Teaching Relational Database Concepts to Computer Literacy Students: The Spreadsheet Metaphor

Geoffrey Steinberg
gsteinbe@kent.edu
College of Business, Kent State University
Kent, OH 44120 USA

Abstract

A challenge facing computer literacy classes is the proper course content level. The challenge is to take the students to a higher level academically but without driving them away because they feel an "easy" course has become "hard." Accepting this challenge, we have moved the database portion of the computer literacy class beyond keystrokes (rote learning) to data modeling using the spreadsheet as a metaphor for a relational database. Although not found in the current literature, this metaphor facilitates the understanding of the foundations of relational theory and enables computer literacy students to design normalized multi-entity databases within several class sessions. Students combine relational theory and keystroke knowledge of Microsoft Access to implement finished applications. The resultant applications are free of data redundancy problems that often plague non-normalized databases.

Key Words: Database, Computer Literacy, Pedagogy, Teaching, Normalization, Data Modeling, Spreadsheet, End User

1. Introduction

A challenge facing computer literacy classes is the proper course content level. Courses containing material too elementary for the student are spurned as "blow-offs" resulting in declining enrollments (Tucci). Literacy classes with an in-depth exploration of computer science material may scare away all but the most mathematically inclined student. The challenge is to take students to a higher level academically without driving them away because they feel an "easy" course has become "hard." Accepting this challenge, we have moved the database portion of the computer literacy class beyond keystrokes (rote learning) to data modeling.

Effective data modeling requires an understanding of the foundations of relational database theory. Without these foundations, students typically develop multi-entity databases using a single table. The resultant applications suffer from data redundancy problems associated with non-normalized databases.

Database theory is given little emphasis in computer literacy classes (Hutchings). Textbooks universally present word processing before spreadsheets are introduced. Appropriately, the final topic is database, a more abstract application than word processing or spreadsheet. Unfortunately, textbooks typically continue the keystroke methodology of word processing and spreadsheet and forgo discussion of relational theory. Usually the most basic database concepts (such as table or attribute) are presented using metaphors and terminology from an Information Systems (IS) based perspective.

In a computer literacy class, the IS perspective is difficult because it fails to build on and is not associated with prior student knowledge. Assimilation theory holds that a familiar metaphor functions as an advance organizer and facilitates the understanding of new knowledge, that is,

concepts from relational database theory (Mayer). This article describes an approach to the introduction of fundamental relational concepts using a familiar metaphor, the spreadsheet. This metaphor is more consistent with the end-user's knowledge base that traditional IS oriented presentations, a factor critical to learning the subject material.

The metaphor of a spreadsheet as a relational database table can be inferred from both the "conventional" IS database definition and from relational theory. An IS-oriented definition of a database is a group of logically related files, files being a group of logically associated records and fields. A spreadsheet may be viewed as a sequential file of fixed length records. [1] The fixed record length characteristic is derived from the spreadsheet requirement that all rows (representing the file's records) of a given spreadsheet has the same number of columns. Thus a spreadsheet is logically equivalent in structure to a fixed-length, fixed field, sequential file.

The spreadsheet metaphor is also derivable from relational database theory as follows. A spreadsheet, in its most basic form, is a rectangular grid. A column's content and domain (set of allowable values, i.e., pool of values for an attribute (Teorey)) are defined by the column width, data type (e.g., currency, fixed decimal, date, string, etc.), and the column heading (attribute name). Columns are attributes. Column cell values embody attribute values; tuples are rows. The set of column headings with the visual formatting characteristics (such as column width) are akin to a relation's heading (fixed set of attribute pairs that define the domain of the attribute, each attribute belonging to only one domain (Teorey)). Thus the spreadsheet metaphor can be derived from the representation of a relation as a table as well as from the IS perspective as a group of related files.

This article is divided into seven sections - this Introduction being the first. Second is an anecdotal survey of the treatment of relational theory in computer literacy. Section 3 reviews the consequences of designing relational databases without a theoretical foundation and presents justification for teaching these concepts. Section 4 presents a review of assimilation theory concepts that becomes the foundation for Section 5 in which the spreadsheet metaphor is presented. The final sections chronicle the practical application of the metaphor in the classroom and end with a conclusion.

