
Price(1..n)
Shipping(1..n)
Amount paid(1..n)
Date paid(1..n)

While these items would need to be defined clearly as mentioned above, the details of these definitions are usually organizationally based. Because of that I will not go through that here but will describe specific definitions as needed. I should also make it clear at this point that the process of identifying data items, normalizing data and defining it is an iterative process. Each part affects the others and changes will take place as the process proceeds.

During the normalization process we must identify 3 basic types of database element: entities, attributes and relationships. Entities are generally real world objects or concepts. A person may be an entity. So might an order or a bill. Attributes are components that describe the entity. For example, first name and

last name are two attributes of the entity “person.” Relationships describe the connection or interdependence among entities. For example, the entity person will have a relation with one or more address entities. Relationships are generally delineated as one to one (1:1), one to many (1:M) or many to many (M:M).

The primary entity in this contact set is the person. That is the entity whom we have contact with so it is central to the database and, for our purposes here, all relationships flow from it. The items identified as 1..n indicate that there could be any number of these for any individual (home phone, work phone, cell phone, etc.)

Once the data items are derived and defined it is important to normalize them. There are 5 rules of normalization (relating to what are called "Normal Form"). Lets work the data items through normalization. Normalization involves breaking data into logical sets that model the real world as efficiently and completely as possible. This involves applying a set of rules to the data and thinking about the real world use and application of your data. The sets are groups of attributes which are descriptive of entities.

Keys and Links

Normalization also involves creating keys for each data set. A key is a unique identifier for the records in the 2 dimensional data set. This can be a single field or can be composed of multiple fields which together form a unique identifier (this latter is called a compound or concatenated key).

Normalization also involves designing the linkages between data sets. Links are always made based on data contained in the 2 or more related data sets. The data must exist in both data sets for the link to be created.

In cases where the link exists on values that are unique in both data sets a 1:1 relationship exists. Since the value can only exist once in each table and the relationship is based on that value only one record in each set will match one record in the other.

In the case of the link values being unique in one data set but not unique in another we say we have a 1:M relationship. Since the value is based on a unique value in the first set but a potentially repeated value in the other set the unique value in the first set can match many records in the second set.

In cases where the link exists on values which are not unique in either set we say we have a M:M relationship. Since the values

can exist repeatedly in both sets the value associated with several records in one set may also be associated with multiple records in the second set. In general practice, however, most many to many links are actually implemented by using two separate tables that have a one to many relationship to an intermediate table which is generally referred to as a relationship table.

There are 2 schools of thought about record keys. One is that it should use real and meaningful data as part of a record key. The second is that all record keys should be meaningless integers (which is generally referred to as a surrogate key). I generally come down leaning towards the meaningless integer (surrogate) key but there are times when meaningful data keys make more sense. I think in some circumstances, keys which are derived from real data, or which have real data embedded in them (e.g. serial numbers which tell you something about the manufacturing location and product run) can be useful. In many cases, however, integer keys are more efficient and secondary indices can be used on fields with real data instead of using them as keys. In the following examples I will use a mix of both methods.

A common problem with meaningful keys is that a designer picks some value they expect will never change. Later, because of business rule changes, real world events or misinformation about the nature of the value, the key value does change. This can be a serious problem for data integrity and data audits. Key values should be selected such that they should never change. Even such seemingly invariant values as a US social security number can end up being changed because of such things as identity theft. This makes it a poor choice as a key. Surrogate keys, on the other hand, will be unique by definition and, since they are meaningless, they will never need to be changed. This is a very strong argument for them.

There are also 2 basic approaches to linking with keys. In one, the key of parent is repeated as part of the key of the child. In cases of a 1:1 relationship this is fully acceptable. In the case of a 1:M relationship the parent key is repeated and the child adds one or more data elements to the parent’s key. This forces the use of compound keys and can be much less efficient and require more redundant data. The second is the key/foreign key approach in which the child records have a unique key of their own which has no data from the parent key and also contains the parent key as an attribute, but an attribute (field) which is not part of the child’s key. Both approaches will be illustrated below, but generally the key/foreign key approach is more efficient as well as more robust from a data integrity point of view and compound (concatenated) keys are best avoided.

