Entity-Relationship Modeling *Re*-revisited

Don Goelman[1] and Il-Yeol Song[2]
[1] Department of Computer Science
Villanova University
Villanova, PA 19085
don.goelman@villanova.edu
[2] College of Information Science and Technology
Drexel University
Philadelphia, PA 19104
song@drexel.edu

ABSTRACT. Since its introduction, the Entity-Relationship (ER) model has been the vehicle of choice in communicating the structure of a database schema in an implementation-independent fashion. Part of its popularity has no doubt been due to the clarity and simplicity of the associated pictorial Entity-Relationship Diagrams ("ERD's") and to the dependable mapping it affords to a relational database schema. Although the model has been extended in different ways over the years, its basic properties have been remarkably stable. Even though the ER model has been seen as pretty well "settled," some recent papers, notably [4] and [2 (from whose paper our title is derived)], have enumerated what their authors consider serious shortcomings of the ER model. They illustrate these by some interesting examples. We believe, however, that those examples are themselves flawed. In fact, while not claiming that the ER model is perfect, we do believe that the overhauls hinted at are probably not necessary and possibly counterproductive.

1 Introduction

Since its inception [5], the Entity-Relationship (ER) model has been the primary approach for presenting and communicating a database schema at the "conceptual" level (i.e., independent of its subsequent implementation), especially by means of the associated Entity-Relationship Diagram (ERD). There's also a fairly standard method for converting it to a relational database schema. In fact, if the ER model is in some sense "correct," then the associated relational database schema should be in pretty good normal form [15]. Of course, there have been some suggested extensions to Chen's original ideas (e.g., specialization and aggregation as in [10, 19]), some different approaches for capturing information in the ERD, and some variations on the mapping to the relational model, but the degree of variability has been relatively minor. One reason for the remarkable robustness and popularity of the approach is no doubt the wide appreciation for the simplicity of the diagram. Consequently, the desirability of incorporating additional features in the ERD must be weighed against the danger of overloading it with so much information that it loses its visual power in communicating the structure of a database. In fact, the model's versatility is also evident in its relatively straightforward mappability to the newer Object Data Model [7]. Now admittedly an industrial strength ERD reflecting an actual enterprise would necessarily be some order of magnitude more complex than even the production numbers in standard texts [e.g., 10]. However, this does not weaken the ability of a simple ERD to capture local pieces of the enterprise, nor does it lessen the importance of ER-type thinking in communicating a conceptual model.

Quite recently, however, both Camps and Badia have demonstrated [4, and 2 (from whose paper the title of this one is derived)] some apparent shortcomings in the ER model, both in the model itself and in the processes of conversion to the relational model and its subsequent normalization. They have illustrated these problems through some interesting examples. They also make some recommendations for improvements, based on these examples. However, while not claiming that the ER model can be all things to all users, we believe that the problems presented in the examples described in those two papers are due less to the model and more to its incorrect application.

Extending the ERD to represent complex multi-relation constraints or constraints at the attribute level are interesting research topics, but are not always desirable. We claim that representing them would clutter the ERD as a conceptual model at the enterprise level; complex constraints would be better specified in a textual or language-oriented format than at the ERD level.

The purpose of this paper is to take these examples as a starting point to discuss the possible shortcomings of the ER model and the necessity, or lack thereof, for modifying it in order to address them. We therefore begin by reviewing and analyzing those illustrations. Section 2 describes and critiques Camps' scenarios; Section 3 does Badia's. Section 4 considers some related issues, most notably a general design principle only minimally offered in the ER model. Section 5 concludes our paper.

2 The Camps Paper

In [4], the author begins by describing an apparently simple enterprise. It has a straightforward ERD that leads to an equally straightforward relational database schema. But Camps then escalates the situation in stages, to the point where the ER model is not currently able to accommodate the design, and where normalizing the associated relational database schema is also unsatisfying. Since we are primarily concerned with problems attributed to the ER model, we will concentrate here on that aspect of the paper. However, the normalization process at this point is closely tied to that model, so we will include some discussion of it as well. We now give a brief recapitulation, with commentary.

