A Critical Evaluation Database Textbooks, Curriculum and Educational Outcomes

Roy Morien
Research Fellow
roymorien@hotmail.com
Centre for Extended Enterprise and Business Intelligence
Curtin Business School
Curtin University of Technology, Perth, Australia

ABSTRACT

Thirty year's experience in the IST industry, including 20 years teaching IS subjects, such as database design, have convinced me that IST education is not done well. Whether this manifests itself as poor teaching, or by poor learning, is a moot point, but the outcome seems to be the same. Graduates venture out into the professional world apparently poorly equipped both technically and managerially.

This is demonstrated by the many horror stories that abound, especially about poor database design. Given that databases are the central focus and foundation of most business systems, and a major resource in both time and effort to develop and maintain, database education needs to be of the highest order of relevance, practicality and correctness.

The problem seems to lie to a considerable extent in the textbooks that are available for college courses. These have problems of fact and process in abundance.

This paper narrates some of the experience of the author in anecdotal fashion, but presents a significant analysis of a number of leading textbooks, highlighting the pedagogical problems that abound in them.

Keywords: Entity Modelling, Relational Modelling, Database Education.

1. INTRODUCTION

This is not a research report, although it does contain elements of research. It is more a personal experience report discussing matters of relevance in Information Systems Education, and to Information Systems teaching academics.

There can be little doubt that the state of system development and project outcomes is still, to this day, in a somewhat parlous state. It is very difficult to find an academic paper published in journals or presented at IS conferences and that does not at least imply that things can be done much better in systems development, and at worst that system development failures are almost an inevitable and very costly outcome. Of recent times the "agile database' proponents have joined the fray (Ambler (2003); Fowler (2003); Morien (2005))

These problems start, in my view, with IS education, and especially with the state of database textbooks. IS education needs to improve, and become more rigorous, a common vocabulary needs to be developed, and essentially a "Database Book of Knowledge" needs to be created, to overcome the substantial and significant fragmentising and singularising of database theory, practice and terminology. Database curriculum especially, as the topic being addressed in this paper, needs to be relevant, practical and correct. It should also contain a significant slice of thoughtful discussion about the business environment, and not be all about technicalities.

I have recently left an information systems school at a university where I taught for 20 years. I taught database, programming, systems development methods and system development technologies. Being also a qualified and experienced accountant, I always tried to teach these matters in a business context and relevance. So I have more than a little experience in IS education. I have also suffered the frustrations of conservatism in curriculum that was almost mind-numbing.

Regrettably, I must conclude from my many years of teaching these subjects that in general many topics are taught badly. Foremost amongst these is the general topic of database, in which I include Entity Relationship Modelling, Logical Data Modelling and Relational Database Design. I could also say that many topics are learned badly, by students who often do not see the relevance of what is being taught to them. So it is not a one-way problem, or perhaps indicates that a more practical way of teaching and learning is called for.

As also being a long-time practitioner in IST, I have always been excited about the field. It is an exciting career, and an important one, and it is a dreadful shame when it is rendered down to its technical basics, and then taught in a confused and "cookbook" fashion.

It should be a matter of great concern to us, the educators, if our students graduate into professional positions ill-equipped, or poorly taught, and create the systems monsters that seem to be prevalent in IT systems today.

To approach this subject, I would first like to recount some conversations and experiences that I have had over those years. But before the criticism is levelled at me that I am recounting only one person's experience, I will also present some analysis of a number of database textbooks that have been offered for use in university and college courses over the past 20 years.

Perhaps I am taking too much upon myself, seemingly placing myself in the role of "grand old gentleman of IS". But, then, I am one of the early generation of IS practitioners who remember NCR, and ICL, and even CICS. I was there when there were no PC's, just microcomputers with interesting names like Ohio Scientific, Commodore PET, Altair and Cromemco, and Z80's competed with 8080's and 6502's for the market, and CP/M was the predominant operating system. I was also there when IMS was prevalent, and ADABAS was new, and System/R was still a research project. I started in computing in 1976, and my first experience of the peculiarities of the IT industry was when I was approached by a Project Manager in charge of a major accounting system development (Yes, they didn't have accounting packages back then). who astonished me was his question … "*We have been asked to develop a Debtors Trial Balance program, and we don't know what it is. Is it like a Balance Sheet?"*. My qualifications and experience in Accounting, and in Corporate Law and Administration, and business and management background and knowledge seemed well placed at that time.

