Data Modeling Using Entity Relationship Diagrams: A Step-Wise Method

Michael A. Chilton
Department of Management
College Of Business
Kansas State University
Manhattan, KS 66506
mchilton@ksu.edu

ABSTRACT

The concept of designing a relational database can be a difficult one for beginning students to assimilate. Evidence of poor database design in the past ten to fifteen years seems to suggest that educators might need to take a closer look at the way these concepts are presented in the hopes that student understanding might improve and hence, database design might also improve. A method for teaching these concepts that emphasizes a "back to basics" approach is presented to directly address this problem. The method makes use of a simple framework for helping students learn database design that can be used to supplement any popular text book. The framework is broad so that general information about the design can be obtained, but expandable so that increasing amounts of detail can be added as the design progresses from the conceptual stage through the logical stage, without losing sight of the final goal. The method by which the steps of this process are accomplished within the framework is explained in detail and it is shown how to develop an entity-relationship diagram (ERD) from the information obtained from the users. Although the E-R model is considered dated by many educators, the proper way to apply the method is provided and arguments in favor of its continued use are presented. Examples are included to illustrate the salient points including some which point out common errors and how to address them.

Keywords: Data modeling, Entity relationship diagram (ERD)

1. INTRODUCTION

A data model is a particular way of expressing facts and how they are related to one another (Connolly and Begg, 2002). Data modeling is the formulation of the data structure such that the meaning of the data is clear and it is easily communicated and shared with those who need to have access to it. Introductory database students often have difficulty in understanding the modeling process and as a result, often develop designs that are either inconsistent with the users' needs or satisfy only the users' current needs. In the latter case problems may occur in the future when the users' needs change. When this happens, the database must either be completely re-designed or what is more likely, any problems with inconsistency must be solved programmatically. Utilizing the front end software to overcome a poor design in the back end suggests that the programmer is assuming duties normally performed by the DBMS itself. This represents a tremendous waste of effort and increases the likelihood of errors.

Poor database designs often result from the inability to achieve data independence in the data model. Data independence prevents or reduces the problem of modification anomalies and allows new and updated applications to be written without having to change the structure of the database. One potential reason for this difficulty is the departure from, or the lack of use of, a simplified method for identifying, grouping and relating the relevant facts that users need. Such a method needs to emphasize finding the answers to four basic questions:

1) what facts should we store;
2) how should we store them;
3) where should we store them; and
4) how are the facts related?

The purpose of this paper is to present a simple, step-wise method for data modeling whose output, an entity relationship model, is consistent with the users' perceptions of their own needs for data, is generally in 3rd normal form, complies with the two cardinal rules of database design (entity integrity and referential integrity), and which can be used to assist the programmer in the formulation of simple data manipulation queries. The method utilizes a framework that tabulates the steps to be performed in each phase of the design process and allows the student to visualize the progress of the design. The student can remain focused on the overall goal of database design, and insert additional detail as the design progresses so that the final product correctly mirrors the users' perceptions of the data storage requirements. Examples are presented to clarify the methodology and to illustrate both the difficulties that are commonly encountered by students and how to deal with them.

The paper opens by providing a background of and discussing the need for a simplified framework for developing data models that can be used along with most of the database textbooks currently in use. The framework itself is then presented and its salient features are addressed. Next is included a discussion of the result—an ERD that should satisfy 3rd normal form—along with potential errors that students might make if they unduly influenced by business processes and fail to correctly apply the framework. Because this subject is controversial and the E-R modeling approach is considered dated by many experienced academics and practitioners, a comparison is made with class diagrams, a tool used in techniques that follow the unified modeling language (UML) standard. Arguments are presented in favor of the continued use of the E-R approach along with a discussion of the proper use of class diagrams.

1.1 Background

Recently published evidence shows that database design has been average to poor in the last decade, when in fact relational database design, which has now been with us for long enough to work its way into the lion's share of systems in use today, should be flourishing. Blaha (2004) published the results of eleven years of analyzing database quality and the surprising result was that overall, database quality is mediocre at best. Academics agree with Blaha's findings. In the preface of his 9th edition, Kroenke (2004) reflects that he has seen too many databases that are poorly designed. Although Blaha (2004) emphasizes reverse engineering in his approach to quality checking, this is an after-the-fact check that does not address the problems that are introduced early on in the design cycle. In his own words, he believes that "universities are not teaching students how to model software. Many universities teach the syntax of modeling, but they don't teach the art and thought processes" (Blaha, 2004, p.24). One interpretation of Blaha's message might be that educators need to renew their emphasis on the fundamental building blocks used in data modeling in favor of newer techniques that unnecessarily complicate the process by including front end interactions in the back end data model. Kroenke (2004) decided to renew his emphasis on the ER model and downplay the semantic object model for the same reasons. This paper introduces a framework for data modeling based on the ER model first developed by Chen (1976) that emphasizes the thought processes involved in obtaining a valid data model the first time.