2. Current Database Education

The importance of the database topic in formal IS education is well documented. Studies have addressed the general technical knowledge and skill requirements of IS personnel (Baroudi, Breivik, Marcum, Neslon). The Association for Computing Machinery (ACM) curriculum committee makes periodic recommendations regarding the content of model curricula for computer science and IS disciplines. At least one and most often several database courses are recommended at all levels: graduate and undergraduate (Kung,Wu, Chrysler, Gorgone, Denning).

Current literature promotes use of relational database software rather than older network and hierarchical approaches. Studies have addressed specifically the content and presentation sequence of IS database courses using relational databases (Wilkins, Connolly, Robbert,Keys). Relational concepts presented as theory or practice application are widely recommended topics. Specifically championed are the theoretical topics of conceptual data modeling and normalization. Classroom projects involving the design and implementation of a relational database are common exercises that demonstrate and reinforce theory.

With respect to non-IS database education little emphasis is given to database theory (Hutchings). In support of this assertion, we appraised non-IS education by surveying textbooks used in computer literacy courses.

[1] This article presumes that the reader possesses a basic understanding of spreadsheet software. References will presume the "typical" orientation with rows being records and columns being fields. There is no loss of generality if the orientation is reversed.

[2] All considered computer literacy textbooks shared similar formats and covered three basic PC applications: (1) word processing, (2) spreadsheets, and (3) database. Universally, word processing is introduced initially. Being the least abstract of the three, it tends to build student confidence and thus decrease student anxiety because students are familiar with the written or typed word. Spreadsheet software follows word processing and is deemed more difficult, being less familiar as well as more quantitative and abstract. Neither topic possesses significant underlying theory regarded as critical for application development. A keystroke or rote learning approach (discussed later) is appropriate for these applications.

In the surveyed texts, the database section is the final topic, and this is appropriate – database being the most abstract topic. Unfortunately textbooks continue the keystroke methodology and relational theory presentation is virtually non-existent. Students are presented only the most basic concepts (such as file or field) using metaphors and terminology from an IS-based perspective. In a literacy class, it is difficult to build on this perspective or associate it with prior knowledge.

Frequently, students use aftermarket books and software manuals accompanying commercial software. These are only marginal improvements. Such sources contain in-depth, keystroke-oriented instruction for manipulating selected software packages. Basic terminology is mentioned in a brief, introductory chapter (typically six or fewer pages), followed by mechanical or keystroke instruction. There is no coverage of important relational concepts. For example, the critical concepts regarding database normalization are not found in any surveyed sources.

Summarizing, we believe non-IS students receive database education through introductory microcomputer-based database courses, supplemented with tutorials and reference manuals that accompany software as well as aftermarket texts describing specific database management system (DBMS) software. No source presents any relational database theory. Students become end-users knowledgeable with regard to the mechanics of the specific software application(s) but ignorant of database design theory, the consequences of which are described in the following section.

[2] Contact the author for a list of text surveyed.

3. Need for Relational Theory

The lack of relational theory presents difficulties for the database designer. Students unfamiliar with theoretical foundations of database design bypass the critical steps of conceptual data modeling and normalization during application development. The consequences of bypassing the critical data modeling and normalization steps can lead to poor design, which in turn, can lead to lost, inconsistent, and redundant data (Teorey). For example, the database in Figure 1 is a multi-entity table that has data redundancy (a vendor twice and a product twice) as well as other problems associated with non-normalized databases.

VENDOR ID	VENDOR NAME	PROD ID	PROD NAME	COST
123	Acme	3095	Hair Spray	4.77
456	Floss	2002	Toothpaste	2.45
123	Acme	2002	Toothpaste	2.45

Figure 1: Multi-entity Spreadsheet

Some end-users recognize their deficiency and rely on IS specialists for critical database design. However, often end-users felt IS involvement might jeopardize their independence (Ahrens). Consequently, end-users undertake design without assistance with the aforementioned results.

We are not alone in recognizing end-users' need for an education in database theory. Others have suggested that end-user database designers will benefit from a greater knowledge of database theory (Hutchings,Robbert,Rob). The software development community has responded to end-user demand with new products. Ahrens and Sankar (Ahrens) and Bostrom, Olfman, and Sein (Sein) promote software tutors to acquaint end-users with critical material for database design. Steinberg, Faley and Chin (Steinberg) have developed software that

uses an English-based, non-contextual approach for teaching relational database design including normalization. Lim and Hunter (Lim) describe DBTool which assists the database developer in the transformation of a conceptual model to an equivalent relational model. Although these approaches show promise, we offer an alternative that is simpler, does not require special software, and is perhaps more appealing: the spreadsheet metaphor.