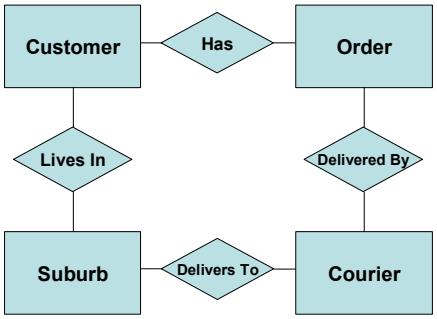
2. CONVERSATIONS FROM THE DARK SIDE: INDUSTRY

Recently in Singapore I was told about a database system that was totally devoid of any referential integrity constraints. Even the department manager was sanguine about this, happy to advise that all that needed to be done to delete the orphaned records was to go to another screen from where those "child" records could be deleted. The fact was that someone had designed that database with this appalling lack of correct and proper consideration for that important matter; Referential Integrity. Where had they learned their craft?

Then in Hong Kong I was shown a Data Diagram purporting to be the schema diagram for a major student administration system at a prominent Hong Kong university. I could not believe how bad it was, with sets of columns in a record such as Result1, Result2 …Result24 (because students had to study twenty four subjects in their course). This schema was so unnormalised as to be an affront, even to the most pragmatic relational database designer. Who had educated the designer of that database schema?

At another time, I was contacted by a graduate in Jakarta, who asked my advice on how to deal with a database table that had a record with about 75 columns designated as being available for whatever purpose an individual designer might assign to them. One problem was that any given designer had no idea which columns had been assigned for what purpose by any other designer and which columns were still available and unused. My advice to him was tantamount to "shoot yourself"; it was the best I could think of at the time.

At one time I personally undertook some contract work for a major telecom organisation. Amongst other tasks, I worked on a small system that controlled the ordering of and distribution of telephone directories to about 600,000 homes and businesses. The Entity Relationship Diagram that they had created is shown here.



This diagram showed that a Customer had an Order that was delivered by a Courier to that Customer, because the Courier delivered to the Suburb that the Customer lived in.

This was fine, until it was realised that the Order quantity could vary over time, as the Customer ordered more or less books. There was no differentiation between current order quantity and delivered quantity. So, when the Courier was paid for the books delivered, they were paid for the number of books that was on order at the time that the payment was calculated. This frequently was different from the number actually delivered. Another problem arose when another Courier actually delivered the books, because the Courier designated as delivering to that Suburb had not been available at the time of delivery. So the outcome was that often the wrong Courier was often paid often for the wrong number of books delivered. There were many other problems in this system, requiring a significant rewrite that cost more than the original development cost. Unfortunately, the designers were recent graduates from my University school.

At a government department where I was called in by a business area manager, there were three database schemas defined for a particular high value system. There was the schema defined by a consultant, who was talking to the major users. There was the schema designed by the DBA, who had his own ideas on the structure – for efficient access, of course. Then there was the schema implied by a group of potential users who were designing the system based on their screen designs. Unfortunately, these three schemas did not agree with each other. For a start, the DBA created his schema almost off the top of his head, and didn't talk to the clients, for two reasons. First, he was affronted that a consultant had been brought in, and second, he felt that the design of the database was a technical matter that the clients wouldn't understand. The outcome was actually quite chaotic, very expensive to rectify, and caused a lot of anger and resentment and conflict. Actually, there was in fact a fourth schema developed by myself, and validated by the creation of a prototype system. This schema in fact proved to support the business requirements better than any of the other three, but was ignored because the IT Manager felt affronted that I had been commissioned by a user area manager against his (the IT Manager) better judgement.

I could go on for many pages. After all, thirty years is a long time to accumulate a lot of anecdotes. I am quite confident that any one with my length of service and experience in this industry could also recount similar horro stories.

What I do want to point out is the international flavour of this problem. Singapore, Jakarta, Hong Kong, Australia; these problems are obviously universal.

3. ANECDOTES FROM ACADEMIA

When I attended an academic conference (the Australasian Conference on Information Systems - ACIS) I was astonished to hear, at a session being presented by an academic from the host campus, the statements "*We have a big problem. In the first semester, our students define their ER Model, and they usually get excellent marks for it*", but then "*The trouble is, when they come to implement it in the second semester, they find out that it is wrong, and cannot be properly implemented. I don't know what we can do to overcome this problem*". Having been a keen advocate of prototyping development for many years prior to this, and having had significant success developing systems in an evolutionary fashion for a long time, the answer was very clear to me – evolve the ER Model and implement it at the same time, thus validating every component of the ER Model as it is identified and defined. What I was obviously hearing was an outcome of the seeming academic reluctance to embrace or teach evolutionary development methods.

