Distilled Entity Relationship Model

Richard Elling Moe
Department of information science and media studies
University of Bergen

ABSTRACT
We formalize the syntax of ER-diagrams, trying to explicate details left unclear in the literature either by neglect or by a lack of consensus with regard to the basic elements of ER-modelling and the restrictions on their use. Some of these restrictions and the consequences of relaxing them are discussed.

KEY WORDS
Entity relationship model, ER-diagram syntax, Conceptual modelling, Database design

1 Introduction

The Entity relationship (ER) model has come of age. It is widely adopted and has been part of common database design-practise for years. Still, nearly thirty years after its conception [6] different authors does not seem to agree about every detail about its contents. Even with decent standard textbooks, the attempt to figure out the complete picture involves reading between the lines and downright guessing.

We need a precise and complete description of ER-diagram syntax. By *syntax*, we mean the basic elements ER-diagrams are made up of and the constraints on their use. This should not be confused with *notation*. We do not adhere to any particular way of drawing the diagrams.

Formal definitions of ER-syntax are not too common in the literature. Approaches can be found in [4, 11]. We provide an alternative account, formalizing the notion of an ER-diagram, with careful focus on details of syntax. In the process we run into matters of debate, neglected points or disagreement between authors. Our main goal then is to choose between the possibilities, polling the literature and practises of ER-modelling. The result is a version of ER, distilled from the varieties found in the literature. Even if this necessarily represents mainstream ER, we get the opportunity to discuss possible novel features, though not necessarily proposing their inclusion into the model.

We focus on the pure ER-model, disregarding extensions such as specialization and categories of the EER-model.

2 Conceptual modelling

ER-modelling is part of the database design methodology and serves as a conceptual level on which the users of the database should be able to participate in the design process.

The meaning of the word ’conceptual’ has been discussed, but for our purposes, in the context of database design, it will suffice to observe that it involves the following principle:

> *The ER-model should describe the part of the ’reality’, the* mini-world*, that is to be represented in the database. It should not reflect aspects of its implementation or use.*

This is not inconsistent with the practise of database experts among themselves putting database specific details into the ER-diagram. When this happens there has been a shift of focus from the users perspective towards viewing the database itself as the mini-world.

3 The syntax of ER-diagrams

The literature on the entity relationship model varies considerably when it comes to ER-diagrams. Not only notation-wise but also with regard to the basic elements and the constraints on their use. Many accounts, it must be said, are unclear and/or incomplete on syntactic matters. Unsurprisingly, this affects the exposition of the semantics too.

3.1 Weak entity types

In particular, the so-called *weak* entity types receives inadequate treatment in many textbooks when it comes to detail and completeness. Atzeni et al. [2] are among the better expositions on this matter, and we adapt their approach using the concepts of internal and external identifiers. However, we stick to the widely used terminology of ’keys’ and ’weak’ entity types (and the use of the term ’strong’ to mean non-weak) with the notions of partial keys, identifying relationship types, owners etc. Chen [7] further classifies weak entity types into existence dependencies and ID dependencies, but we go with the bulk of standard textbooks on this matter and disregard this distinction.

3.2 Names

In ER-diagrams, all sorts of things are assigned names: Entity types, attributes, relationship types and the roles they combine/specify. When drawing an ER-diagram, you would typically label all entity types, attributes and relationship types, but not necessarily the roles. Moreover, the entity types would have unique names but otherwise the reuse of names is allowed, free of implications for the interpretation of the diagram. However, it is not necessary to allow the reuse of names. Any ER-diagram could be rewritten to avoid such ambiguity without any harm to its descriptive power.

3.2.1 The formal use of names

For our purposes, the actual choice of names will have no significance. Our need for names is only to distinguish between the different bits and pieces found in a diagram. We are not trying to replicate the labelling conventions found in the actual drawing of an ER-diagram. Rather we will assume that names are unique. Such uniqueness has its parallel in real diagrams in that no elements coincide spatially. I.e. different attributes are distinguished by the fact that they occupy locations.

Attributes should be labelled: When an entity type or relationship type holds several attributes it must be possible to tell them apart.

Roles should be labelled: Take the case of a so-called *recursive* relationship type where two roles refer to the same entity type. Then, if these roles can not be distinguished by other means, they must be given names to avoid confusion. For simplicity we demand that *all* roles should be labelled.

When all roles have unique names, there is no need for relationship types to be labelled at all: They can be distinguished by the collections of roles they are associated with.