4. Rote and Meaningful Learning – Assimilation Theory

Our contribution to database education is the introduction of a new presentation metaphor for relational database theory, the spreadsheet. The choice of a spreadsheet metaphor is best explained using terms from assimilation theory.

Assimilation theory defines two types of learning rote learning (for memorization) and meaningful learning. "Rote learning incorporates new knowledge with existing knowledge in an arbitrary and verbatim way. Rote learners memorize information with little or no regard for its meaningful connection to prior knowledge" (Hung).

Traditional computer literacy database education consists of rote learning, whereby the student is taught the mechanics of database generation without regard for relevant relational design theory. Students learn the keystrokes necessary to generate and manipulate single-table databases. The instruction terminology is IS-based, with little or no regard for its meaningful connection to the student's prior knowledge base.

The second type of learning, meaningful learning, occurs "when an individual connects new information in a non-arbitrary and substantive manner with knowledge that already exists in memory." With meaningful learning comes a fundamental understanding of concepts underlying the newly acquired information and ability to apply those concepts to situations not yet encountered. Advance organizers are "familiar" material injected into the learning process prior to the introduction of new material. The purpose of the advance organizer is to facilitate retrieval of current knowledge from long-term memory that will be necessary and/or useful in the synthesis of forthcoming information (Mayer). Thus, models, metaphors, and analogies make learning new material easier because they organize the new material in advance for the learner (Hung).

The traditional, IS-oriented database approach fails to make effective use of advance organizers when dealing with the computer literacy student, defining relational concepts using unfamiliar terms and analogies. Virtually, all IS-oriented database classes begin defining the term "database" after the terms "field," "record," and "file" are discussed with reference to the hierarchy of data. Thus, although the hierarchy of data serves as an advance organizer to IS students, the new terms become an additional burden to the computer literacy student.

For example, the following definition, a composite of many sources, is used in our Information Systems database classes: A database is a group of logically associated files organized for storage and retrieval of data. As the typical IS students' background includes elementary programming classes in one or more of, say, Java, C#, or Visual Basic, the following association is expected: The database is composed of some number of files, each of which is composed of fixed-length records, that in turn, are composed of fixed-length fields. As the definitions of "relation," "tuple," and "attribute" are introduced, the IS student mentally references programming experiences with "files," "records," and "fields," the descending hierarchical structure of data. Thus, the hierarchy of data is used as an advance organizer, a learning facilitator for the definition of "database."

The advance organizers concept is used to introduce other relational theory concepts. The organizers are drawn from the IS student's programming background, as would be expected in IS curricula or texts. However, the background or knowledge base of the computer literacy student differs from that of the IS student. Therefore, the IS-oriented advance organizers are ineffective as they are not integrated within the end-user's knowledge base. In fact, they may serve as a learning inhibitor by increasing

the total amount of new information introduced.

We contend that the spreadsheet metaphor is effective for teaching relational database theory to students. This metaphor is proposed because (1) as an advance organizer, it lies within the students' existing knowledge base; (2) the spreadsheet, inherently, can be used as an example of relational concepts; (3) we have obtained good results using this metaphor, as presented in Section 6. To illustrate the metaphor's potential; next we explain selected relational database concepts using the spreadsheet metaphor.

5. The Spreadsheet Metaphor

The introduction of basic relational terms employs the spreadsheet metaphor at its most elementary level. The students, already familiar with spreadsheets, are introduced to relational terms using the spreadsheet terms as advance organizers. In a classroom environment, the instructor makes an conscious effort to employ interchangeably the relational and spreadsheet terms in explanation of succeeding concepts, reinforcing the terms already defined, treating as synonyms the relational term and spreadsheet counterpart. The remainder of this article is written in this style.

Continuing with the terminology development, the following relational theory terms are defined using the spreadsheet metaphor. An "instance" of the relation (tuple) is a row within the spreadsheet. The set of instances at any moment comprises the relation's "extension," and the set of columns comprises its "intension."