Research performed on database teaching methods has been sparse, especially concerning introductory courses aimed at students who will become database designers and/or administrators, and with little or no exposure to the subject. Ahrens and Sankar (1993) evaluated the use of automated tutors designed to teach introductory level database concepts to end users. Their empirical research focused on those end users with no previous knowledge in database design. They found that tutors were "moderately effective in closing the gap between skills required and skills learned by end users" (Ahrens and Sankar, p. 429). They also found that the tutors were less effective in teaching data analysis concepts and decision rule discrimination. This result suggests that another method for learning these skills might be appropriate, especially for students whose ambition is to design and administrate databases as an occupation. Another study that looked into using automated systems to teach database concepts was conducted by McIntyre, Pu and Wolff (1995). They investigated the use of expert systems as tools in database design. Their study was focused on advanced graduate students who had previously learned

Author(s)	Orientation	Data Modeling Approach
Connolly & Begg (2002a, 2002b)	Business/CS	Focus is on fact finding and user views; creating an optimal data model
Date (2000)	CS	Semantic modeling
Elmasri & Navathe (2004)	CS	No method prescribed
Lewis, Bernstein & Kifer (2002)	CS	No method prescribed
Kroenke (2003, 2004)	Business	Determine requirements, specify entities, specify relationships, determine identifiers, specify attributes, specify domains, validate model
Mannino	Business	(Entities are given), Identify primary keys, add relationships, refine the initial design
McFadden, Hoffer & Prescott (1999)	Business	No method prescribed
Post (2002)	Business	No method prescribed
Riccardi (2001)	Business	No method prescribed
Rob & Coronel (2002)	Business	No method prescribed
Watson (2004)	Business	Underline nouns to identify entities, find descriptors of entities (attributes), determine relationships

Table 1: Data Modeling Approach in Current Texts

relational database design. The focus of this paper is on introductory database students, and so a different approach is used.

There are quite a few excellent textbooks that can be used for teaching an introductory course in database design, all of which have been written by experienced authors and academics, yet one shortcoming that seems to be persistent is the lack of simplicity and clarity in explaining how to take the information received from the users and transform it into a data model. Table 1 shows several current textbooks reviewed in this analysis and the method presented in each for data modeling.

Although the table indicates that some textbooks have no prescribed method, these texts do have good discussions of what entities, attributes and relationships are and how they interact. In many cases, an example is provided with a given set of entity types and a discussion regarding their attributes. Discussion is also provided for enhancing and improving the design in a second iteration by taking into account such things as generalization/specialization and adhering to certain database rules, such as entity and referential integrity. In many cases these textbooks provide a thorough discussion of how to obtain documents that contain data and how to interview users to obtain their perspectives on information collected by the organization, but they do not specify a clear method that helps the data modeler move the extracted data from its raw form into a finalized database design. In short, they do not provide a step-wise method that links the thought processes involved in discovering the facts that the organization and its users need to store with the resulting data model.

The E-R model was developed by Chen to take advantage of the strengths of the network model, the relational model and the entity set model by achieving data independence and capturing the important semantic information found in the real world and doing so in a way that produces a more natural view (Chen, 1976). The E-R diagram is less confusing and easier to follow for introductory database students because it provides a pictorial representation of the data structure. In addition, it allows a programmer to quickly formulate queries by visually mapping a query to the data. The E-R approach has been cited as the premier model for conceptual database design (Teorey, Yang & Fry, 1986), but has fallen out of favor over the years as newer approaches are developed. The framework introduced here makes an excellent supplement to current textbooks because it helps the student to remain focused on the final goal, while adding increasing amounts of detail as he or she proceeds through the modeling process. It emphasizes the processes involved in data modeling and not just the syntax of data definition, as advocated by Blaha (2004).

The production of an E-R diagram does not guarantee a good database design, however, because the student must know how to form entities that correctly mirror the data that each user is exposed to and how to relate these entities. This process can break down if the student allows the processes that are inherent in any system to affect the database design. These processes are used by the system analyst in his design. They are reflected in data flow diagrams and show the data in motion throughout a system. In database design, we are concerned with the data at rest. The questions to be answered are: 1) have we stored the data correctly to prevent unwanted redundancies (i.e., one fact in one place), and 2) have we stored the data so that the systems analyst and programmers can access it and utilize it as either input or output to the processes that the users execute? To do so, we simply need to understand that we are interested only in what facts the users need to capture, retain, manipulate and view and how these facts are related, not what they are going to do with them (Date, 2000).

Here is an example to help illustrate this point. It involves a database for a company that sells, installs and services air conditioning, ventilation and heating systems. Let's assume that in order to perform certain types of maintenance work on these systems, workers are required to be certified on that system. How can a manager be assured that an automated system will correctly schedule only certified workers to be sent on service calls such that their certifications match the equipment to be worked on? The answer to this question from a database design point of view is that it doesn't matter. This is an issue for the programmers and systems analysts to work out, because it represents a process within the system and therefore reflects the data in motion, not the data at rest. If the database designers capture the necessary data to track employee certifications, certification requirements and service calls, the alignment of workers, certifications and jobs is easily captured by joining the tables in a query. However, if the database designers become overly concerned with this requirement, it might compel them to include an additional entity in the design to match certifications with the scheduling of service calls. This entity would be unnecessary and may introduce some modification anomalies that the programmers will have to deal with. To design a proper data structure the database designers must ensure that all the appropriate data is captured and captured in the correct format and in the correct place and must not allow system processes to unduly influence their design.