One might be tempted to avoid names for entity types in a similar manner by relying upon the collections of attributes to set the difference. However, we should allow for the possibility that entity types have no attributes specified. This is in line with the overall desire to avoid unnecessary restrictions in the expressive power on the conceptual level[1]. Besides, it should be possible to make sense of a preliminary ER-diagram, where the attributes have yet to be specified. Hence, we require all entity types to have names. In fact, the entity type will be the name itself, which reflects the fact that what we are dealing with here is mere syntax.

Definition 3.1 *We fix three mutually disjoint name spaces:*

- *$AName$ is an infinite set of* attribute names.
- *$RName$ is an infinite set of* role names.
- *$EName$ is an infinite set of* entity type names.

[1]Furthermore, some people (of the NIAM persuasion) would claim that the distinction between attributes and relationships is artificial or even harmful [10].

3.3 The boxes, the lines and the whatnots

Now, let us scrutinize the ER-model and establish the formal notion of a diagram, with the restrictions that apply to its construction.

Definition 3.2 *An* attribute *is a finite tree where every node is uniquely labelled with an element of $AName$. We may refer to an attribute using the label of its root.*

Definition 3.3 *An* entity type *is an element of $EName$*

Definition 3.4 *A* role *is a tuple $\langle l, e, min, max\rangle$ where*

- *$l \in RName$*
- *e is an entity type*
- *$min \in \boldsymbol{N}$*
- *$max \in \boldsymbol{N} \cup \{*\}$*

Here, *min* and *max* represents the lower and upper cardinality constraints for the role. The '$*$' corresponds to an unspecified 'many' cardinality.

For notational convenience: When given a set R of roles where all entity types are members of some specific set E, we may refer to the following functions:

- $label : R \rightarrow RName$ returning the role-name of any given role.
- $etype : R \rightarrow E$ returning the entity type for a given role.
- $min : R \rightarrow \boldsymbol{N}$ returning the lower cardinality constraint of a role.
- $max : R \rightarrow \boldsymbol{N} \cup \{*\}$ returning the upper cardinality constraint of a role.

Definition 3.5 *A* relationship type *is a finite set T of roles with* $|T| \geq 2$.

Let E be a set of entity types. The relationship type T is said to be based on E *iff for all $r \in T : etype(r) \in E$.*

Definition 3.6 *A* diagram *is a pair $\langle E, R\rangle$ where*

- *E is a finite set of entity types*
- *R is a finite set of relationship types based on E.*

Associated with the diagram $\langle E, R\rangle$ are some further syntactic specifications, given by the following relations:

- $Root \subseteq AName$ specifies *precisely* which attribute names identify roots, in the sense that an attribute is a root in the diagram *if and only if* it is a member of $Root$. $Root$ should be finite.

- $Eattribute \subseteq E \times Root$ specifies precisely which attributes belong to which entity types. I.e. an entity type e has the attribute a iff $Eattribute(e,a)$
- $Rattribute \subseteq R \times Root$ specifies precisely which attributes belong to which relationship types.
- $Multivalued \subseteq Root$ specifies precisely which attributes are multi-valued. Note that only roots can be multi-valued.

In order to specify keys and weak entity types in the diagram we adapt the notions of internal and external identifiers (see [2]).

Definition 3.7 *An* identifier *for an entity type e is a pair* $\langle Attrs, Roles \rangle$ *where*

- $Attrs \subseteq \{a \mid Eattribute(e,a)\}$.
- $Roles \subseteq \{r \mid r \in \bigcup_{T \in R} T,\ etype(r) = e\}$

So, an identifier for an entity type consists of a set of attributes and a set of roles, either of which can be empty. Note that an entity type may be associated with several identifiers.

The following relation specifies all identifiers in the diagram $\langle E,R \rangle$ with the entity types they belong to:

$$Id \subseteq E \times \mathscr{P}(Root) \times \mathscr{P}(\bigcup_{T \in R} T)$$

I.e. $Id(e, Attrs, Roles)$ holds iff $\langle Attrs, Roles \rangle$ is an identifier for the entity type e.

Furthermore, for each $e \in E$ there should be some *Attrs* and *Roles* such that $Id(e, Attrs, Roles)$ holds. I.e. all entity types should have at least one identifier, even if it is just $\langle \emptyset, \emptyset \rangle$. Note that this requirement is made for convenience only: The presence or absence of the identifier $\langle \emptyset, \emptyset \rangle$ would make no difference to us, because in ER-diagrams there is no status of being *undetermined*, marking a distinction from being either strong or weak.