At first, Camps considers an enterprise with four ingredients: **Dealer**, **Product**, **State**, and **Concession**, where **Concession** is a ternary relationship among the other three, implemented as entity types. Each ingredient has attributes with fairly obvious semantics, paraphrased here: d-Id, d-Address; p-Id, p-Type; s-Id, s-Capital; and c-Date. The last attribute's semantics represents the date on which a given state awards a concession to a given dealer for a given product. As for functional dependencies, besides the usual ones, we are told that for a given state/product combination, there can only be one dealer. Thus, a minimal set of dependencies is as follows:

{s-Id, p-Id} → d-Id (A)
{s-Id, p-Id} → c-Date
d-Id → d-Address
p-Id → p-Type
s-Id → s-Capital

An ERD for this is given in Figure 1 (attributes are eliminated in the figures, for the sake of clarity), and the obvious relational database schema is as follows:

State(<u>s-Id</u>, s-Capital) (B)
Product(<u>p-Id</u>, p-Type)
Dealer(<u>d-Id</u>, d-Address)
Concession(<u>s-Id, p-Id</u>, d-Id, c-Date)

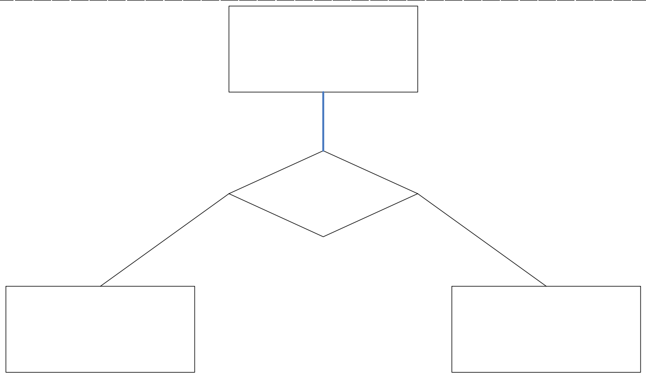

Figure 1. Example of 1:N:N relationship (from Figure 1 in [4], modified).

The foreign key constraints derive here from the two components of Concession's key, which are primary keys of their native schemas. Since the only functional dependencies are those induced by keys, the schema is in BCNF. Here Camps imposes further constraints:

p-Id → d-Id
s-Id → d-Id

In other words, if a product is offered as a concession, then it can only be with a single dealer **regardless of the state**; and analogously on the state-dealer side. The author is understandably unhappy about the absence of a standard ERD approach to accommodate the resulting *binary constraining relationships* (using the language of [12]), which he renders in a rather UML-like fashion [17], similar to Figure 2. At this point, in order to highlight the generic structure, he introduces new notation (A, B, C, D for State, Dealer, Product, Concession, respectively). However, we will keep the current ones for the sake of comfort, while still pursuing the structure of his narrative. He notes that the resulting relational database schema includes the non-3NF relation schema Concession(<u>s-Id,p-Id</u>,d-Id,c-Date). Further, when Camps wishes to impose the constraints that a state (respectively product) instance can determine a dealer if and only if there has been a concession arranged with some product (respectively state), he expresses them with these conditions:

$$\pi_{\text{s-Id,d-Id}}(\text{Concessions}) = \pi_{\text{s-Id,d-Id}}(\text{State}) \qquad \text{(C)}$$
$$\pi_{\text{p-Id,d-Id}}(\text{Concessions}) = \pi_{\text{p-Id,d-Id}}(\text{Product})$$

Each of these can be viewed as a double inclusion dependency and must be expressed using the **CHECK** construct in SQL.

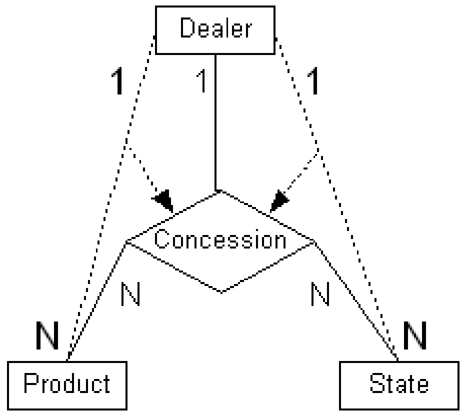

Figure 2. Two imposed FDs (from Figure 2 of [4]).