In order to accomplish this, introductory database students need a process to produce an effective ERD in a step-wise fashion which they can refer back to as they proceed through a database design. This process needs to emphasize data structures and de-emphasize system processes to prevent an unnecessary influence on the database design. It should also be expandable so that the additional details of database design (e.g., determining primary keys) can fit neatly into the framework. The approach provided here presents such a framework that has been successful and helps the student to address the important details in the design without losing sight of the overall goal of producing an accurate data model.

2. THE FRAMEWORK

2.1 The Process

Building an effective ERD begins as a simple three step

process. It consists of: 1) determining the data requirements; 2) grouping the data together to form entities; and 3) relating the entities. The database designer must iterate through these three steps until all of the data is accounted for. Steps 1 and 2 can easily be reversed if there are entities that are fairly obvious in the organization for which the database is being designed. However, if the database designer is not familiar with the business entities or there exists some confusion in the way entities and the facts which describe them can be combined, then the steps should be performed in the order listed. Regardless of the order in which steps 1 and 2 are performed, there should almost always be some left over facts—data that doesn't quite match well with other entities, yet these facts are still important and must be retained. These additional "orphan" facts can be set aside in the first iteration of steps 1 and 2 and re-visited in step 3. Some of these facts may be the result of relationships between entities and should therefore be handled somewhat differently than others. An additional fourth step, verifying that the design reflects the users' perceptions of the data requirements, may be necessary, but it should come after several iterations of the first three steps. This process is so simple that students can lose sight of it as they become involved in it, and so they must be re-focused occasionally to get back on track.

The steps just presented provide the broader framework that the data modeler uses in the conceptual design stage. As the design moves into the logical phase, additional important steps that provide greater detail are performed. Table 2 provides the modeling framework with the steps listed in order from top to bottom and left to right. It includes a fourth step, verifying the design, but this step should be performed just prior to the physical design phase. It is divided into the three design phases with the steps appropriate to each stage listed in their respective columns.

The data modeler should proceed from the upper left corner in the conceptual design phase with step 1, determining the data requirements, complete all steps listed in that column in sequential order from top to bottom. The bottom of this first column represents the completion of the logical design phase and the result from this phase should be the ERD (Mannino, 2004). Once satisfied that all facts have been collected, the designer should move to the top of column 2 and begin the logical design phase. The tasks listed in this column should also be completed in sequential order from top to bottom. While working in this phase of the project, the designer may discover some facts that have not yet been included and must therefore refine the ERD that resulted from the conceptual phase. In addition, if the project was divided and a series of smaller ERDs were produced from the users' views of the data, now is the time to integrate these views into a single, overall ERD. The final column has listed a few of the steps that should be included in an introductory database course, but it is not complete. For continuity, the steps listed in the final column relate only to those steps performed in the previous two columns. There are of course, additional steps that must be performed during the physical design phase, but these are beyond the scope of this paper.

2.2 An Example

"*Few things are harder to put up with than the annoyance of a good example.*" (Twain, 1903, p. 164). We now use a simple but non-trivial example to demonstrate the use of this method, how students can get side tracked while in the midst of a complex design, and what to do about it. The example concerns itself with parts as they flow through a machine shop. It is simple because it focuses only on one section of a business that is dedicated to supplying parts to either internal or external customers. Much of the rest of the business is eliminated from this example in order to simplify it and highlight the pedagogical goals of the framework introduced herein. This example is however, non-trivial because it involves a scheduling operation, which can be difficult for beginning database students to visualize and which contains processes that the system analyst must use to develop a computerized tracking and scheduling system. These processes only get in the way of our database designers and can cause them to make some bad decisions about the database design. The goal of showing this example is to 1) work through the framework to demonstrate its effectiveness, 2) highlight some areas in which students might get sidetracked, show the decisions they might make as a result and demonstrate how to get them back on track with the framework, and 3) show how the processes used within this shop might exert a negative influence that would further skew the database result. Let us first provide a narrative and a few business rules and assumptions to set the stage.

The shop receives raw stocks that it must machine into parts for use by either internal or external customers. Within the shop there are several machines that are operated by qualified machine operators and each machine can perform several operations such as milling, shaping, drilling, cutting and so forth. As an example a large piece of aluminum stock might be milled to a certain thickness, cut into several pieces, shaped to form the skin of an aircraft (e.g. a wing root), and pre-drilled with holes for fasteners during assembly to an aircraft frame. The finished parts then exit the system where they are packaged and delivered to customers. In order to accomplish this on a daily basis, operators are assigned using a schedule that details a) what must be done, b) by whom it must be done, and c) which machine must be used. Joe, a machinist qualified on several machines is given a schedule of tasks for a day in August as shown in table 3.