Among the reasons why concatenated keys are best avoided are the fact that the growing key size as you proceed into deeper levels

of the database increase the size of your keys and amount of data redundancy and data transport. You can end up in situations where you have more fields in the key than in the actual attributes of the key. Compare this to carrying a single foreign key field to the next child table. In addition, in some database software this will potentially make your indices more fragile and susceptible to damage. Loss of data integrity may result. There are additional reasons but they will not be addressed in this paper.

Rules of Normalization

There are 5 rules of normalization. Few databases go through progressive normalization such as will follow. As one gains more experience one begins to see relationships and jump to higher levels without going through the others first. Also, as we progress through each normal form here we will not take every data set in the database to the next normal form but will illustrate it with certain sets.

The first rule of normalization is to **Eliminate repeating groups**. Make separate sets for each group of related attributes and give each a primary key (first normal form).

In Figure 1 I have taken our data elements and broken them into groupings to eliminate repeating groups and we have created a key for each group. I have also created a relationship for each data set to the NAME data set. (Note: overlaps between tables indicate they are related by a link, an * after a field name indicates the field is an element of the key in that table and (FK)after a field name indicates the field is a foreign key in that table which relates it to the parent.)

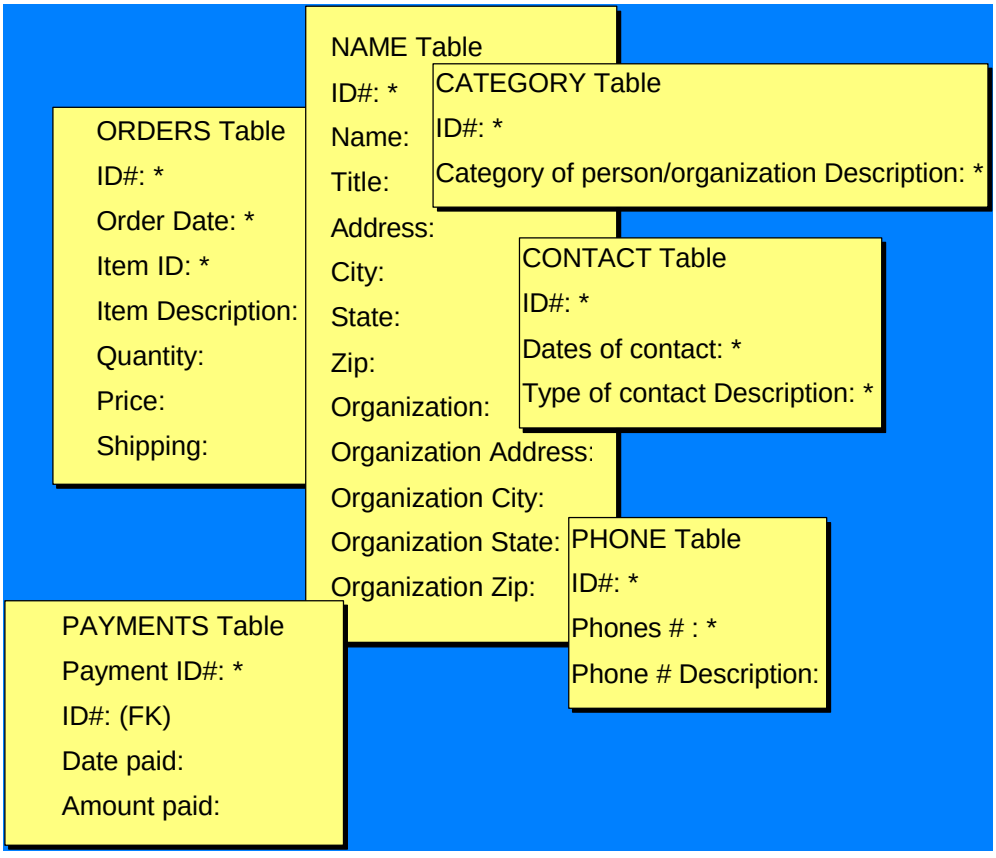

Figure 1 First Normal Form

I have taken the data related to the individual that does not repeat and placed it in a table called NAME. The data items we had previously

identified as 1..n have now been broken out into separate tables with a 1:M relationship to the NAME table records. For example, rather than have separate items in the name table for Work Phone, Cell Phone, Beeper, etc. and have many of them be blank, we have broken the Phones out into a table of their own. We now can store only phone types the person has.