Concepts of intermediate difficulty are introduced with a minimum of difficulty. Consider the concept of stability of the database's intension and extension, certainly abstract topics for computer literacy students. These topics are introduced through a discussion regarding the types of changes made to a spreadsheet. Students will agree that after initial spreadsheet development, changes such as row (tuple) addition or deletion are more likely than the addition or deletion of a column (attribute). Hence, the spreadsheet student is already aware that a relation's intension (number of columns) is relatively stable, as compared with its extension (number of rows).

Further relational theory topics are introduced using the spreadsheet as an advance organizer. For example, the need for normalization (the process of organizing data into relations so as to remove or update anomalies (Lightstone)), and the nature of (cardinality of) relationships between entities are abstract topics easily explained with the spreadsheet metaphor. These topics are introduced by creating a multi-entity spreadsheet. For example, the relational university model (RUM) spreadsheet (Figure 2) might be considered a "typical" end-user spreadsheet, created to reflect the recording needs of the university.

STUD ID	STUDENT NAME	GPA	CLASS ID	CLASS NAME
113	Betty	3.51	324	History
113	Betty	3.51	411	Philosophy
423	Fred	2.76	322	History
423	Fred	2.76	233	Physics
761	Barney	2.86	322	History
956	Wilma	3.67	233	Physics

Figure 2: Relational University Model (RUM)

In the spreadsheet an instance contains attributes that describe two physical objects, students and classes. It is not uncommon for end-users designed spreadsheet rows to contain data about multiple entities (objects about which information is stored). The natural grouping and association of attributes within a row renders ease of reading. Information about different objects within an instance (row) reflects the relationships among objects: students enrolled in classes. IDs represent each entity's unique identification (primary key); the other columns represent the non-key attributes. However, this spreadsheet is not without problems.

The most apparent problem, which the students immediately identify, is data duplication. Each row represents a class taken by one student. For each individual student's classes, the attributes (field) values for ID, NAME, and GPA are replicated. Obvious duplication of information exists for the CLASS entity.

These discoveries are typically followed by a discussion of the problems inherent to redundant data. The consumption of unnecessary primary and secondary storage is readily apparent. The potential entry and maintenance errors are more subtle, but nevertheless are realized by the students as discussion progresses.

This simple spreadsheet illustrates another serious problem that exists when spreadsheet instances reflect information about more than one entity; the logical data model cannot accurately reflect the physical world. For example, consider the common circumstances that cannot be depicted with RUM: (1) Dino, a student who sits out a semester is enrolled in no classes; (2) Philosophy II, a class not being taught this semester has no students. These problems, as many others, are caused by the inability of a multiple entity instance to provide for the existence of one entity coincident with the absence of an associated entity. This general class of problems, known as data dependency problems, arises when a spreadsheet instance contains data about multiple entities. The spreadsheet requires the presence of information about all entities within a data instance. For any multi-entity spreadsheet (Figure 1), the unique identifying item (primary key) is a concatenation of the primary keys (VENDOR ID + PROD ID) of the individual entities. Should one or more entities fail to exist, the spreadsheet's integrity is comprised because instance identification becomes impossible; part of the primary key is missing.

Students invariably propose to solve both the data redundancy and data dependency problems through data instance subdivision. Intuitively, the division is according to the logical grouping of attributes, that is, by entity. Thus, students begin the normalization process, the process of decomposing a relation (table) to reduce data redundancy and data dependency.

When using database software, the normalization process consists of creating separate tables, one for each entity. Students intuitively mirror the process by partitioning the one physical spreadsheet into multiple "logical" spreadsheets, "spreadsheets within a spreadsheet." (The sub-grouping, columns contained within a spreadsheet function independently, hence the name logical spreadsheets.) Thus the student's partitioning illustrates the creation of separate relations for each entity within a database. The logical spreadsheets derived from Figure 2 are illustrated in Figure 3, the student not currently enrolled in a class (Dino) and the class not currently being taught (Philosophy II) has been added, creating an accurate reflection of the physical world.