Several assumptions are necessary in the development of an adequate data model for this example. First and foremost, we need to assume that the schedule is produced by a management application that interfaces with the database. Such an application would control and optimize the flow of work through the machine shop and ensure that machines are not assigned to more than one operator/operation at once. We can also assume that the start and end times listed on each machinist's daily schedule include enough time to acquire new stock, set up the machine for the intended operation(s) and reset it to its original condition (if necessary) at the end of a job. Finally, we can assume that

Step	Conceptual Design Phase	Logical Design Phase	Physical Design Phase
1	<----------Determine the data requirements---------->		
1.a	Identify the facts required by users		Decide on DBMS
1.a.1		Integrate views	Decide file structures
1.a.2		Determine data types (formats and length)	Data dictionary entries
1.a.3		Specify domains	Data masking
2	<----------Determine the entities---------->		
2.a	Group facts based on commonalities		
2.a.1		Identify unique identifiers (primary keys)	Identify additional indexes
2.a.2		Identify strong entities	
2.a.3		Identify weak entities	
2.b	Set aside any left over facts		
3	<----------Determine relationships between entities---------->		
3.a	Find all direct relationships		
3.b		Determine cardinality	
3.b.1		Resolve many-to-many relationships	
3.b.2		Identify foreign keys	
3.c	Re-visit any left over facts		
3.c.1		Can they be combined to form a new entity?	
3.c.2		Do they result from the relationship itself?	Create DDL scripts
4	<----------Verify the design---------->		
4.a		Does the design mirror (the user's) reality?	
4.b		Normalization checks	

Table 2: The Data Modeling Framework

templates and other specifications for each operation are in some way given to the machinist and are not a part of the database design (as yet).

Each operator is given a schedule similar to the one shown in table 3. In Joe's schedule we see that he is scheduled to move from one machine to another throughout the day. At 11:20, he interrupts his work, takes a lunch break, and then continues with another job immediately after lunch.

We now apply our 3-step method to this report. First we look for facts on the report that need to be collected and stored. This report makes it quite easy because they are all labeled with column headings. We then group together those facts that share something in common and form entities. We end up with a list that might look like table 4.

Arranging the facts based on commonality produces 5 separate groups: the schedule itself, the operator, the machine, the operation and the part. Forming entities from these groups produces the result shown in figure 1. We have added the attributes to the entities for clarification.

Schedule date	Operation to be performed
Operator name	Machine used
Start time of each operation	Part(s) needed
End time of each operation	Quantity of parts needed

Table 4: Facts from the Schedule Report

From other sources of information in the machine shop, we should discover additional facts that need to be collected about each entity. Once we have these facts, we can modify the diagram in figure 1 to include these facts and we can

Daily Schedule for Aug 6, 2003						
	Times					
Operator	Start	End	Operation	Machine	Part	Qty
Joe	8:00	10:16	Milling	M1	P1	4
					P9	1
					P76	3
	10:16	11:20	Drilling	M2	P2	4
					P10	1
					P77	3
	11:20	11:50	<=========Lunch break========>			
	11:50	12:45	Sanding	M2	P3	4
					P11	1
					P78	3
	12:45	1:40	Buffing	M3	P4	4
					P12	1
					P79	3
	1:40	3:56	Milling	M1	P1	6
					P9	2
					P76	4
	3:56	5:00	Drilling	M2	P2	4
					P10	1
					P77	3
<==========End-of-Day==========						

Table 3: Typical Daily Work Schedule

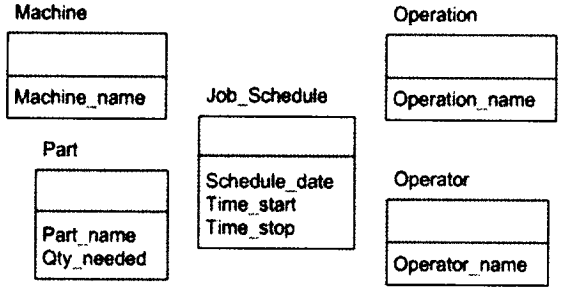

Figure 1: Entities & Attributes, Steps 1 & 2

create some attributes as necessary to serve as primary keys. The diagram should now look like figure 2[1]. Note that the fact, "Qty_needed" has been set aside for now and that a different attribute in the Part entity, "Qty_on_hand" has replaced it. Students may have difficulty understanding why this must be so the first time they see this example, but it should become clear to them when they are led to the third step, relating the entities. An additional fact that the student may discover is that each operator must be qualified on each machine for each operation to be performed. This might present additional difficulty to the student, and can also be treated as left over facts that must be dealt with in the next step. Finally, it should become clear that each operation can include a specification of some sort but that each part may include an additional specification for a particular operation. We can call this a part-specific specification and place it with our left over facts as well.