We have repeated the ID# which we created as a unique identifier for NAME records in most of the related tables to provide a link with all the subsidiary (child) tables. In some cases we have used that as the first element of a multi part key but have added enough other elements in the child table key to make each record in the child tables unique (these are concatenated keys). For example no individual will have the same phone number twice (the key is ID# & Phone#) . This will have to be partly mitigated by having a phone description that includes individual use lines and multi use lines (e.g. we will need separate categories for Work, Fax and Work & Fax)

Note also that a better approach in most circumstances is the Key/Foreign Key approach. In this approach the key of the child table is another unique integer or other type of key if we are not using surrogate keys. The link between the parent and child records in the tables is made by using a secondary index on the foreign key, which is the key value from the appropriate record in the parent table. This foreign key value is not part of the key in the child table, however, as concatenated keys are generally best avoided, especially for linking purposes as described above.

This Key/Foreign Key approach is illustrated in the PAYMENTS table (in Figure 1 above) where each payment has a Payment ID# Key and the link is on the Foreign Key ID# which is the key of the Parent table but not part of the key in Payments.(Note: You may find issues above which seem presently unresolvable such as what happens if a person is involved with more than one organization. These will be addressed as we continue.)

The second rule of normalization is **Eliminate redundant data**. If an attribute depends only on part of a multivalued key remove it to a separate table (second normal form).

Note that in the Figure 2 (below) we have separated the ORDERS Table to ORDERS and ITEM DESCRIPTION Tables. Since our item numbers are unique and the descriptions are only related to the item number there is no need to have both in the table. Note also the CONTACT and CONTACT TYPE tables and CATEGORY and CATEGORY LISTING tables have been created from the original CONTACT and CATEGORY tables. While this is not strictly necessary we have decided to do this because codes are easier to type and remember accurately and codes will take less storage space than full

descriptions. By doing this we create small tables with lookup values which are related to the other tables. These tables will contain a finite (at any point in time) list from which contact categories and organization categories can be selected at any point. Only the code need be stored in the transaction table (the addition of a category in this case).

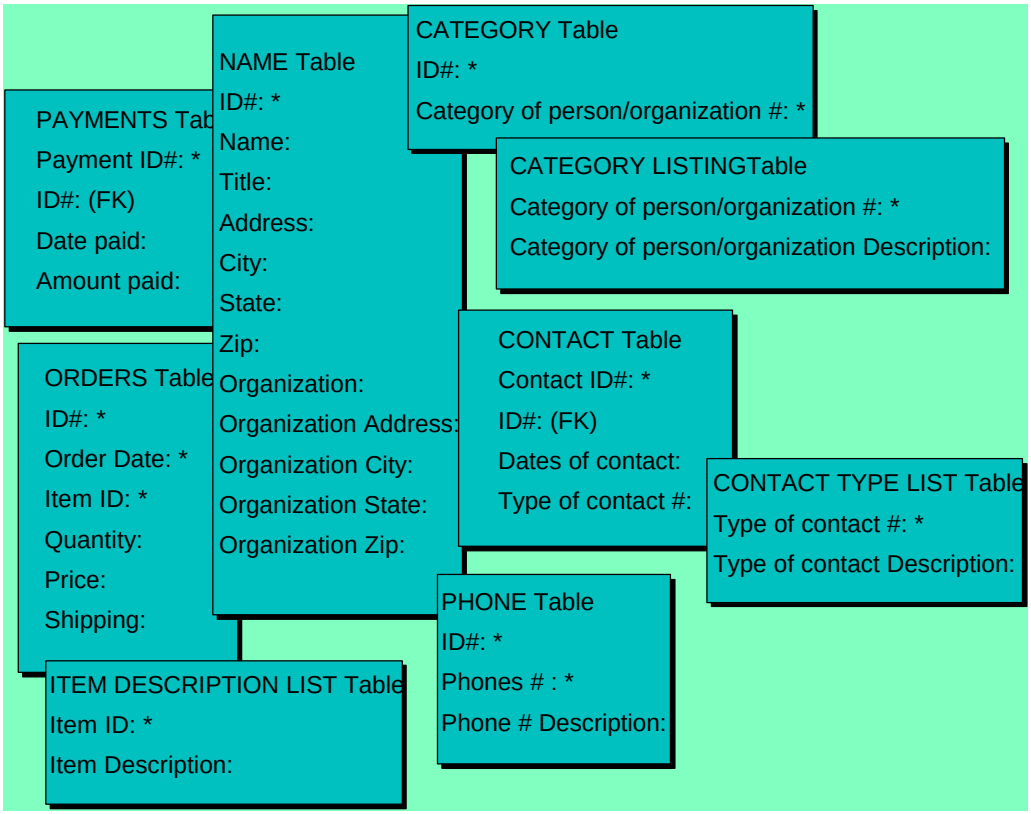

Figure 2 Second Normal Form

Note also that the contact table has also been converted to the more efficient Key/Foreign Key format for linking. Again, this gives us a single key for the table while maintaining a 1:M link with the parent through the foreign key which is the parent key value of the linked master.