Now we note that it is actually possible to capture the structural properties of the enterprise at this stage by the simple (i.e., ternary-free) ERD of either Figure 3a [13] or Figure 3b [18]. The minimal set of associated functional dependencies in Figure 3a is as follows:

s-Id → s-Capital (D)
p-Id → p-Type
d-Id → d-Address
s-Id → d-Id
p-Id → d-Id
{s-Id, p-Id} → c-Date

One, therefore, obtains the following relational database schema, which is, of course, in BCNF, since all functional dependencies are due to keys:

State(<u>s-Id</u>,s-Capital,d-Id) (E)
Product(<u>p-Id</u>,p-Type,d-Id)
Dealer(<u>d-Id</u>,d-Address)
Concession(<u>s-Id,p-Id</u>,c-Date)

———————————————————————————————

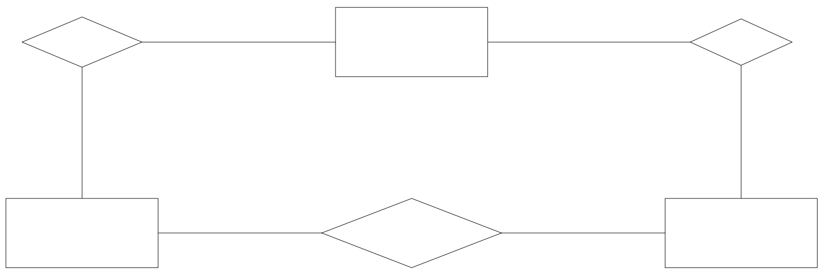

Figure 3a. A binary model of Figure 2 with Concession as a M:N relationship.

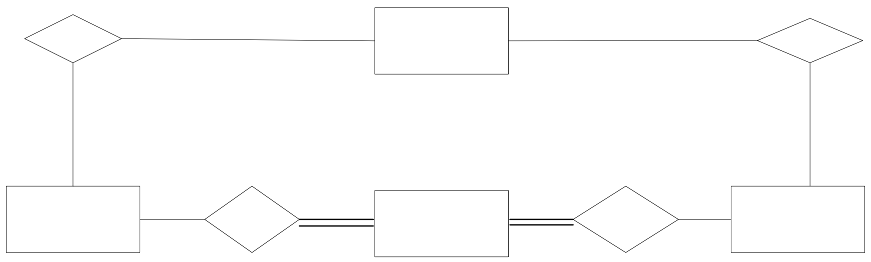

Figure 3b. A binary model of Figure 2 with Concession as an intersection (associate) entity.

———————————————————————————————

Admittedly, this approach loses something: the ternary character of **Concession**. However, any dealer-relevant information to a concession instance can be discovered by a simple join; a view can also be conveniently defined. The ternary relationship in Figure 2 is therefore something of a red herring when constraining binary relationships are imposed to a ternary relationship. In other words, it is possible that an expansion of the standard ERD language to include n-ary relationships' being constrained by m-ary ones might be a very desirable feature, but its absence is not a surprising one.

Jones and Song showed that the ternary schema with FDs imposed in Figure 2 can have lossless decomposition, but cannot have an FD-preserving schema (Pattern 11 in [13]). Camps now arrives at the same schema (E) (by normalizing his non-3NF one, not by way of our ERD in Figure 3a). The problem he sees is incorporating the semantics of (C). The constraints he develops are:

$\pi_{\text{s-Id, p-Id}}$ (Concessions) $\subseteq \pi_{\text{s-Id, p-Id}}$ (State*Product) (F)
$\pi_{\text{s-Id}}$ (State) $\subseteq \pi_{\text{s-Id}}$ (Concessions) iff State.d-Id is not null
$\pi_{\text{p-Id}}$ (Product) $\subseteq \pi_{\text{p-Id}}$ (Concessions) iff Product.d-Id is not null

The last two conditions seem not to make sense syntactically. The intention is most likely the following (keeping the first condition and rephrasing the other two):