Our next step is to relate the entities together. Our database design steps are proceeding smoothly, but this is now the point at which a student might allow processes to interfere with the design. For example, a potentially confusing aspect of the facts thus far collected is that the quantity of parts needed is a subset of the work schedule and not of the part entity. This is where the 3-step method becomes so useful, because it compels us to group like facts together and form entities. Looking at the fact, "qty_needed," the student must ask himself if this is a fact that truly describes a part that we have on hand to be machined or does it describe something else[2]. Clearly the answer is that it describes the work schedule because it relates directly to the amount of work that each operator must perform and has nothing to do with describing a part or an operation on the part.[3] The student has just discovered that the commonality shared by facts are

[1] There may be more facts required for each entity than is shown. We add only enough for this example to serve an illustrative purpose.

[2] During class discussion regarding this example, most students wanted to place this attribute in the Part table.

[3] We assume that each operation is performed on only one part at a time for the sake of simplicity in this example. If this is not the case, then the fact, "qty_needed," would describe an operation as well as the work schedule.

not always obvious and may require some thought, but as long as the 3-step process is strictly adhered to, this discovery can easily be made.

We now draw the relationships in the diagram to connect the entities. Each of the four strong entities—machine, part, operation and operator—are directly related to the job_schedule entity, and so we connect these tables with job_schedule. As we look at our left over facts, we can see that qty_needed should be included in the job_schedule entity and that machine_quals and part_specific_specifications result from the relationships between machine and operator and between part and operation respectively.

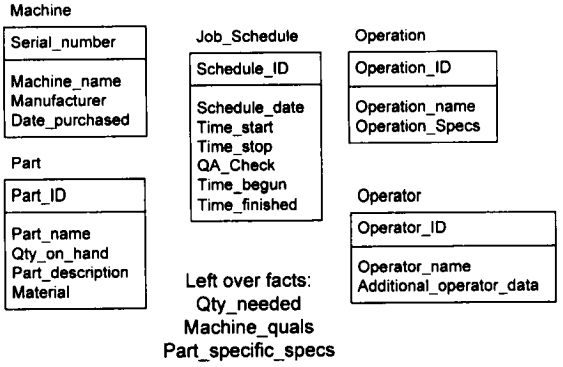

Figure 2: Enhanced Entities

We should therefore connect these tables and include these facts as relationship attributes. This is shown in figure 3.

Applying the rules for resolving many-to-many relationships, we add two intersection tables to our diagram and include the relationship attributes within these tables. An additional relationship is extended from the Operation entity to the Machine_Quals entity because operators are qualified on operations and machines.

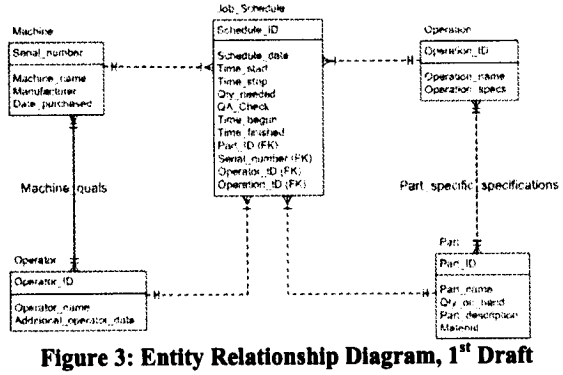

Figure 3: Entity Relationship Diagram, 1st Draft

Otherwise, our final ERD takes a shape very similar to the first draft depicted in figure 3. Although there may be more than one solution to the machine shop, a working solution is presented in figure 4. To the facts listed above we have added a few others that were discovered through additional means such as other input forms, interviews with users, operators and managers and the like. In this diagram strong entities are depicted as rectangles, weak entities have rounded corners, and identification dependencies are shown with solid lines. Crow's foot notation is used to document minimum and maximum cardinalities. A visual check of the ERD reveals that there are no violations of 3rd normal form (and consequently, by definition, no violations of 2nd or 1st normal form as well).

Formal normalization checks can be performed once the designer is satisfied that all the data requirements have been met. As shown in table 2, normalization is included as part of the logical design phase, verifying the design. However, the beauty of this modeling technique is that if the steps are performed with care, the model should already adhere to 3rd normal rules, and any checks for functional dependencies should indeed simply verify that this is true.

The utility of this method of data modeling is further illustrated by looking at a potential mistake (and its cause) that students can make during the design of this example database. In an analogous manner to the air conditioning example provided earlier, students may see a need to track the scheduled machinists so that only those who are qualified on a particular operation and machine are scheduled to do so. This is a process within the business that must be controlled programmatically and not by the database directly. If the students allow this process to influence their database design, then they might create additional tables to maintain this information and not utilize a simpler design as depicted in figure 4. This will bring about modification anomalies that will require the programmers to ensure that they have updated information in all the tables that it appears; clearly a labor intensive effort that will need to be reflected in all applications written that interface with this database.

Earlier we assumed that the schedule is produced by an application that interfaced with the database and not the database itself. Although many DBMSs allow the development of stored procedures in which a scheduling application could be developed, such a technique is normally beyond the scope of an introductory class. Additionally, a scheduling application might include linear programming algorithms in order to optimize workflow that the database is not capable of. If the students do not recognize this fact and attempt to include a scheduling component in their design, they may produce additional tables that contain data which is already stored in other tables or they may create tables that incorrectly categorize the data. Students may also attempt to include attributes that prevent machines from being scheduled to perform multiple jobs at one time. While this may be a necessary constraint on the physical work flow, it need not be included in the database design.