The third rule of normalization is **Eliminate items not dependent on the key**. If attributes do not contribute to the description of the key remove them to a separate table. Usually this level of normalization is adequate(third normal form). Note also, that when we say key in this context we mean the unique entity itself, not necessarily the value of the key field as it may be just a meaningless integer defining uniqueness.

Note that in figure 3 we now have an ORGANIZATION LIST table. Organization was not dependent on the key of the NAME table although it is related. Many people may belong to one organization. In fact any individual may also belong to more than one organization we deal with although that will be somewhat less likely than the other case. So I have broken out the organizations to a table of their own. We still need a relationship between the person in NAME and the organization(s) to which they belong.

Organizations now also need a unique key of their own. We have options for how to handle that in third normal form. If we wish to capture organizations only once but multiple people may be associated with them, then we can place the Organization ID field in the NAME table. This allows us to place the link between an organization and many individuals while still only entering the organization once. If instead we wanted to have one person related to multiple organization records we could place the Person ID field in the ORGANIZATION table. Then a record in NAME could be related to as many records in ORGANIZATION as we want. Note in either case the foreign key (the one that comes from the other, related table) is not part of the key for the table to which it is foreign but is all or part of the key in the other table. If, however, multiple people and multiple organizations need to be related to each other (M:M) then we can not capture that relationship in this way and must add a relationship table. This is demonstrated in the CATEGORY table as discussed below.