STUDENT

STUD ID	STUDENT NAME	GPA
113	Betty	3.51
423	Fred	2.76
761	Barney	2.86
956	Wilma	3.67
734	Dino	1.59

CLASS

CLASS ID	CLASS NAME
324	History
411	Philosophy
322	History
233	Physics
322	History
511	Philosophy II

Figure 3: Partitioned Relational University Model

Students are asked to analyze the partitioned spreadsheet compared with the single-table spreadsheet of Figure 2. Students easily identify the partitioning (normalization) benefits. Logical databases are more easily modified than the equivalent, multi-entity database, as data redundancy is eliminated. The singular existence reduces resources requirements (such as memory or disk) and perhaps more importantly, reduces the likelihood of error caused by inconsistent or omitted updates. Each logical spreadsheet can be maintained independently. Attributes may be added to or deleted from one without affecting the other. Rows may be added to one spreadsheet and not the other. Row addition independence allows the existence of one entity instance (record) without requiring the presence of another, resolving the data dependency problem.

Also, students recognize immediately the need for a logical association between specific spreadsheet instances, in this case STUDENT and CLASS. Otherwise valuable information between specific spreadsheet instances is lost. For example, it would be impossible to determine the classes of a particular student or the enrollment in a particular class if the spreadsheets remain unlinked. The information is easily obtained from Figure 2 but cannot be determined

from Figure 3. The pursuit of the resolution to this problem triggers discussion of the relational concepts associated with primary and foreign keys, the features that facilitate logical associations between unique instances. Thus, students discover cardinality, the type of relationship that exists between entities.

At this juncture, students require judicious guidance supplied by the instructor. The suggestion that the needed correlations be enumerated allows the students to see them (in the physical sense). Using the un-normalized database (Figure 2) the instructor extracts the STUD ID and CLASS ID columns (attributes) and creates an intersection table (Figure 4) to enumerate the relationships.

STUDENT-CLASS

STUD ID	CLASS ID
113	324
113	411
423	322
423	233
761	322
956	233

Figure 4: Intersection Table for Relational University Model

Students discover the concept of the intersection or cross-reference table as the implementation technique for M:M (many-to-many) relationships, STUDENT-CLASS being a specific example. Students easily recognize the need to create a new, logical spreadsheet that contains the connections. The adjective logical is used because the new entity reflects nothing tangible, merely the association between STUDENT and CLASS. Students enrolled in multiple classes are represented by multiple rows in the STUDENT-CLASS entity. Similarly, classes with multiple students have multiple instances in STUDENT-CLASS. Students note that the concatenation of the STUD ID and CLASS ID keys is needed to form a unique identifier for STUDENT-CLASS instances.

This example also illustrates the principal that a normalized database (spreadsheet) does not eliminate *all* data redundancy – but *controls* data redundancy. Duplication of key values is required to facilitate the logical association between specific instances of two entities. Thus, normalization controls data redundancy by eliminating unnecessary data redundancy.

This example becomes the advance organizer for the generalized resolution of the M:M relationship, the creation of an intersection table concatenating the individual primary keys to form the intersection's primary key. In addition, the M:M relationship is an advance organizer for the 1:M (one-to-many) relationship that follows. Thus, the RUM spreadsheets are used to introduce and illustrate the advantages of implementation techniques associated with normalization.

Continuing the introduction of cardinality with the spreadsheet metaphor, a second spreadsheet, specialty merchandising model (SMM) is introduced (Figure 5). The change of example permits reinforcement of the M:M normalization process and introduction of the 1:M cardinality through the introduction of a third entity. This spreadsheet reflects the needs of a special retailer. During the example's introduction it is important to include the assumption that each product has only one vendor. The intent of this assumption is to introduce a 1:M into the database, later contrasting its implementation with that of M:M.

CUST ID	CUST NAME	PROD ID	PROD NAME	VENDOR ID	VENDOR NAME
174	Ed	637	Statue	2324	Valentine
174	Ed	459	Bow	6111	Cupid
424	Alice	637	Statue	2324	Valentine
424	Alice	823	Arrows	6111	Cupid
244	Ralph	459	Bow	6111	Cupid
198	Trixie	714	Safe	2324	Valentine

Figure 5: Special Merchandising Model (SMM)

Students identify the three entities in this spreadsheet: CUSTOMER, PRODUCT, and VENDOR and create the appropriate relation (table) with redundancy removed for each entity (Figure 6).