But then, in an attempt to be helpful and provide guidance to his students, an academic of my acquaintance issued an FAQ information sheet to his project students. The very first 'question' was "*Our client wants a*

prototype in 3 months. What do we say? ***Answer:*** *Say NO! One of the outcomes of this* (first of two) *unit is to produce an Analysis and Design document which will be used in the following unit to build the system"*. This is insisting on precisely the same predicament as described by the puzzled and desperate academic from that prior conference.

Following on this same train of thought, I saw one of the project supervisors about to leave the premises with a pile of large, impressive looking reports. He told me that he was off to spend the weekend assessing these documents. He seems somewhat startled when I asked "How? What assessment criteria will you apply?" In subsequent discussion I pointed out that all he could do, in his assessment, was to consider the neatness of the diagrams, the perhaps weigh the documents and use the large number of pages as indications of the excellence or otherwise of the specification contained in those documents. There was no way that he could be sure of the validity of the content. How could he know if that specification in any way reflected the needs and requirements of the client?

At the start of a new semester, which was at the start of the student project when I was given overall supervision of those projects, I ran a short test on ER Modelling and database design matters. Here are some of the outcomes.

In answer to the question *'Why is a Relational Database called "relational"?'* nearly all of the students answered *'Because you can represent relationships in it'*. This is, of course, quite wrong. Those students who didn't answer this way answered nothing. Not one student could answer the questions *'Who was Dr. Edward Codd?'* and *'Who was Professor Peter Chen?'* In answer to the question "What is an associative entity?" those who answered said "It replaces a Many-to-Many relationship", which is also quite wrong; certainly as an unadorned response. In fact, most students believe that in the ER Model it is wrong to have a Many-to-Many relationship, and it must be replaced by two One-to-Many relationships, which again is quite wrong, unadorned or otherwise, both from a modelling viewpoint and from a semantic viewpoint. In the ER Model the Many-to-Many relationship is a totally valid semantic construct, and to replace it with two One-to-Many relationships does not maintain the semantics and logic of that situation (apart from being somewhat contradictory to the idea of automatically replacing the relationship with an associative entity). What was also clear was that none of the students could differentiate between an ER Diagram and a Relational Diagram; basically they had always been taught that they are one and the same thing. This is what they were taught, and this is what their textbook also seemed to support.

The outcome of this test was a disaster. Most or all of the students answered wrongly because they had been taught that wrong fact, or had not been taught at all. This was notwithstanding the fact that they had covered ER Modelling and database design issues in at least two prior units. More troublesome is the fact that some of this misinformation is perpetuated in textbooks, which I will discuss later.

During my time as a project supervisor, I began to realise that some groups, when designing their database, had an autoincrement integer field as the primary key of the table on all and every table. One reason this came to my attention was that some groups were having difficulty in processing the referential integrity rules because of this. In another case, the system had to append the data from a number of tables in remote locations into a centrally located table, and the autoincrement field was causing problems. When I asked why there was this inevitable autoincrement field, I was told 'Because we were told that you can't have a string field as a primary key'. Apparently this was an efficiency issue. The tutor who told them this had a degree in computer science from Brazil. Again the international dimension of this problem was clear.

I have been assured by academics that entities do not have identifying attributes. Only tables have these and they are called primary keys.

Students are frequently taught that Referential Integrity is what you do when you declare foreign keys in the database tables and specify cascade deletes as automatic actions. What they aren't taught is that Referential Integrity is a general principle and practice, of considerable significance and import in database processing.

4. A CRITIQUE OF THE TEXTBOOKS

I will first address a number of topics that I consider relevant and in some cases central to database curriculum, and see how these topics are treated in the textbooks.

5. WHAT IS A RELATIONAL DATABASE?

Well, first, to provide the correct answer; it is a database where the data is stored in tables, based upon the mathematical principles of sets and relations, as described by Dr. E Codd in 1970 (Codd, 1970). The term 'relational' comes from this reference to relations.

In Palinski (1997, at page 16) the question is put, and answered, in this manner: *"What is a Relational Database?* (it is) *... a set of tables or holders of records that are 'related' to each other by a common value".* At a substantial way through the book, at page 112, when discussing Set Operators, the almost throwaway statement is made that *"Relational databases are founded on set theory"*. However, I can find no explanation of the importance of this. Nor does this statement in any way negate or explain the implication clearly to be drawn from the earlier statement that it is relational because you can relate tables to each other by a common value.

I searched in vain for any mention of Codd. Personally, I think a course on Relational Databases without mention of Codd is like a medical course that never mentions Pasteur, or a psychology course that makes no mention of Freud or Jung.