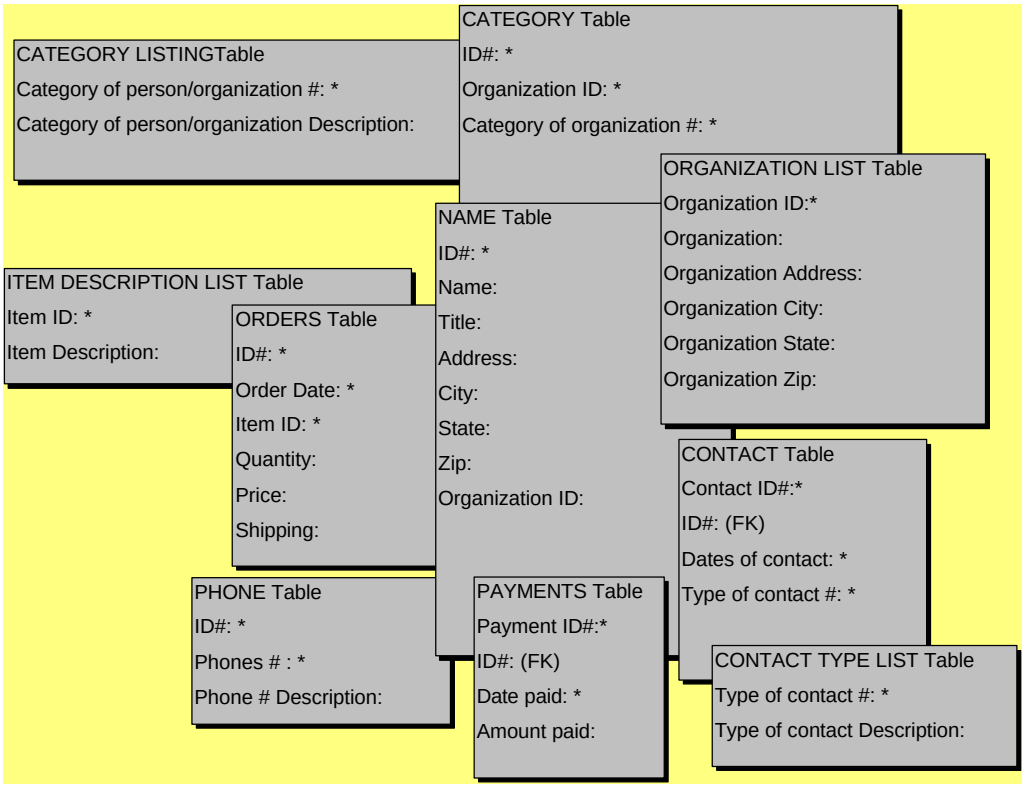

Figure 3 Third Normal Form

Once we break the organizations out it has ramifications for the CATEGORY table. Since previously we were categorizing a table with both people and organizations in the same entry we now have to decide whether to categorize the people, the organization or both. We also have to decide whether we will do this from an integrated list or from 2 separate lists. In this case we chose to have the lookup table shared since the categories may be shared. For example, you might categorize the organization as non-profit and community hospital and good sales prospect. You also might classify the person as good sales prospect and purchasing administrator. Since some of the classifications could be shared by people and organizations it could then be one lookup

table. In Figure 3 we demonstrate having an integrated category lookup table shared by the CATEGORY table for categorizing both and the ORGANIZATION table and the NAME table. This relationship, however, has certain problems associated with it.

While we could have captured the relationship by placing a Category ID in to both the ORGANIZATION table and the NAME table this would allow us also only one category per person and one per organization. We are now capturing a many to many relationship. In this instance any organization can have any number of categories and any name can have any number of categories. The other ID in the table will be blank (Organization ID is blank if we are categorizing names and ID is blank if we are categorizing organizations). This is not a good design, and in fact would not be done this way in a real system, but it is illustrative here. In a better design we would have 2 relationship tables using one lookup list and each relation table relating the list to one of the tables, organization or person.

The fourth rule of normalization is **isolate independent multiple relationships**. No data set can contain two or more one to many or many to many relationships that are not directly related (fourth normal form). Note again the CATEGORY table in Figure 3. In this table we are capturing 2 many to many relationships. Any