$\pi_{\text{s-Id, p-Id}}$ (Concessions) $\subseteq \pi_{\text{s-Id, p-Id}}$ (State*Product) (G)
$(\forall s_0 \in \pi_{\text{s-Id}}(\text{State}))(s_0 \in \pi_{\text{s-Id}}(\text{Concessions})$ iff $(\exists d_0)(<s_0,d_0> \in \pi_{\text{s-Id, d-Id}}(\text{State})))$
$(\forall p_0 \in \pi_{\text{p-Id}}(\text{Product}))(p_0 \in \pi_{\text{p-Id}}(\text{Concessions})$ iff $(\exists d_0)(<p_0,d_0> \in \pi_{\text{p-Id, d-Id}}(\text{Product})))$

At any rate, Camps shows how SQL can accommodate these conditions too using **CHECK**s in the form of ASSERTIONS, but he considers any such effort (to need any conditions besides key dependencies and inclusion constraints) to be anomalous. We feel that this is not so surprising a situation after all. The complexity of real-world database design is so great that, on the contrary, it is quite common to encounter a situation where many integrity constraints are not expressible in terms of functional and inclusion dependencies alone. Instead, one must often use the type of constructions that Camps shows us or use triggers to implement complex real-world integrity constraints.

3. The Baida Paper

In his paper [2] in turn, Badia revisits the ER model because of the usefulness and importance of the ER model. He contends that, as database applications get more complex and sophisticated and the need for capturing more semantics is growing, the ER model should be extended with more powerful constructs to express powerful semantics and variable constraints. He presents six scenarios that apparently illustrate some inadequacies of the ER model; he classifies the first five as **relationship constraints** that the model is not up to incorporating and the sixth as an **attribute constraint**. We feel that some of the examples he marshals, described below in 3.3 and 3.6, are questionable, leading us to ask whether they warrant extending the model. Badia does discuss the down side of overloading the model, however, including a thoughtful mention of tradeoffs between

minimality and **power**. In this section we give a brief recapitulation of the examples, together with our analyses.

3.1 Camps Redux

In this portion of his paper, Badia presents Camps' illustrations and conclusions, which he accepts. We've already discussed this.

3.2 Commutativity in ERD's

In mathematical contexts, we call a diagram commutative [14] if all different routes from a common source to a common destination are equivalent. In Figure 4, from Badia's paper (there called Figure 1), there are two different ways to navigate from **Course** to **Department**: directly, or via the **Teacher** entity. To say that this particular diagram commutes, then, is to say that for each course, its instructor must be a faculty member of the department that offers it. Again, there is a SQL construct for indicating this. Although Badia doesn't use the term, his point here is that there is no mechanism for ERD's to indicate a commutativity constraint. This is correct, of course. Consider the case of representing this kind of multi-relation constraints in the diagram with over just 50 entities and relationships, which are quite common in real-world applications. We believe, therefore, that this kind of a multi-relation constraint is better to be specified as a textual or a language-oriented syntax, such as OCL [17], rather than at a diagram level. In this way, a diagram can clearly deliver its major semantics without incurring visual overload and clutter.

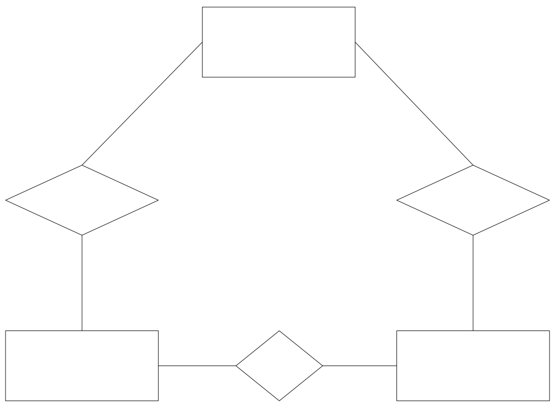

Figure 4. An example of multi-paths between two entities (from Figure 1 in [2]).