The machine shop example was used as an assignment in a class of 20 introductory students. Although the students had been taught the method discussed here, no mention was made of separating the processes from data storage requirements, but the narrative included the necessity of scheduling only qualified machinists on certain machines (a process).

The result was that over half the students attempted to define an entity that tracked the scheduled activities, but failed to properly relate this table to the others in their design.

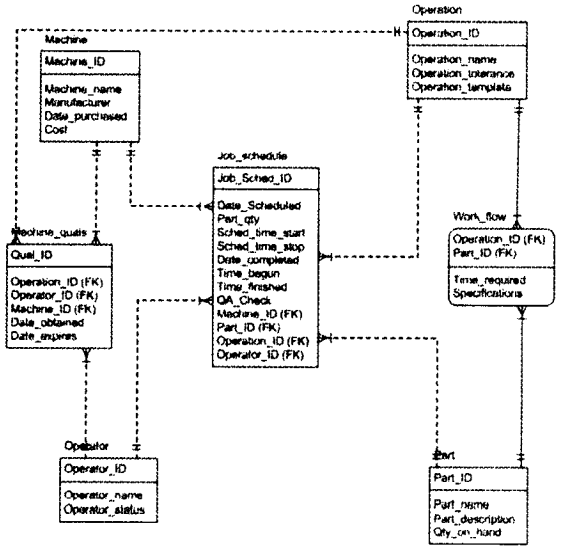

Figure 4: Completed ERD for the Machine Shop

Most of the students were able to correctly identify the facts that needed to be stored, but were unable to place them properly within the set of relations they had included in their data model. None of these designs would have allowed a query to extract a schedule that prevented machinists from being assigned to tasks for which they were not qualified. This is a classic example of allowing the processes to obscure the correct data storage requirements and properly relate the resulting tables. The question they needed to ask was, "Do the facts regarding the schedule relate directly to the worker, the machines being used, the parts used or to the operations being performed, or do they relate solely to the relationship between the worker, the machines, the parts and the operations?" Students who utilize the 3-step method and ask themselves how the facts are related to one another quickly realize that the latter is true. They soon become aware that the processes serve only as a hindrance in their data modeling process.

Figures 5 and 6 show two examples of student-produced ERDs that attempt to model the machine shop. The first example shows how the student can allow the processes to adversely influence the data model design. Here, the certifications of each machinist are maintained in the work_schedule table in an attempt to ensure that only certified machinists are allowed to perform the required tasks. This requires that the certifications be entered for each operation that is scheduled and does not follow the "one fact in one place" rule. As such, each time the certifications change, they must be changed in every tuple in which they are recorded. The second example demonstrates what might happen when a student fails to correctly categorize the necessary facts. Here the certifications are stored with the machinist in the Operator table. Since we don't know exactly how many certifications each machinist will have, we will have to over-estimate to ensure that there are enough fields to capture them all. If we assume some number, say 10, and the average for all machinists is 6, then we have an average of 40% of our certification attributes with null values.

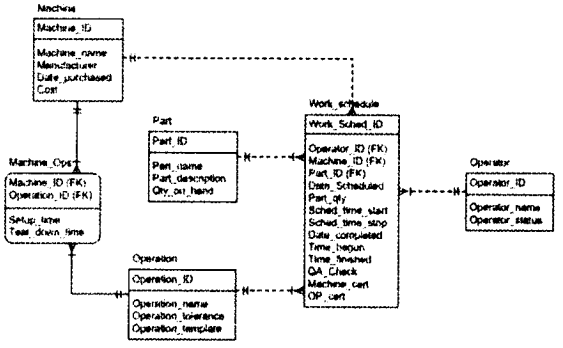

Figure 5: Certifications Stored with the Schedule

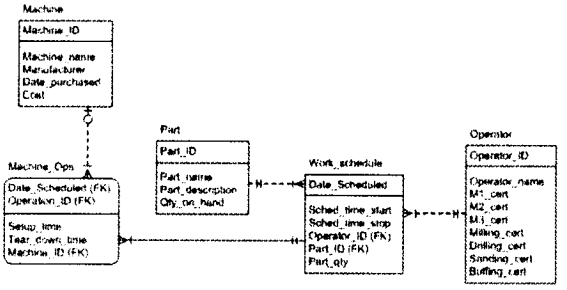

Figure 6: Certifications Stored with the Operator

2.3 Class Diagrams

The UML is a standard adopted by the Object Management Group for object oriented applications that provides several diagramming tools for systems analysts. The class diagram is one of these tools and it closely parallels the ERD because it can include data objects and show their relationships. It is primarily used for application development (Kroenke, 2004) and has not seen much use in database development as yet. Although it offers promise and extensibility, it also introduces several problems that may serve to confuse beginning database students. Methods for each class are depicted directly on the diagram. These methods are the operations that the class is supposed to perform when asked to do so by other classes or by themselves in response to internal requests. Thus, instead of separating business processes from the data stores as is advocated in the 3-step method, the UML integrates them within the class diagram. It also does not identify the primary key for each instantiation of a class on the diagram. These two characteristics can make it more difficult for the beginning student (and those new to class diagrams) to properly visualize, design and implement the data structure.