In furthering my search, I looked for "relation", and unfortunately found it. I say unfortunate because the term was used in *"This type of table is called a* relation table *… Relation tables do not have any significance without a tie to a base table"*. The type of table being referred to is where, if we were referencing this back to an entity model, would be how a many-to-many relationship between two entities would be represented in the Relational Model. It is referred to in other texts as an association table. However, it is almost the definition of a weak entity from Courtney & Paradice (1992) which is "An entity whose existence depends on another entity ..". Batini et al. define weak entity as *"entities that have only external identifiers"*. Awad & Gotterer (1992) do not mention weak entities at all, neither it seems does Stamper & Price (1990), nor Watson (1996), Post (1999) and Pratt & Adamski (2002). Apart from all of this, whatever it is, it is certainly not a "relation table". In correct relational database terminology, a table is called a relation; a relation is not a special kind of table, as implied here.

The bottom line here is that often the concept of relational databases is wrongly taught, and wrongly defined in textbooks. I find this a great pity, because if students were taught from the roots of the concept, they would have a much better comprehension, and a greater ability to design a relational database correctly. And I am not talking about arcane relational calculus or algebra.

6. WHAT IS AN ASSOCIATIVE ENTITY?

According to Satzinger et al (2002), this is a concept stated, at p. 173 as *"A data entity that represents a many-to-many relationship between two other data entities" … "…Analysts often discover that many-to-many relationships involve additional data that must be stored…the solution is to add a data entity to represent the relationship between* (the entities) *… sometimes called an associative entity".* Whatever else might be said about this quote (such as the complexity of the statements), one thing seems obvious to me; is this statement saying that a data entity represents a many-to-many relationship *per se*, or just when it has attributes? If it is the existence of attributes in the relationship that is the crux of the matter, then what about a one-to-many relationship that has attributes? Or are these authors denying that possibility?

I will discuss that matter elsewhere.

Looking elsewhere in the textbooks, Rob and Coronel discuss associative entities and relationship in this manner. Their Figure 4.4 on p.190 restates the Many-To-Many 'Contains' Relationship (as in *CLASS 'contains' STUDENT*) by turning it into what they term a composite entity. At p.83, this book states *'…by creating a composite entity or bridge entity'*. This is considered troublesome for a number of reasons. First, what this book is calling a *'composite entity'* or *bridge entity'* is the *'data entity'* of Satzinger et al, who do indicate that it is called an *'associative entity'* elsewhere in the literature. But it is also known as a Relationship Entity, or a Gerund (McFadden et al, 1999) who use both the terms Gerund

and associative entity), or an Intersection Entity elsewhere, depending on which book you read. The Rob & Coronel book states '*We also must create a composite entity between CLASS and STUDENT*'. But we can ask Why? Why *must* we do this? What is so essential that we *must*? What benefit do we gain by introducing these artefacts? And it is very confusing because of all the different names used for it. Just a point here … It is this very situation that Palinsky asserts will result in a "relation table".

McFadden et al. obfuscate this situation to a considerable degree. In the Glossary of Terms, at p.599, an Associative Entity is defined as *"an entity type that associates the instances of one or more entity types and contains attributes that are peculiar to the relationship between those entity instances".* My immediate comments would be Why is it necessary? What is wrong with leaving the Relationship as it is? That is, if you can understand from this definition that we are in fact talking about an entity standing in the place of a relationship! This definition also includes mention of entity types and entity instances. Make of that what you will.

At p.99, McFadden discusses Associative Entities in these terms: *"The presence of one or more attributes on a relationship … the relationship should perhaps instead be represented as an entity type*". At p. 224: *"…when the data modeler encounters a many-to-many relationship he or she may choose to model that relationship as an associative entity in the E-R Model"*. There is no mention of attributes here. At p.219 it was stated *"Associative entities (also called Gerunds) are formed from many-to-many relationships between other entity types"*.

McFadden et al. do try to provide criteria for identifying an associative entity. At p100 they say *"How do you know whether or not to convert a relationship to an associative entity type?"* The given list of criteria includes:

- *All of the relationships for the participating entity types are 'many' relationships.*
- *The resulting associative entity type has independent meaning to end users, and preferably can be identified with a single-attribute identifier.*
- *The associative entity has one or more attributes, in addition to the identifier.*
- *The associative entity participates in one or more relationships independent of the entities related in the associated relationship.*

Now, if that doesn't confuse the database student, then nothing will! For mine, I am still at a loss to understand exactly why an associative entity is even necessary and useful. I do not believe that it adds the slightest semantic richness to the model, and frankly just makes the whole thing more confusing by adding this extra but poorly defined concept and practice to the modelling activity and the diagram. After reading McFadden, if I didn't have 25 year's experience in ER Modeling and Relational Modeling, I think I would be mightily confused about how to identify an associative entity. That is of course, if I wasn't using a textbook that uses an entirely different term for this concept, or indeed a textbook that doesn't mention this concept at all. This leaves unanswered questions such as "What about a 1:M relationship with attributes?" and "What about an M:M relationship that doesn't have attributes?". Actually, Rob & Coronel perhaps provide a simplifying assumption that relationships don't have attributes, which is of course contradicted in most other database textbooks.