In certain limited situations [8] the **Offers** relationship might be superfluous and recovered by composing the other two relationships (or, in the relational database schema, by performing the appropriate joins). We would need to be careful about dropping **Offers**, however. For example, if a particular course were at present unstaffed, then the **Teaches** link would be broken. This is the case when **Course** entity has partial (optional) participation to **Department** entity. Without an explicit **Offers** instance, we wouldn't know which department offers the course. This is an example of a **chasm trap** which requires an explicit **Offers** relationship [6]. Another case where we couldn't rely on merely dropping one of the relationship links would arise if a commutative diagram involved the composition of two relationships in each path; then we would surely need to retain them both and to implement the constraint explicitly.

We note that allowing cycles and redundancies in ERD's has been a topic of research in the past. Atzenti and Parker [1] advise against it; Markowitz and Shoshani [15] feel that it is not harmful if it is done right. Dullea and Song [8, 9] provide a complete analysis of redundant relationships in cyclic ERD's. Their decision rules on redundant relationships are based on both maximum and minimum cardinality constraints.

3.3 Acyclicity of a Recursive Closure

Next, Badia considers the recursive relationship **ManagerOf** (on an **Employee** entity). He would like to accommodate the hierarchical property that nobody can be an indirect manager of oneself. Again, we agree with this observation but can't comment on how desirable such an ER feature would be at a diagram level. Badia points out that this is a problem even at the level of the relational database, although some Oracle releases can now accommodate the constraint.

3.4 Fan Traps

At this point the author brings Figure 5 (adapted from [6], where it appears as Figure 11.19(a); for Badia it is Figure 2) to our attention. (The original figure uses the "Merise," or "look here" approach [17]; we've modified it to make it consistent with the other figures in this paper.) The problem, called a **fan trap** arises when one attempts to enforce a constraint that a staff person must work in a branch operated by her/his division. This ER anomaly percolates to the relational schemas as well. Further, if one attempts to patch things up by including a third binary link, between **Staff** and **Branch**, then one is faced with the commutativity dilemma of Section 3.2. In general fan traps arise when there are two 1:N relationships from a common entity type to two different destinations. The two typical solutions for fan traps are either to add a third relationship between the two many-side entities or rearrange the entities to make the connection unambiguous. The problem in Figure 5 here is simply caused by an incorrect ERD and can be resolved by rearranging entities as shown in Figure 6. Figure 6 avoids the difficulties at both the ER and relational levels. In fact, this fix is even exhibited in the Connolly source itself. We note that the chasm trap discussed in Section 3.2 and the fan trap are commonly called **connection traps** [6] which make the connection between two entities separated by the third entity ambiguous.

———————————————————————————————

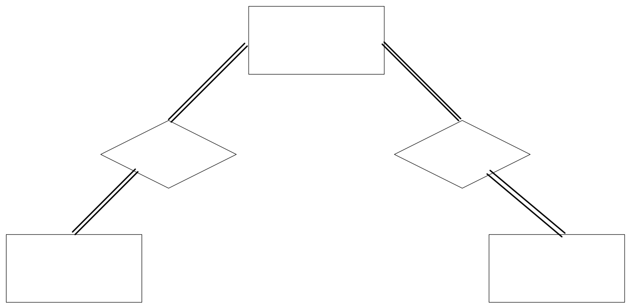

Figure 5. A semantically wrong ERD with a fan trap (from Figure 2 in [2] and Figure 11.19(a) from [6]).

———————————————————————————————

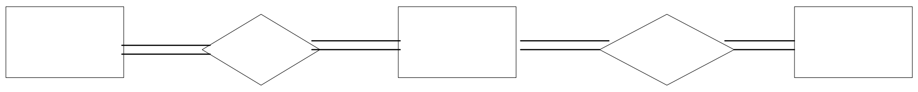

Figure 6. A correct ERD of Figure 5, after rearranging entities.

———————————————————————————————

3.5 Temporal Considerations

Here Badia looks at a **Works-in** relationship, M:N between **Employee** and **Project**, with attributes **start-date** and **end-date**. A diagram for this might look something like Figure 7b; for the purposes of clarity, most attributes have been omitted. Baida states that the rule that *even though en employee may work in many projects, an employee may not work in two projects at the same time* may not be represented in an ERD. It appears impossible to express the rule, although the relationship is indeed M:N. But wouldn't this problem be solved by creating a third entity type, **TimePeriod**, with the two date attributes as its composite key, and letting **Works-in** be ternary? The new relationship would be M:N:1, as indicated in Figure 7c, with the 1 on the **Project** node, of course. In figures of 7a through 7d, we show several variations of this case related to capturing the history of works-in relationships and the above constraint. We'll comment additionally on this in Section 4.