Shoval and Shiran (1997) compared the ER technique to object-oriented data modeling and found that the ER technique results in higher quality, less time to design and is preferred by most designers. Class diagrams should be used primarily by systems analysts for the front end development

because their strength is that they clearly identify how the data structure blends in with the front end applications. The data modeler should remember that the data structure is not cleanly isolated in a class diagram as it is in an ERD. This latter fact is of paramount importance for beginning students who are just beginning to discern the data structure apart from the business processes in the system. Siau and Cao (2001) investigated the complexity of the UML and found that the "UML can be daunting to the novice users" (p. 32). For experienced data modelers who also have experience in object-oriented techniques, it may provide a more integrated view of the system that might help in refining and tuning a database that is already in production.

2.4 The Framework in Action

When the instructor observes the students using the method, its value becomes apparent. Recently a class was assigned a project in which they were to produce a data model for a publication that reproduced television listings. A newspaper, for example, runs the daily television schedule that includes information regarding times, channels and shows that are televised both locally and on cable channels. It sometimes also includes a short synopsis of the show and lists the cast of characters for both regularly broadcast and special occasion shows. News shows, sporting events, movies and public broadcast shows are all included in the mix. The designs of two of the student groups are included in this discussion to amplify what can happen when the technique is not used or applied incorrectly. Each group was assigned the project as a semester long class project in which the first steps were to design the data model based on information found in a local newspaper. The groups were to look at a newspaper listing of daily television shows and attempt to design a data model that could be easily transformed into a database and respond to queries. A set of business questions was given to the groups after they had settled on a data model. By converting these questions into SQL queries, running them against a populated database and checking the results, the groups could test the effectiveness of their design. The first group became pre-occupied with the TV listing in the newspaper and allowed it to skew their design. The newspaper provided a tabular list that included the day, the time, the channel and the show scheduled for that time slot. Because the team became overly concerned with the format of the output, they focused more on the results than on the entities themselves. It seemed as if they had abandoned the framework in favor of a "shotgun" approach and ended up with a few massive tables that contained a mixture of redundant and null data. The team quickly discovered that they could not construct the queries in SQL from the business questions that became a part of the project. These questions were simple ones (e.g., produce a listing of all sit-coms on Saturday night between 6 and 11 pm), yet the team struggled with the problem of matching the shows to the times they were broadcast to the channels on which they were broadcast. After reviewing their design and re-visiting the 3-step process of the framework presented here, the team correctly categorized the facts that they needed to store and began to assemble a better set of related entities. Once their design improved, they discovered that the business questions were quite easily formed into SQL queries through simple table joins.

The second group worked independently on the same project. This group noticed that the TV shows were different with respect to the types of facts that needed to be collected about each, but failed to relate them together correctly. Their resulting design included an entity for each type of show and all of these entities were related to a single entity in which the schedule was recorded. This table had a foreign key for each of the other tables, but since only one table could be referenced at a time, it contained a large number of null values. In order to produce the answer to a query that requests a list of all shows scheduled for a particular date and time, their SQL included several `UNION` operators, one for each pair of entities. This team failed to properly apply the process and the result was a design that had to be programmatically "adjusted" to obtain the correct output. After establishing a super-class for the broadcast shows, the team was able to perform the same query using much simpler syntax.

In both of these cases, the first result was a poor data model. The first team was unable to formulate SQL queries against their data model at all. They became sidetracked because they allowed the process of producing an output to interfere with their design. The second team managed to formulate their queries only after performing some research into more advanced SQL techniques, which they later found were not necessary. This team did not notice the common features in the data and failed to correctly categorize them.

4. SUMMARY

This paper has presented a framework for logical database design that can be used in all of the stages of the design and has shown to be successful in a teaching environment geared toward introductory database students. It helps students clarify the process of the conceptual and logical design phases and provides them with a step by step procedure to follow as they uncover the facts which must be managed (i.e., stored, retrieved, modified and related) in an organization. This method is simple yet can be used on complex systems, it can complement any database textbook in use, and it is expandable so that additional detail can be added as the design progresses without losing sight of the overall goals of the design. The end result of the application of the method is an E-R diagram, the fundamental building block of database design and visual tool that can be used in formulating queries. Using this particular method, the resulting design should generally satisfy 3rd normal form upon completion of the conceptual design stage, although a formal test of normalization is performed later during the logical design stage. This diagram has significant utility because it can help the database administrator visualize the structure of the data, it can allow the programmer to quickly develop a script in SQL that will create all entities either manually or through the use of software, and it can provide an easy way to formulate queries from business questions.