So, how would I state the matter? Simply! I have tried to teach the following ER Modeling guidelines:

- All relationships have attributes, even if they are just the identifying attributes of the entities that participate in the relationship, as foreign identifiers in the relationship.
- Relationships may themselves participate in relationships.
- All relationships, be they many-to-many, one-to-many or one-to-one, may ultimately be represented in the Relational Data Model as tables (relations). One-to-many relationships may alternatively be represented as foreign keys in the Table at the 'many' end, in the Relational Model.
- The concept of Associative Entities is redundant, unnecessary and fails to provide semantic richness and value whilst introducing a complexity into ER Modeling theory and practice.

What, might I ask, could be simpler?

7. HANDLING MANY-TO-MANY RELATIONSHIPS

My view of handling M:M relationships is simple:

- M:M relationships are a valid and necessary and frequent construct in the ER Model.
- All M:M relationships have attributes, at least the foreign attributes of the identifiers of the entities that participate in the relationship.
- M:M relationships transform into a table in the Logical Data Model.
- Relationships can participate in relationships, and this is in fact a commonly seen situation.

However, many textbooks make a huge "song-and-dance act" about this, totally unnecessarily, in my view, often erroneously, and usually unhelpfully.

On a web site (TechRepublic, 2005) where, amongst other things, there are threads of discussion, I read a thread that started "*Solve a many-to-many relationship in Microsoft Access".* My first reaction to this was This is not sensible! It is not even a problem! By the time the matter at hand is in Access, the many-to-many relationships from the ER Model have been satisfactorily represented as a table. You cannot, of course, have a many-to-many relationship between tables, so how can there still be a many-to-many relationship when you are using Access?. The various contributions to this thread that followed were most intriguing, and there were all sorts of suggestions about comboboxes, and list boxes and so on.

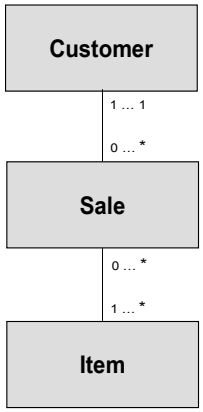
Rob and Coronel, at p.83, state *'…we can easily avoid the problems inherent in the many-to-many relationship by creating a composite entity or bridge entity'*. Apart from introducing two terms (composite entity and bridge entity) that seem clearly to state that many-to-many relationships are real and proper, and should be replaced by an entity of some naming convention, this statement implies that having a many-to-many relationship is a problem. Why is this? Aren't many-to-many relationship able to provide us with information, semantics etc. about some aspect of the business? If we look at the context of this statement in the textbook, we would see that it is made during a discussion of tables, and how tables are linked. This is actually introducing a confusion into the discussion. Properly we should say that entities and relationships are part of the Conceptual Modeling activity, when manifested as an Entity-Relationship Model, not the Relational Modeling activity, except only that when you are defining the Relational Model you are making well considered judgments about how to represent the entities and relationships – which are part of the Conceptual Model – in the Relational Model.

The situation is further complicated when we read p.210, where in Section 4.3.11, still in the Chapter dealing with Entity Relationship Modeling, the following statement is made – '*In the original E-R Model…relationships do not contain attributes…If M:N relationships are encountered, we must create a bridge between entities…The bridge is an entity composed of the primary keys of each of the entities connected*'. This statement is very troubling for a number of reasons. First, it introduces the notion of relationships not having attributes. Then it implies the requirement that a bridge entity must be created. It also implies that an M:N relationship must be restated as a bridge entity. This imperative statement is not justified anywhere, neither here nor at p.83 where there is similar discussion. Why 'must' this be so? There is no answer to be found – it is merely an assertion. A distinct problem that arises from all of this is that there does not appear anywhere in this book a clear set of criteria for converting a Relationship into a Composite Entity. The question still must be asked, however, what is the point of the whole concept of a composite entity (or bridge entity – take your pick as to terminology).

We may find an answer to this conundrum in a popular textbook on database, McFadden et al. (1999, op.cit.). However, as already discussed above, McFadden's confused and confusing guidelines and discussion on transforming many-to-many relationship into associative entities, or gerunds, is not helpful.