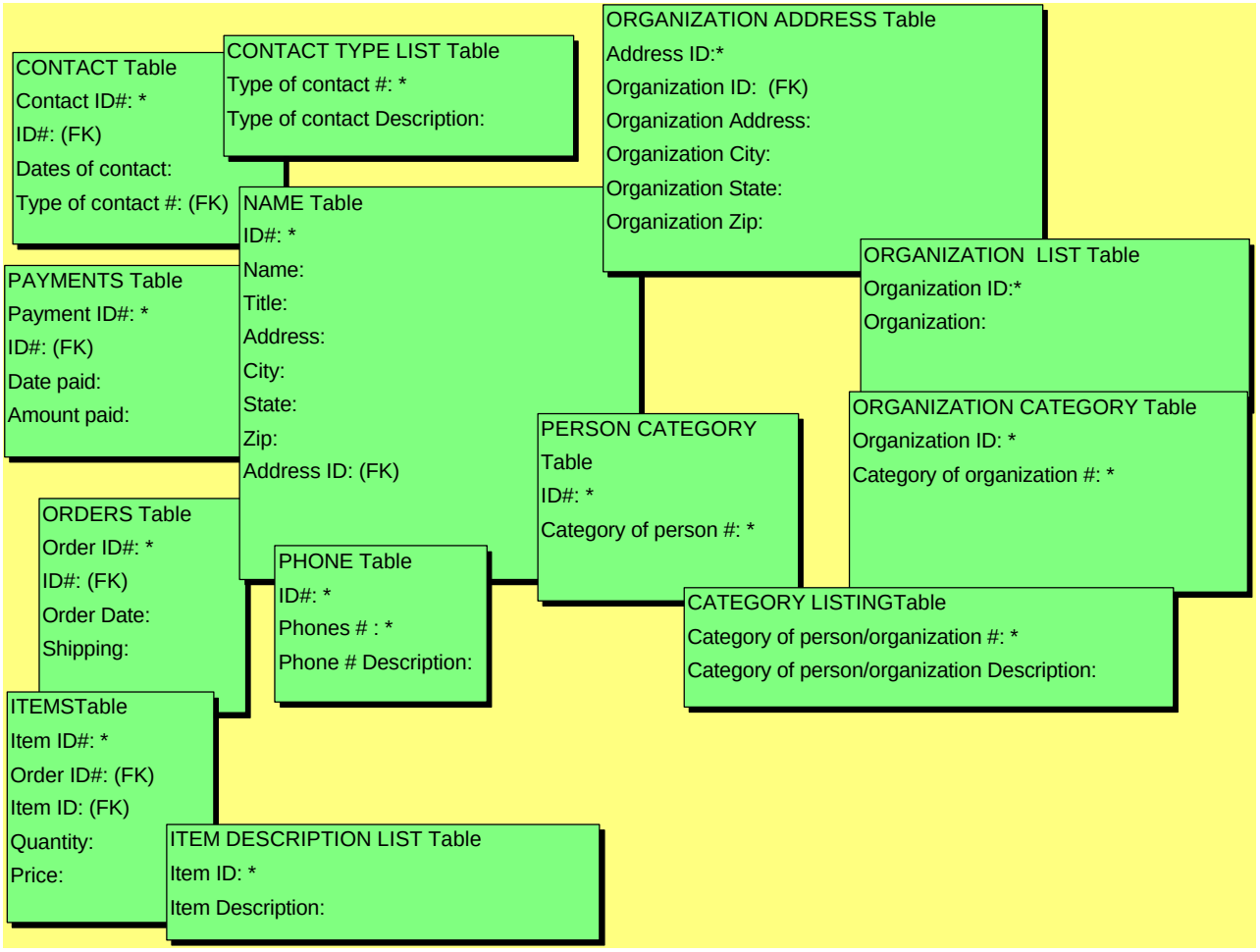

Figure 4 Fourth Normal Form

organization can have as many categories as we wish and any name can as well. And any category can be related to as many names or organizations (or both) as we wish. This violates the rule.

To meet fourth normal form I have now separated the two many to many relationships in Figure 4. We have still retained a single CATEGORY LISTING table but we now have 2 separate tables for the relationships. The PERSON CATEGORY table now contains the Person ID and the Category of Person #. The ORGANIZATION CATEGORY table now contains only the Organization ID and the Category of Organization #. We can still capture the relationships we could before but in a cleaner way and without carrying the baggage of the empty field.

I have made other changes here to the structure although these are not, strictly speaking, about fourth normal form. While one person is most likely only involved with one organizational address the likelihood of some organizations having more than one address is high. So we would have to redundantly store the organization information and all its categorization if we had an organization with more than one address and we had people related to those separate addresses.

Technically this is a third normal form issue again since the organization address information is not dependent on the key of the ORGANIZATION table. So I have broken out the organization address. In that ORGANIZATION ADDRESS table we now have an Address ID. That Address ID is now placed as a foreign key in the NAME table and the Organization ID is now place as a foreign key in the ADDRESS table. This allows any person to be at one address and any address to have many people and any address to have only one organization but any organization to have any number of addresses. Notice that while moving to fourth normal form in one area we have also addressed a third normal form issue in another area.

In fact we might even want to do the same thing for the NAME Table. If one person may belong to many organizations or if we want to capture more than their home address we would have to disassemble the NAME table too.

Note also that in Figure 4 all the foreign keys have been labeled. They were not all identified in earlier figures to avoid some confusion. But also note that since data relationships work in 2 directions the foreign key can be on either side of the relationship as diagramed. For example I have now broken out orders into ORDERS, ITEMS and ITEM DESCRIPTION (a far more functional approach) and used the Key/Foreign Key approach. Notice that many orders can be connected to a person but an order can only be connected to one person. So the FK is on the ORDERS side of the 1:M relationship. But also notice that the Item can

only have one description (1:1) but the description can be connected to many records in the ITEMS table. So the FK is in the ITEMS Table even though the relationship is diagramed as 1:1 (although it is in fact M:1) from ITEM to ITEM DESCRIPTION since the value can repeat in different records and if we were to reverse the relationship to display all line items with that description the relationship would now be 1:M ITEM DESCRIPTION to ITEM. (Remember, a full 1:1 relationship links the actual Key in both sides of the relationship. In diagrams, however, a M:1 is often represented, or interpreted a 1:1 functionally although not in fact.)

The fifth rule of normalization is to **isolate semantically related multiple relationships.** There may be practical constraints on information which justify separating logically related many to many relationships (fifth normal form). In practice this is done for efficiency of entry and updates more than for representing the logic of the data. It tends to grow naturally out of the normalization process.