Database educators should find the method useful for teaching introductory database classes, because it emphasizes the basic concepts of database design and helps students avoid some of the pitfalls that might adversely affect their design. Two of these pitfalls, improper categorization of data and allowing business processes to skew the design, were discussed with actual examples provided. Newer methods, such as class diagrams that accompany UML notation, include the business processes directly in the diagram as methods, and may have an undue negative influence on the final design. This is not to say that the UML cannot be used, but it should be reserved for more advanced students, after they have had practice in separating the data at rest, which should be stored in the database, from the data in motion through the organization, which is captured in the business processes. Because introductory students often have difficulty seeing this, it is important that they first be exposed to techniques which help them achieve a state of data independence in their design to help prevent and minimize modification anomalies. The evidence of poor database design in the last 10 to 15 years (cf. Blaha, 2004; Kroenke, 2004) overwhelmingly supports this conclusion.

Practitioners should find the method useful because it can help them clarify the steps leading to a logical database design. Provided the method is followed with care and subjected to an iterative process, all data should be accounted for and correctly captured in the design. The result will be a better and more efficient conceptual database design.

The real test of this method can be seen in the semester projects of students who are required to design a database for an organization, develop the ERD that results from their design, and create the tables and populate them with sample data. This has been a resounding success in my own classes where I have seen improvement in the quality of the students' designs as I began to incorporate the method introduced here. The success can be seen in improved ERDs, in parsimonious SQL queries and in the time savings that accrues from not having to correct data modeling mistakes programmatically. The approach used here, which emphasizes the basic concepts of the relational database technique, should provide productivity gains in future database designers.

5. REFERENCES

Ahrens, J. D. and Sankar, C. S. (1993). "Tailoring Database Training for End Users," *MIS Quarterly*, (17:4), pp. 419-439.

Blaha, M. (2004). "A Copper Bullet for Software Quality Improvement," *IEEE Computer*, (37:2), pp. 21-25.

Chen, P. P. (1976). "The Entity-Relationship Model—Toward a Unified View of Data," *ACM Transactions on Database Systems*, (1:1), pp. 9-36.

Connolly, T. and Begg, C. (2002a). *Database Systems: A Practical Approach to Design, Implementation and Management, 3rd Ed.*, Addison-Wesley, Harlow, U.K.

Connolly, T. and Begg, C. (2002b). *Database Solutions: A step-by-step guide to building databases*, Addison-Wesley, Harlow, U.K.

Date, C. J. (2000). *An Introduction to Database Systems, 7th Ed.*, Addison-Wesley, Reading , MA.

Elmasri, R. and Navathe, S. B. (2004). *Fundamentals of Database Systems, 4th Ed.*, Addison-Wesley, Boston.

Kroenke, D. M. (2003). *Database Concepts*, Prentice-Hall, Upper Saddle River, N.J.

Kroenke, D. M. (2004). *Database Processing: Fundamentals, Design and Implementation, 9th Ed.*, Prentice-Hall, Upper Saddle River, N.J.

Lewis, P. M., Bernstein, A. and Kifer, M. (2002). *Database and Transaction Processing, An Application-oriented Approach*, Addison-Wesley, Boston.

Mannino, M. V. (2004). *Database Design, Application Development, & Administration, 2nd Ed.*, Irwin, McGraw-Hill, Boston.

McFadden, F. R., Hoffer, J. A. and Prescott, M. B. (1999), *Modern Database Management, 5th Ed.*, Addison-Wesley, Boston.

McIntyre, D.R., PU, H. and Wolff, F.G. (1995). The use of software tools in teaching relational database design, *Computers and Education*, (24), pp. 279-286.

Post, G. V. (2002). *Database Management Systems: Designing & Building Business Applications, 2nd Ed.*, McGraw-Hill, Boston.

Riccardi, G. (2001). *Principles of Database Systems with Internet and Java Applications*, Addison-Wesley, Boston.

Rob P. and Coronel, C. (2002). *Database Systems: Design, Implementation, & Management, 5th Ed.*, Course Technology, Boston.

Siau, K. and Cao, Q. (2001). "Unified Modeling Language (UML)—A Complexity Analysis," *Journal of Database Management*, (12:1), pp. 26-34.

Shoval, P. and Shiran, S. (1997). "Entity-relationship and object-oriented data modeling—an experimental comparison of design quality," *Data Knowledge and Engineering*, (21), pp. 297-315.

Teorey, T. J., Yang, D, and Fry, J. P. (1986). "A logical design methodology for relational databases using the extended Entity-Relationship model," *Computing Surveys* (18:2), pp. 197-222.

Watson, R. T. (2004). *Data Management: Databases and Organizations, 4th Ed.*, Wiley and Sons, New York.

AUTHOR BIOGRAPHY

Michael Chilton is an Assistant Professor in Management Information Systems at Kansas State University, where he teaches database design and telecommunications and networking. His teaching and research interests are in database modeling and design, management of IT personnel and computer communications and networking. He has published in the *Database for Advances in Information Systems* and other journals. He is a guest editor for an upcoming special issue of this journal which will foster debate between the ERD and the UML diagramming methods.

Information Systems & Computing
Academic Professionals

STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 1055-3096