Post (1999) attempts to bring relational database design into the OO world. At p.35 he states *"Many-to-many associations between classes cause problems in the database design. They are acceptable in the initial diagram like Figure 2.6, but they will eventually have to be split into one-to-many*

relationships". His Figure 2.6 is shown here, and is interestingly labelled *"Class diagram or entity-relationship diagram"* thus implying that these are the same thing.



Personally, I would view Sale as a relationship, and if this was a Class Diagram would show an Association Class. In the ER Model I would deal with it in another manner.

Post also, a p.94, says *"Overview class diagrams often contain many-to-many relationships. In the relational database many-to-many relationships must be split into 2 one-to-many relationships …".* I was a little puzzled as to what an "overview class diagram" is, and if it is really different to an ordinary class diagram, but perhaps I am being too critical. Also, I would prefer to express this by the statement of some simple transformation rules from the ER Model to the Relational Model. But of course, to make these transformation rules understandable, the concept of an ER Model being different to and separate from a Relational Model must be understood. Batini, Ceri & Navathe (Batini et al, 1992) did say, at page vii, *'In fact, many organisations are discovering the need to do conceptual and logical design at the same time as they move over to relational and object-oriented database technology'.* However, I do not think they meant that the conceptual and logical models were the same thing. It is also interesting to note that Batini et al do not mention the concept of associative entities under any guise or name. My respect for this book increases when I see, at p.292 and thereabouts a clear understanding of the separation of these models, and the mapping of the M:M relationship from the ER Model into the Relational Model. However, I am a little nonplussed to discover that Batini et al. do discuss transforming such a relationship into a weak entity. Go figure! And also try to see the confusion that might be faced by a student if he or she were reading from more than one textbook on the subject.

A comment that arises from Batini et al's quoted statement above: *"the need to do conceptual and logical design"*: some books talk about Logical Design as if it is Conceptual Design. Some books do attempt to differentiate and explain the terms. The supposed 3-tier schema of Conceptual – Logical – Physical is represented in many other ways, and other terminology … again very confusing to the student.

8. SEPARATION OF MODELS

In the modern era, Model Driven Architecture (MDA) and Model Driven Development (MDD) have promised greater productivity, greater quality and represent a different development paradigm. (OMG, 2005). The crux of MDA is the different models that can be created, and the transformation rules that govern transforming one model into the next, often according to design patterns as templates for code generation.

I refer to this new and exciting development to support my view that it is highly desirable, if not essential, to understand the existence of these two concepts; different models, and transformation rules.

This can be simply applied to the ER Modeling / Relational Modeling dichotomy.

I was recently reading an ebook downloaded from the Internet … "Entity Relationship Modelling Principles (Pederson, 2005). Very soon I found this quote *"In a relational database, all entities have bonds between them. A relationship is a link between entities, and it tells us something about which relationships exist between our entities"*. I was confused. Were we still talking about Entity Modelling? Why did "relational database" suddenly appear. Relational databases have tables, and rows etc., not entities. When Peter Chen (1976) wrote his original monograph on Entity Modeling, relational databases did not exist, for all intents and purposes. Chen in fact discusses network databases but from the viewpoint of there being a separate data model. I believe that again we see here a confusion and failure to correctly differentiate between Entity Modeling and Data Modeling, associated as they are, but not the same thing. In fact, this ebook failed to mention Peter Chen at all. Again, as with my view in regard to Codd, how can we teach Entity Modelling without mentioning the progenitor of that methodology?

I believe that we need to emphasise the separate models, which will make the

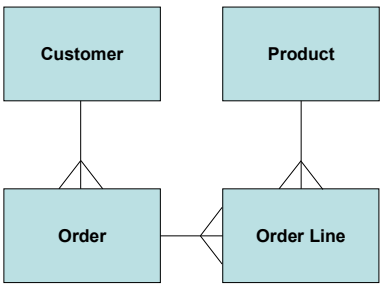
thinking much simpler and clearer to students.

However, when we have one view of the 3-tier schema as Conceptual, Logical and Physical, and another as External, Conceptual, Internal (Courtney & Paradice) and the Conceptual level is shown as "Logical Schema", then we do have a problem of contradiction, with varying views.

9. TEACHING ENTITY MODELING: MADE DIFFICULT?

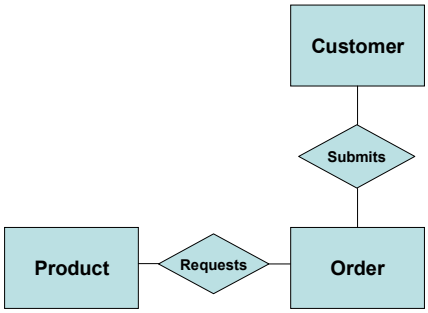
From a pedagogical viewpoint, McFadden et al. provides a case study, in my view, in confusion, contradiction and illogicality. It has all the appearance of being written by three different authors, each of whom provided chapters, without reference to each other, and standardisation of concepts.