Suppose, for example, we wanted to capture not only every phone number where we can reach someone but also the address to which that phone number is related since most phone numbers are address

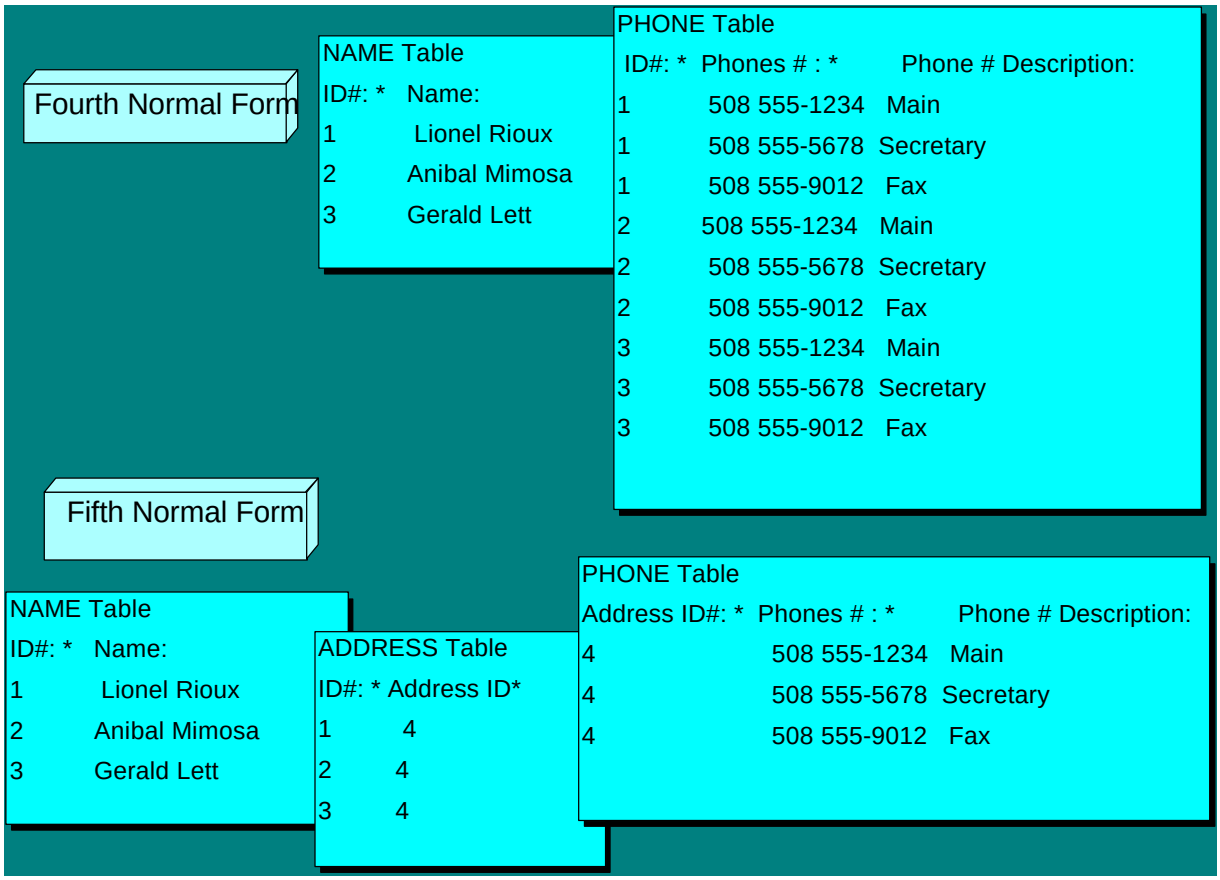

Figure 5: Comparison of Fourth and Fifth Normal Form

specific unless they are mobile. This suggests a person to phone to organization address relationship which we have not captured. Suppose we have one organization address with 3 people who all have the same 3 phone numbers (Main, Fax, Secretary). To capture this relationship in forth normal form we use Address ID, Name ID Phone Number. This creates 9 separate entries to capture the