———————————————————————————————————————

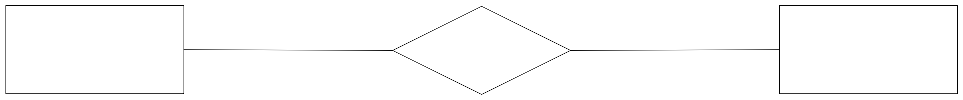

Figure 7a. An employee may work in only one project and each project can have many employees. The diagram already assumes that an employee must work for only one project at a time. This diagram is **not** intended to capture any history of *works-in* relationship.

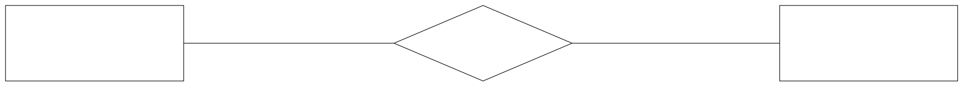

Figure 7b. An employee may work in many projects and each project may have many employees. The diagram assumes that an employee may work for many projects at the same time. This diagram is also **not** intended to capture any history of *works-in* relationship.

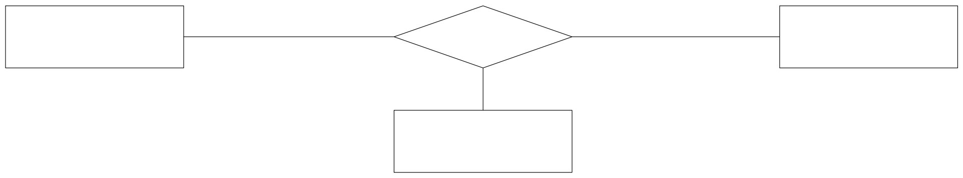

Figure 7c. An employee may work in only one project at a time. This diagram can capture a history of *works-in* relationship of an employee for projects and still satisfies the constraint that an employee may work in only one project at a time.

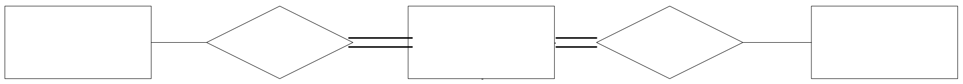

Figure 7d. In Figure 7.c, entity TimePeriod is not easily materialized, we can reify the relationship Works-in to an intersection entity. This diagram can capture the history of works-in relationship, but does not satisfy the constraint that an employee may work in only one project.

———————————————————————————————————————

Employee

3.6 Range Constraints

While the five previous cases exemplify what Badia calls **relationship constraints**, this one is an **attribute constraint**. The example given uses the following two tables:

Employee (<u>employee_id</u>, rank_id, salary, …)

Rank (<u>rank_id</u>, max_salary, min_salary)

The stated problem is that the ERD that represents the above schema cannot express the fact that the salary of an employee must be within the range determined by his or her rank. Indeed, in order to enforce this constraint, explicit SQL code must be generated. Baida correctly sates that the absence of information at the attribute level is a limitation and cause difficulty in solving semantic heterogeneity. We believe, however, that information and constraints at the attribute level could be expressed at the data dictionary level or in a separate low level diagram below the ERD level. Again, this will keep an ERD as a *conceptual model* at enterprise level without too much clutter. Consider the complexity of representing attribute constraints in ERDs for real-world applications that have over 50 entities and several hundreds of attributes. The use of a CASE tool that supports a conceptual ERD with its any low level diagram for attributes and/or its associated data dictionary should be a right direction for this problem.

4 General Cardinality Constraints

While on the whole, as indicated above, we feel many of the alleged shortcomings of the ER model claimed in recent papers are not justified, some of those points have been well taken and are quite interesting. However, there is another important feature of conceptual design that we shall consider here, one that the ER model really does lack. In this section, we briefly discuss McAllister's general cardinality constraints [16] and their implications.