As early as pp.10-11 we find such contradiction and confusion of terminology, concept and definition. On p.10 we have the definition of an entity as *"an object or concept that is important to the business…high level entities are: Customer, Product, Employee, Customer Order, Department"* . High level entities? What are they? Is there also a concept of "low level entities" somewhere in the text? The Enterprise Data Model is defined on p.10 also as *"a graphical model that shows the high-level entities…and associations between them"* . Well, apart from the fact that they seem to be saying that the model and the diagram are synonymous, and enshrining the concept of high level entities, I can only wonder at the contradiction here. Entities, high level or otherwise, are first defined as concepts, then are said to comprise a data model (albeit the Enterprise Data Model). Their Figure 1.3, on p.10, reproduced here, is captioned *"Segment from the enterprise data model"*.



However, on the same page, just below, the statement is made that this diagram in "Figure 3" *"is referred to as an entity-relationship diagram*".

What becomes even more confusing is this. Having stated that ER Modeling is so important that an entire chapter is dedicated to the topic, we can turn to Chapter 3, at p. 87, and see their Figure 3-1: Sample ER Diagram that contains the snippet shown in this next figure shown here.



So, for what really purports to be the same diagram, we see a totally different diagrammatic form. This form is singular and unique to this book, and is a peculiar amalgam of Chen's diagramming form, and the crows-foot form is Information Engineering, published in the late 1980s by James Martin and Clive Finkelstein. Second, where did Order Line go? Why is it shown in the previous diagram and not in the subsequent diagram? Both of these diagrams purport to say the same thing. Again, confusion, in my view. Also on this one page we have the interesting but confusing statements that an entity is a concept important to the business, that appears in an Enterprise DATA Model (my emphasis) which is also called an entity relationship model (which presumably is not yet concerned about data and database artefacts, such as tables).

While we are on p87, we can see the statement *"An entity relationship model is a detailed logical representation of the data for an organisation".* Well, so much for being an Enterprise Data Model, and so much for showing entities which are business concepts. It would be intriguing then to see what they think is a Relational Model, or a Logical Data Model. In fact, they do not seem to show a Relational Diagram or Logical Data Diagram

at all, except as relations shown as horizontal boxes with separate parts for the data fields.

10. TEACHING ENTITY MODELING: MADE SIMPLE

In a number of discussion and tutorial papers, that I regularly provided to my students (Morien, 2003a, 2003b, 2003c, 2003d, 2003e) I have outlined a simple approach to ER Modeling and Relational Modeling. In the first paper, *"Simplifying the Entity Modelling Activity – A simple set of rules to follow"*, I outline and discuss some basics. These can be summarised as:

(1) Acknowledge the separate existence and purpose of the various systems models; The Conceptual Model, manifested as an ER Model, the Logical Data Model, manifested as a Relational Data Model, the Physical Data Model and the associated Process Model.

(2) Each of these models, especially the ER Model and the Relational Data Model, has their own set of diagramming artefacts, and terminology. The ER Model deals with entities, relationships, attributes and identifiers. The Relational Model deals with tables, data fields, indexes and keys (primary and foreign).

(3) The ER Model can be transformed according to certain simple rules into the Relational Data Model, which in turn can be transformed into the Physical Data Model, and processing requirements that are inferred can be the beginning of the Process Model.

(4) All relationship types are appropriate and semantically valuable – many-to-many, one-to-many, one-to-one, and all relationships will have attributes. Relationships may themselves participate in relationships. There are some simple rules for transforming relationships into the Relational Model.

(5) The associative entity concept has no relevance, and adds nothing of semantic value to the ER model, whilst providing complexity in the modelling activity. So, forget about it!

In the paper entitled *"Identifying Entities, Relationships and Attributes"* (Morien, 2003b) I provide guidelines for identifying and validating the components of the ER Model, including the attributes. One outcome of the attribute guidelines is that the subsequent tables that will be defined to represent the entities and relationships will be automatically in 3NF and BCNF.

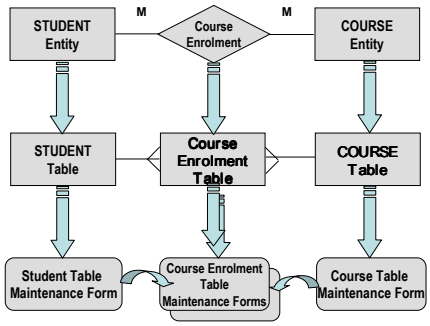
The transformation of the ER Model into the Relational Model is described in *"Representing Entities, Relationships and Attributes in the Relational Model"* In this paper I publish a set of simple transformation rules for the ER Model artefacts into the Relational Model artefacts.