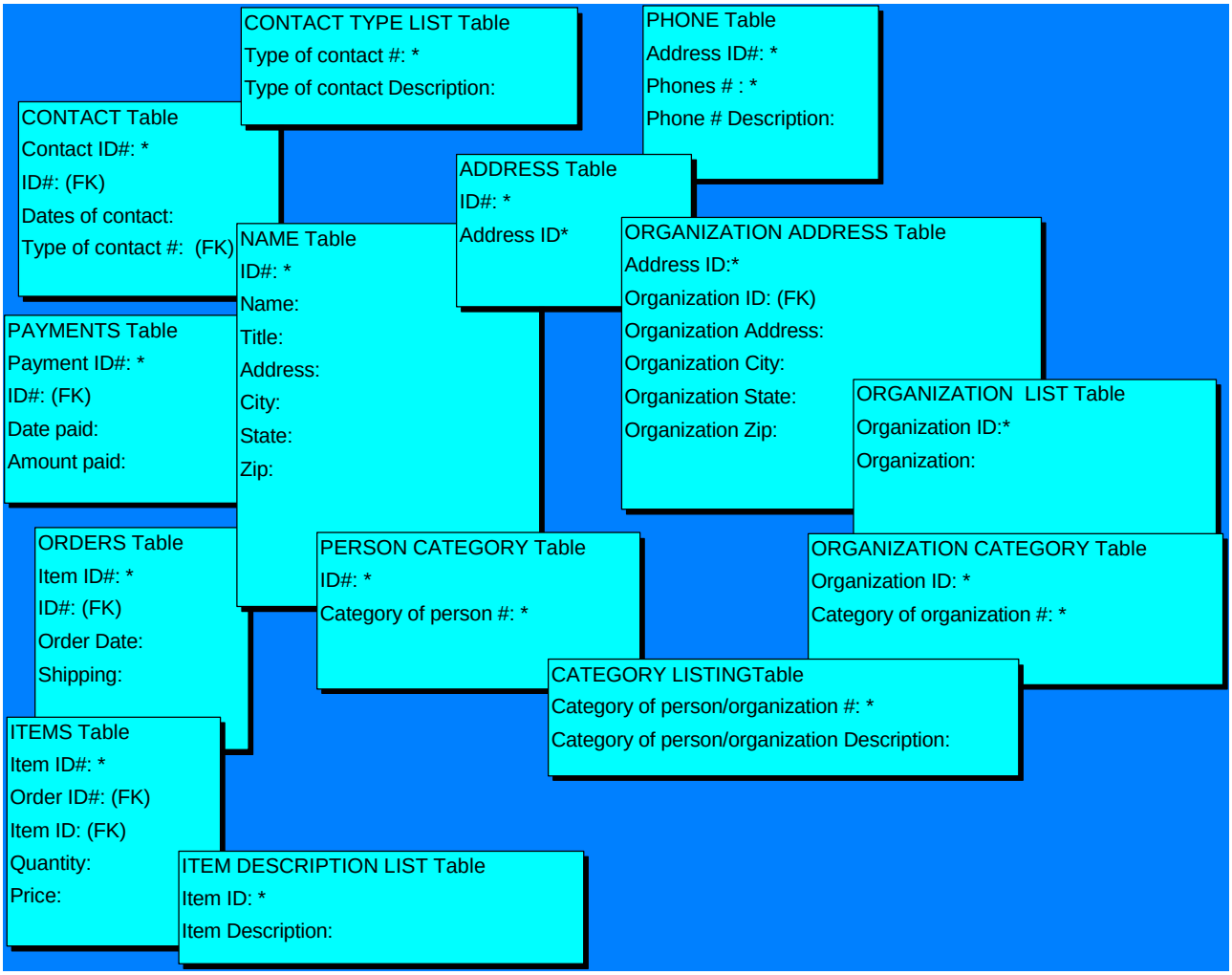

Figure 6 Fifth Normal Form

relationships and works fine. To be more efficient, however, we could create 2 tables. One would contain the Address ID and the Name ID (Currently in NAME) and the second would contain the Address Id and the Phone#. This captures the same relationship in 6 entries (see Figure 5).

In Practice this would change our organization address relationship as well (see figure 6).

Conclusion

An important thing to remember about normalization is that it is not an end in itself. The primary purpose of normalization is to

model your real world situation in the computer database while eliminating the potential for introducing anomalous data and error. Few databases are fully normalized out to fifth normal form nor do most need to be. Once you have normalized sufficiently to eliminate potential data entry or relationship anomalies you are most likely sufficiently normalized.

Also you should realize that this may seem to increase the complexity of your development and make it somewhat more complicated to display your data and report it. Beginning developers often confuse data capture and storage issues with data display and reporting issues. Temporary denormalization is often used for those latter purposes since display and reporting are not related directly to data entry and storage. Denormalization for those purposes does not create error in the actual stored data as you create temporary tables which are denormalized but have no effect on the actual tables used for data entry and storage.

Thanks to reviewers Elmar von Muralt and Larry DiGiovanni for their valuable contributions.