McAllister's setting is a general n-ary relationship **R**. In other words, R involves **n** different **roles**. This term is used, rather than **entity types**, since the entity types may not all be distinct. For example, a recursive relationship, while binary in the mathematical sense, involves only a single entity type. Given two disjoint sets of roles A and B, McAllister defines **Cmax(A,B)** and **Cmin(A,B)** as follows: for a tuple <a>, with one component from each role in A, and a tuple <b>, with one component from each role in B, let us denote by <a,b> the tuple generated by the two sets of components; we recall that A and B are disjoint. Then Cmax(A,B) (respectively Cmin(A,B)) is the maximum allowable cardinality over all <a> of the set of tuples <b> such that $<a,b> \in \pi_{A \cup B}(R)$. For example, consider the **Concession** relationship of Figure 1. Then to say that

Cmax({State, Product},{Dealer}) = 1 is to express the fact that

{s-Id, p-Id} $\rightarrow$d-Id. And the condition **Cmin({Product},{State,Dealer}) = 1** is equivalent to the constraint that **Product** is total on **Concession**. Now, as we see from these examples, Cmax gives us information about functional dependencies and Cmin about participation constraints. When B is a singleton set and A its complement, this is sometimes called the “Chen” approach to cardinality [11] or “look across”; when A is a singleton set and B its complement, it is called the “Merise” approach [11] or “look here.” All told, McAllister shows that there are $3^n-2^{n+1}+1$ different combinations possible for A and B, where n is the number of different roles.

Clearly, given this explosive growth, it is impractical to include all possible cardinality constraints in a general ERD, although McAllister shows a tabular approach that works pretty well for ternary relationships. He shows further that there are many equalities and inequalities that must hold among the cardinalities, so that the entries in the table are far from independent. The question arises as to which cardinalities have the highest priorities and should thus appear in an ERD. It turns out that the Merise and Chen approaches give the same information in the binary case but not in the ternary one, which becomes the contentious case ($n>3$ is rare enough not to be a serious issue). In fact one finds both Chen [as in 10] and Merise [as in 3] systems in practice. In his article, Genova feels that UML [17] made the wrong choice by using the Chen method for its Cmin’s, and he suggests that class diagrams include both sets of information (but only when either A or B is singleton). That does not seem likely to happen, though.

Still, consideration of these general cardinality constraints and McAllister’s axioms comes in handy in a couple of the settings we have discussed. The general setting helps understand connections between, for example, ternary and related binary relationships as in Figure 2 and [12]. And it similarly sheds light on preservation (and loss) of information in Section 3.5 above, when a binary relationship is replaced by a ternary one. Finally, we believe that it also provides the deep structural information for describing the properties of decompositions of the associated relation schemas. It is therefore indisputable in our opinion that these general cardinality constraints do much to describe the fundamental structure of a relationship in the ER model; only portions of which, like the tip of an iceberg, are currently visible in a typical ERD. And yet we are not claiming that such information should routinely be included in the model.

5 Conclusion

We have reviewed recent literature ([4] and [2]) that illustrate through some interesting examples areas of conceptual database design that are not accommodated sufficiently at the present time by the Entity-Relationship model. However, some of these examples seem not to hold up under scrutiny.

Capabilities that the model does indeed lack are constraints on commutative diagrams (Section 3.2 above), recursive closures (3.3), and some range conditions (3.6) as pointed out by Badia. Another major conceptual modeling tool missing in the ER model is that of general cardinality constraints [16]. These constraints are the deep structure that underlies such more visible behavior as constraining and related relationships, Chen and Merise cardinality constraints, functional dependencies and decompositions, and participation constraints. How many of these missing features should actually be incorporated into the ER model is pretty much a question of triage, of weighing the benefits of a feature against the danger of circuit overload.