CUSTOMER

CUST ID	CUST NAME
174	Ed
424	Alice
244	Ralph
198	Trixie

PRODUCT

PROD ID	PROD NAME
637	Statue
459	Bow
823	Arrows
714	Safe

VENDOR

VENDOR ID	VENDOR NAME
2324	Valentine
6111	Cupid

Figure 6: Partitioned Special Merchandising Model (SMM)

The M:M relationship between CUSTOMER and PRODUCT is readily apparent to the student and easily implemented through creation of the logical spreadsheet CUSTOMER-PRODUCT (Figure 7).

CUSTOMER-PRODUCT

CUST ID	PROD ID
174	637
174	459
424	637
424	823
244	459
198	714

Figure 7: Intersection Table

Students recognize that the relationship between product and vendor is different from CUSTOMER-PRODUCT. Students can visually compare CUSTOMER-PRODUCT (Figure 7) with the VENDOR ID and PRODUCT ID columns as well as remember the example's introduction. Quickly students extract the PRODUCT ID and VENDOR ID form their respective entities (Figure 8-left) and then eliminate duplicate rows (Figure 8-right).

PROD ID	VENDOR ID
637	2324
459	6111
637	2324
823	6111
459	6111
714	2324

PROD ID	VENDOR ID
637	2324
459	6111
823	6111
714	2324

Figure 8: Extract Columns (left) - Reduced Table (right)

Each product is associated with only one VENDOR, that is, a PROD ID appears only once in the listing as compared with several listings of VENDOR. Figure 8-right illustrates visually a 1:M relationship. One VENDOR has many PRODUCTs, but each PRODUCT is supplied by only one VENDOR. In addition, the visual difference between CUSTOMER-PRODUCT (Figure 7) and PRODUCT-VENDOR (Figure 8-right) is an advance organizer to suggest that implementation of 1:M relationships is different from that of M:M.

Students, remembering the goal of eliminating redundancy, explore two open choices: (1) place PROD ID in VENDOR or (2) place VENDOR ID in PRODUCT. The choice is easily resolved. All attributes are "single-valued" (another relational term), therefore, one attribute in VENDOR cannot simultaneously "point" to multiple PRODUCT instances. However, a PRODUCT instance may reference the one associated VENDOR instance. Therefore, students invariably place VENDOR ID within the PRODUCT relation (Figure 9). Thus, this example will server as an advance organizer in the discussion of foreign key placement.

PRODUCT

PROD ID	PROD NAME	VENDOR ID
637	Statue	2324
459	Bow	6111
823	Arrows	6111
714	Safe	2324

Figure 9: SMM Product Entity

The formal introduction of the term "foreign key" proceeds naturally. A foreign key is an attribute (simple or composite) of one table whose values are required to match those of the primary key (unique identifier) of another entity (table) (Teorey). Using the SMM example, the instructor notes that the foreign key placement is critical for a 1:M relationship. The foreign key must be placed in the MANY entity instance, pointing to the ONE entity instance. This somewhat abstract discussion proceeds smoothly because the exploration for resolution of the previous example served as an advance organizer for the foreign key topic. It is easily demonstrated that the foreign key attribute need not possess the same name as the associated primary key; only the values need to be matched.

In summary, we have used this section to demonstrate that the spreadsheet metaphor may be employed to illustrate relational theory concepts at all levels of abstraction, from intermediate nomenclature to advance abstract topics such as normalization and cardinality. We use other spreadsheet examples as advance organizers during the introduction of further relational theory concepts to successfully teach data modeling to computer literacy students. Results obtained by using this metaphor are detailed in the next section.

6. Practical Application of the Spreadsheet Metaphor

The spreadsheet metaphor has been employed in the computer literacy classes at Kent State University for five semesters. Students are primarily freshmen and have diverse areas of concentration, but they are not IS majors.

Students received computer instruction following the now traditional sequence: word processing, spreadsheet, and finally database. The database portion of lectures is based on relational theory (using the spreadsheet metaphor) for database design and keystroke using Microsoft's Access. Thus, students could manipulate previously defined databases as well as design new applications.

The database design segment consisted of approximately 2.5 class hours of spreadsheet metaphor lectures over four weeks. Assignments required the students to read a problem situation and design and implement a normalized database using Microsoft Access that would support the informational needs dictated by the problem. Problem level difficulty ranged from easy (two entities and 12 attributes) to moderately difficult (six entities and 45 attributes). A sample of a midlevel assignment follows:

> Veterinarians in town can be identified by a license number. Other characteristics of the vets are their name, office address and phone number. The vet treats many dogs each of which has one owner. There are no strays. Each owner, however, can have more than one dog, and the owners have unique names. The dog's names are not unique, nor are their breeds. All owners reside with their dogs at a location that is identified by its address. The people never get their dogs mixed up because each license number is different.