Clearly, the weak entity types are those associated with an identifier where $Roles \neq \emptyset$, in which case *Attrs* specifies its partial keys. The elements of *Roles* are thus the *identifying roles* referring to the *identifying relationship types*, which in turn specify the *owners* of the weak entity types. Accordingly, in cases where $Roles = \emptyset$ we may refer to the entity type as being strong, *Attrs* being the key.

Some further restrictions apply for any diagram $\langle E,R \rangle$. First,

(1) Attribute names should be unique.

Furthermore, they should belong to a single entity- or relationship type only:

(2) There is no a such that either of the following holds

a) $Eattribute(e,a)$ and $Eattribute(e',a)$ for any $e \neq e'$

b) $Rattribute(r,a)$ and $Rattribute(r',a)$ for any $r \neq r'$

c) $Eattribute(e,a)$ and $Rattribute(r,a)$ for any e and r

Role names too should be unique:

(3) For any pair of relationship types $\{T_1, T_2\} \subseteq R$:
$label(r) \neq label(r')$ for any pair $\{r, r'\} \subseteq T_1 \cup T_2$

This requirement entails that the elements of R are mutually disjoint.

Identifiers are restricted in the usual way:

(4) For any $e, Attrs$ and $Roles : Id(e, Attrs, Roles)$ implies
a) $Attrs \cap Multivalued = \emptyset$ and
b) $min(r) = 1 = max(r)$, for all $r \in Roles$

I.e. keys and partial keys can not be multi-valued, and the identifying roles should have the appropriate cardinality constraints.

A relationship type specifies the ownership for at most one weak entity type:

(5) For each $T \in R$:
$|T \cap \{r \mid Id(e, As, \{r\} \cup Rs)$ for some $e, As, Rs\}| \leq 1$

3.3.1 Groundedness for weak entity types

A further syntactic restriction is that every ownership shall be based on proper entity types. A weak entity type may well be an owner of an other weak entity type, but all such propagated ownerships must originate in a non-weak entity type. This requirement is hinted at in Chen's original paper on ER-modelling [6], and it seems to be underlying in the bulk of standard textbooks, but it is rarely mentioned explicitly. Exceptions are Atzeni et al [2] and Batini et al. [5] where the problem of circularity in the ownership structure is addressed. Thalheim [11] also discusses related 'identification problems' of weak entity types, so does Balaban and Shoval [3].

Apart from phrases like 'circularity should be avoided' we have found no formal explication of the requirement that ownership loops should be banned. Here is our proposal:

Definition 3.8 *Let* $\langle E,R \rangle$ *be a diagram.*

- *An* ownership chain *is a sequence* $\langle e_1, e_2, \ldots \rangle$*, possibly infinite, such that for each consecutive pair* e_i*,* e_{i+1} *there is* $T \in R$ *and* $\{r, r'\} \subseteq T$ *such that* $etype(r) = e_{i+1}$ *and* $Id(e_i, Attrs, Roles \cup \{r'\})$ *holds for some Attrs and Roles.*
- *An ownership chain is* grounded *iff it is on the form* $\langle e_1, e_2, \ldots, e_k \rangle$ *where* $Id(e_k, Attrs, \emptyset)$ *for some Attrs.*

Note that grounded ownership chains are, by definition, finite. If we require that in any diagram,

(6) all ownership chains should be grounded,

ownership loops would be ruled out so that no weak entity type can directly or indirectly be identified in terms of itself.

4 A second look at the syntax restrictions

Much work in the field of entity relationship modelling involves proposing enhancements to the model. This has not been our goal at all. We would be reluctant to add to the jungle of proposed constructs and notation that has grown over the years. Nonetheless, we shall look briefly at what could be the consequences of lifting some of the constraints on the ER-diagrams. (Hoping that the removal of constraints will not create a need for new notation.) Other requirements, such as (2) and (3), are so fundamental that we feel they are best left untouched.

4.1 Restriction (1)

The demand that attribute names should be unique was included for convenience, in order to simplify the *formal* notion of an ER-diagram. It is not held to be a restriction on the practise of drawing diagrams.

Still, if we were to attach special meaning to the sharing of attribute-names we propose that a *unique* domain is associated with every attribute. I.e. Attributes sharing a name should take the same kind of values.