We believe that some complex constraints such as multi-relation constraint are better to be represented as a textual or a language-oriented syntax, such as OCL [17], rather than at the ER diagram level. We also believe that information and constraints at the attribute level could be expressed at the data dictionary level or in a separate low level diagram below the ERD level. In these ways, we will keep an ERD as a *conceptual model* at enterprise level to deliver major semantics without visual overload and too much clutter. Consider the complexity of an ERD for a real-world application that has over 50 entities and hundreds of attributes and representing all those complex multi-relation and attribute constraints in the ERD. The use of a CASE tool that supports a conceptual ERD with its any low level diagram for attributes and/or its associated data dictionary should be a right direction for this problem.

We note that we do not claim that some research topics suggested by Baida, such as relationships over relationships and attributes over attributes, are not interesting or worthy. Research in those topics would bring interesting new insights and powerful ways of representing complex semantics. What we claim here is that the ERD itself has much value as it is now, especially for relational applications, where all the examples of Baida indicate. We believe, however, that extending the ER model to support new application semantics such as biological applications should be encouraged.

The "D" in ERD connotes to many researchers and practitioners the simplicity and power of communication that account for the model's popularity. Indeed, as the Entity-Relationship model nears its 30th birthday, we find its robustness remarkable.

References

1.Atzeni, P. and Parker, D.S., "Assumptions in relational database theory", in *Proceedings of the 1st ACM Symposium on Principles of Database Systems*, March 1982.

2. Badia, A. "Entity-Relationship Modeling Revisited", *SIGMOD Record,* 33(1), March 2004, pp. 77-82.

3. Batini, C., Ceri, S., and Navathe, S., *Conceptual Database Design*, Benjamin/Cummings, 1992.

4. Camps Paré, R. "From Ternary Relationship to Relational Tables: A Case against Common Beliefs", *SIGMOD Record*, 31(20), June 2002, pp. 46-49.

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

6. Connolly, T. and Begg, C., *Database Systems*, 3d Edition, Addison-Wesley, 2002.

7. Dietrich, S. and Urban, S., *Beyond Relational Databases*, Prentice-Hall, to appear.

8. Dullea, J. and Song, I.-Y., "An Analysis of Cardinality Constraints in Redundant Relationships," *in Proceedings of Sixth International Conferences on Information and Knowledge Management (CIKM97)*, Las Vegas, Nevada, USA, Nov. 10-14, 1997, pp. 270-277.

9. Dullea, J., Song, I.-Y., and Lamprou, I., "An Analysis of Structural Validity in Entity-Relationship Modeling," *Data and Knowledge Engineering*, 47(3), 2003, pp. 167-205.

10. Elmasri, R. and Navathe, S.B., *Fundamentals of Database Systems*, 4th Ed., Addison-Wesley, 2003.

11. Genova, G., Llorenz, J., and Martinez, P., "The meaning of multiplicity of n-ary associations in UM*L"*, Journal of Software and Systems Modeling, 1(2), 2002.

12. Jones, T. and Song, I.-Y., "Analysis of binary/ternary cardinality combinations in entity-relationship modeling", *Data & Knowledge Engineering*, 19(1), 1996, pp. 39-64.

13. Jones, T. and Song, I.-Y., "Binary Equivalents of Ternary Relationships in Entity-Relationship Modeling: a Logical Decomposition Approach." *Journal of Database Management*, 11(2), 2000, (April-June, 2000), pp. 12-19.

14. MacLane, S., *Categories for the Working Mathematician*, Springer-Verlag, 1971.

15. Markowitz, V. and Shoshani, A., "Representing Extended Entity-Relationship Structures in Relational Databases: A Modular Approach", *ACM Transactions on Database Systems*, 17(3), 1992, pp. 423-464.

16. McAllister, A., "Complete rules for n-ary relationship cardinality constraints*"*, *Data & Knowledge Engineering,* 27, 1998, pp. 255-288.

17. Rumbaugh, J., Jacobson, I., and Booch, G., *The Unified Modeling Language Reference Manual*, Addison-Wesley, 1999.

18. Song, I.-Y., Evans, M., and Park, E.K., "A Comparative Analysis of Entity-Relationship Diagrams," *Journal of Computer and Software Engineering,* 3(4) (1995), pp. 427-459.

19. Teorey, T., *Database Modeling & Desig*n, 3d Edition, Morgan Kaufmann, 1999.