Students were evaluated on the basis of enumeration of the entities, association of the attributes, the correct primary keys, and the correct foreign keys. The evaluation was done objectively; over- or under specification of attributes and/or entities resulted in a penalty to the student. [3] Summary results for the students' homework assignments are given in Figure 10. Data are presented for the five semesters prior to and after the introduction of the Spreadsheet Metaphor. Specific assignments changed each semester. The structure of the assignments (entities, attributes, etc.) did not change, therefore the mix of objective score measurements did not change. The average homework score increased pre to post introduction of the metaphor by more then four points.

PRE METAPHOR INTRODUCTION

SEMESTER	STUDENTS	HOMEWORK
2002 Fall	402	65.23
2003 Spring	387	64.07
2003 Fall	395	66.22
2004 Spring	415	64.09
2004 Fall	412	65.92
Average	402	65.11

POST METAPHOR INTRODUCTION

SEMESTER	STUDENTS	HOMEWORK
2005 Spring	398	69.290
2005 Fall	403	69.160
2006 Spring	416	69.870
2006 Fall	410	68.400
2007 Spring	394	70.790
Average	404	69.502

Figure 10: Student Homework Summary

To measure student learning about database design, students were required to answer questions about normalization in an exam during the database portion of the class as well as questions on the final. Therefore, students were evaluated both from academic (test) and practical demonstration (implementation) perspectives.

A sample test question of intermediate difficulty was:

> Given this scenario: A car has a color and is identified by vehicle identification number (VIN). The cars have a purchase cost and an owner. The owners have an address, phone number, and a social security number. A salesperson has a name, sells the cars ands has a unique tax identification number (TID). The salespeople only work at one dealership.

For this question, students were required to identify the number of entities, the number of attributes, the number of foreign keys in

[3] Contact the author for details.

the entity "car," and the cardinality between car and salesperson. A summary of student performance is given in Figure 11. Data are presented for the five semesters prior to and after the introduction of the Spreadsheet Metaphor. Although a statistical analysis has not been performed, a general upward trend can be observed perhaps indicating increasing success with the pedagogical technique.

PRE METAPHOR INTRODUCTION			
SEMESTER	STUDENTS	DB EXAM	FINAL EXAM
2002 Fall	402	70.40	73.08
2003 Spring	387	69.64	71.99
2003 Fall	395	71.09	72.57
2004 Spring	415	70.80	73.65
2004 Fall	412	69.95	71.22
Average	402	70.38	72.50

POST METAPHOR INTRODUCTION			
SEMESTER	STUDENTS	DB EXAM	FINAL EXAM
2005 Spring	398	72.65	75.15
2005 Fall	403	73.91	76.53
2006 Spring	416	73.04	75.03
2006 Fall	410	74.28	76.75
2007 Spring	394	74.96	77.26
Average	404	73.77	76.14

Figure 11: Testing Results Summary

The "hands on" perspective required students to synthesize their relational theory and keystroke knowledge of Access to develop applications. The resultant applications were generally free of the data redundancy problems that plague non-normalized databases.

In summary, the spreadsheet metaphor was used for the introduction of relational database theory concepts in a computer literacy class. Literacy students were able to read a relational database description, synthesize it, and design normalized databases; these tasks usually required only of IS students. Literacy students later demonstrated their mastery with the implementation of their designs using Access.

7. Conclusion

This article describes a methodology employed to take computer literacy students to a higher level academically. Computer literacy students were introduced to data modeling using the spreadsheet metaphor as an advance organizer for the relational database concepts. By using examples more familiar to the subject audience that the abstract concepts of, say, fields or files, the spreadsheet metaphor facilitates understanding of the foundations of relational theory and enables computer literacy students to create normalized multi-entity relational databases free of data redundancy problems associated with non-normalized databases. Preliminary results of student performance indicate an improvement in knowledge and practical skill regarding normalization and database design.