In *"Visualisation of the Entity Model: Visual Development Methods for Understanding the Entity Modelling Process"* I present an agile database development approach, which I term a Focal Entity Prototyping Approach to database system development, which is heavily visual, and evolutionary. My experience of this approach is that the delivered system is correct and validated on the day that it is delivered, and meets most if not all of the user's requirements, as of the day of implementation.

Recently I published a paper at the Agile2005 conference entitled " *Agile Development of the Database: A Focal Entity Prototyping Approach*". One significant aspect of the evolutionary database development approach published there is that the database system development is done in a manner that is ER Model-driven, acknowledges the multiple model concept, and allows iterative delivery of fully validated and correct schema components and associated processing and reporting processes.

My teaching experience has been that when I describe my approach to ER Modeling and Relational Modeling, most students have what can almost be called an epiphany. Stripping away much of the technical jargon reveals the underlying techniques, which, in reality, are not that complex.

These thoughts and practices can be gathered together somewhat in this diagram.



11. CONCLUSION

I have been constrained by space to reviewing just a short list of database texts. However, I think the point has been made, and that is that database education, and the supporting textbooks, are in a confused, ambiguous and contradictory state. For a topic of such immense importance I am certain that considerably more thought and careful construction of definitions of concepts is required, and the clear, simple and logically consistent statement of theory and practice carefully adhered to.

Perhaps with the more universal acceptance of the UML this may be achieved. However, it is a bit disconcerting to realise that the UML does not have a Relational Database diagram recommendation amongst the 13 diagrams expanded in UML 2.0.

12. REFERENCES

Ambler, Scott (2003), Agile Database Techniques, Wiley

Awad, Elias M. & Malcolm H. Gotterer (1992), Database Management, Boyd & Fraser Publishing Company.

Batini, Ceri, Navathe (1992) Conceptual Database Design: An Entity Relationship Approach, Benjamin/Cummings Publishing Company

Chen, P. (1976), "The Entity-Relationship Model – Towards a Unified View of Data", *ACM Transactions on Database Systems*, March 1976

Codd, E.F. (1970) A Relational Model for Large Shared Data Banks, Communications of the ACM 13(6): pp. 377-387.

Courtney, James F. & David B. Paradice, (1992), Database Systems for Management, Irwin, 2nd Ed.

Fowler, Martin (2003) at http://www.martinfowler.com/articles/evodb.html, accessed May 19th, 2005

McFadden, Hoffer & Prescott (1999), Modern Database Management, 5th Ed., Addison-Wesley.

Merrett, T.H. (1984). Relational Information Systems, Reston Publishing Co.

Morien, R. (2003a). Simplifying the Entity Modelling Activity – A simple set of rules to follow, unpublished paper, School of Information Systems, Curtin University.

Morien, R. (2003b). Identifying Entities, Relationships and Attributes, unpublished paper, School of Information Systems, Curtin University.

Morien, R. (2003c). Representing Entities, Relationships and Attributes in the Relational Model, unpublished paper, School of Information Systems, Curtin University.

Morien, R. (2003d). Visualisation of the Entity Model: Visual Development Methods for Understanding the Entity Modelling Process, unpublished paper, School of Information Systems, Curtin University,

Morien, R. (2003e). Referential Integrity in the Relational Model, unpublished paper, School of Information Systems, Curtin University.

Morien R. (2005), Agile Development of the Database: A Focal Entity Prototyping Approach, Agile2005 Conference, Denver, Colorado, July.

Palinski, John (1997), Oracle Database Construction Kit, QUE

Pederson, Alf A., 2005, Entity Relationship Modeling Principles, ebook at http://db.ittoolbox.com/documents/document.asp?i=2883 accessed May 30th, 2005

Post, G.V (1999), Database Management Systems: Designing and Building Business Applications, Irwin/McGraw-Hill.

Pratt, Philip J. & Joseph J. Adamski (2002), Concepts of Database Management, Thomson – Course Technology, 4th Ed.

Rob & Coronel, Database Systems Design, Implementation & Management.

Satzinger, Jackson and Burd, (2002) Systems Analysis and Design in a Changing World, 2nd Edition, Course Technology

Stamper, David & Wilson Price (1990) Database Design & Management: An Applied Approach, McGraw-Hill

TechRepublic http://techrepublic.com.com/5100-6329-5034790.html accessed May 19th, 2005

Watson, Richard T. (1996), Data Management: An Organisational Perspective, John Wiley & Sons.