References

Ahrens, J., D., and Sankar, C. S., (1991) "Tailoring Database Training for End-Users," MIS Quarterly, vol 17. No 4, pp 419-439

Baroudi, J.J. (1995). "The Impact of Role Variables on IS Personnel Work Attitudes and Intentions," MIS Quarterly, vol 9, no 4, pp 341-356

Bostrom, R., Olfman, L., Sein, M., (1988). "End-User Computing: A Research Framework for Investigating the Training/Learning Process," Human factors in MIS, J Carey, ed. Norwood, NJ, Ablex, pp 2221-250

Breivik, P. (1998). "Student Learning in the Information Age," Oryx Press

Chrysler, E., and Van Auken, S. (2002). "Entry Level Value Versus Career Value of MIS Courses: Faculty Expectations Versus Alumni Perceptions, "Journal of Computer Information Systems, vol. 42,, no. 3, pp 38-43.

Connolly, T., and Begg, C., (2006) "A Constructivist-Based Approach to Teaching Database Analysis and Design," Journal of Information Systems Education, Spring 2006

Denning, P. & McGettrick, A. (2005). Recentering Computer Science. Communications of the ACM, vol. 48, no. 11, pp 15-19.

Gorgone, J., Gray, P., Stohr, E., Valacich. J., and Wigand, R., (2006). "MSIS 2006: Model Curriculum And Guidelines For Graduate Degree Programs In Information Systems," Communications of the Association for Information Systems, vol 17, 2006, pp 1-56

Hung, W., Chao, C., (2007). "Integrating Advance Organizers and Multidimensional Information Display in Electronic Performance Support

Systems," Innovations in Education & Teaching International, v44 n2 p181-198

Hutchings, D. and Stasko, J. (2002). "QuickSpace: New Operations for the Desktop Metaphor," Extended Abstracts of the Conference on Human Factors in Computing Systems.

Keys, A.C. (2003). Using Group Projects in MIS: Strategies For Instruction and Management. Journal of Computer Information Systems, vol. 43, no. 2, 42-50.

Kung, M., Yang, S., and Zhang, Y., (2006). "The Changing Information Systems (IS) Curriculum: A Survey of Undergraduate Programs in the United States," The Journal of Education for Business, Volume 81, No. 6

Lightstone, S., Teorey, T., and Nadeau, T., (2007). "Physical Database Design: the database professional's guide to exploiting indexes, views, storage, and more", Morgan Kaufmann Press

Lim, B. and Hunter, R. (1992). "DBTool: A Graphical Database Design Tool for an Introductory Database Course," SIGCSE Papers of the Twenty-third Symposium, pp 24-27

Marcum, J., (2002) "Rethinking Information Literacy," Library Quarterly, v72 n1 p1-26

Mayer, R.E. (1979). "Can Advance Organizers Influence Meaningful Learning?" Review of Educational Research, vol 49, no 2, pp 371-383

Neslon, R. and Lyons, N., (1991). Educational Needs as Perceived by IS and End-User personnel: A Survey of Knowledge and Skill Requirements," MIS Quarterly, vol 15, no 4, pp 503-536

Rob P., and Adams, C.N., (1990). "Microcomputer Databases in the Classroom: Its Time to Pay the (design) Piper,: Journal of Computer Information Systems, vol 31, no 1, pp 18-24

Robbert, M., Wang, M., Guimaraes, M., and Myers, M.E. (2000). The Database Course: What Must be Taught. SIGCSE Bulletin, vol. 32, no. 1, pp 403-404.

Steinberg, G., Faley, R., Chin, S., (1994). "Automatic Database Generation by Novice End-Users Using English Sentences," Journal of Systems Management, vol. 45, no 3, pp 10-15

Teorey, S. Lightstone, T. Nadeau, (2005). "Database Modeling & Design: Logical Design, 4th edition", Morgan Kaufmann Press.

Tucci, L., (2005). "College Students Continue To Shun Computer Science," CIO News, August 2005.

Wilkins, M., & Nolltt, C., (2000). "Critical Skills of IS Professionals: Developing a Curriculum for the Future," Journal of Information Systems Education Vol 11, no. 3-4, pp 105-110.

Wu, J., Chen, Y., Chang, J., Lin, B., (2007). "Closing off the Knowledge Gaps in IS Education," International Journal of Innovation and Learning, vol 4, no. 4. pp 357 - 375