4.2 Restriction (4)

We discuss point b) stating that a weak entity should have one, and only one, owner. Chen [7] relaxes this requirement to allow for many-many ownerships. He explains this using an example: When storing information about employees and their dependents, such as children or spouses, one might run into the situation where a dependent is associated with several employees. For instance the children having both their parents registered as employees in the database. In this case, according to Chen, a many-many ownership could be used in the ER-diagram meaning that the child should remain in the database as long as at least one parent is still registered.

We feel that Chen's argument relies a little too much on the presence of a database, and details of its use, for granting the many-many ownership the status of an entirely *conceptual* notion. We propose an alternative interpretation which is much in the same vein as Chen's but remains independent of the database, its implementation and use.

4.2.1 Groups and entities

As the name suggests, the *entity* relationship model is concerned with entities only and does not involve *groups* of entities as a basic notion. Certainly, it is possible to model various kinds of groups that might exist in a mini-world but only by viewing them as entities possessing some kind of group identifier. The idea of letting a group be identified by its members alone is simply not present in the ER-model. Here lies a weakness. Many applications involve concepts that are groups by nature. Take for instance spatial information systems: The concept of a polygon is defined solely as a collection of lines [1], but in order to fit polygons and such into the ER-model we are forced to make use of some sort of key which is conceptually redundant.

We suggest that groups of entities could be included in the ER-model for the interpretation of many-many ownerships. I.e. that a weak entity type may have a group of entities as its owner, without the implication that each of the members of the group is an owner in itself. Then a polygon could straightforwardly have a group of lines as its owner.

So far so good, but if groups are to become full members of the ER-model it seems that it ought to be possible to specify attributes that pertain to a group as such. Then it seems inevitable that new notation must be introduced for that purpose.

4.3 Restriction (5)

This requirement appears to be fundamental in almost every account of weak entity types. To our knowledge, only Thalheim [11] discusses the matter and identifies problems that would arise if the requirement was dropped.

4.4 Restriction (6)

Is the groundedness restriction (6) really necessary? It has certainly been called for in order to avoid so-called infinite keys (see for instance [3]). Naturally, there is no room for infinite keys in an actual database, but recall that we are dealing with the *conceptual* level here. If the state of the world involves infinite keys, then we should be able to say so. The (dirty) tricks we would have to come up with in order to squeeze some finite representation of an infinite whole into a database, is not our concern at this point.

For example: There was a time when the human mind had yet to recognize the Social Security Number as a basic necessity for everyday life. What would be the key constraints for the PERSON entity type when the world described stretches into those dark ages?

Back then, a person was often referred to by his/her ancestry. That is, a person was identified by his/her parents, along with the name. The parents, in turn, were identified in terms of their parents and names, and so on.

Certainly, this model has its flaws when we think of it. Our view of the world involves Big Bangs, Origins of Species and, hence, an initiation of mankind. So there can not be an infinite spiraling structure of human predecessors. However, if we leave such deep reflections out of the picture, our everyday view of ancestry fits this description quite nicely[2]. (See also [11, 9])

[2]For an actual database there would be no need for such keys to be infinite. It would suffice that the chains of predecessors are long enough to distinguish every person from every other. However, we suspect that database designers will typically find the idea of keys with variable and unbounded length unacceptable.

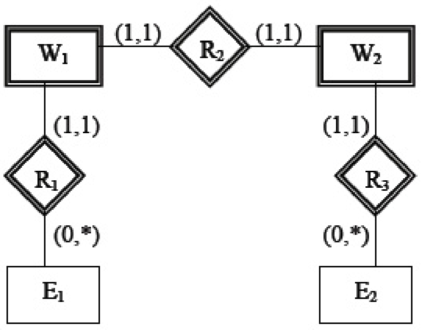
Now it would seem that PERSON could be a weak entity type with a partial key 'Name' and two owners, via the roles 'father' and 'mother', *themselves being* PERSON*s*. So, apparently we can make good use of ownership loops but, unfortunately, they violate the groundedness requirement (6). In view of this, the demand that ownership chains must be grounded may seem too strong and should perhaps be abandoned.

If so, should we be making a great fuzz about least fixpoints? Loops in the diagram can be thought of as giving rise to a system of (recursive) 'equations', which may have several 'solutions'. The least fixpoint, if it exists, would be a distinguished solution singled out to be the chosen meaning of the circular/recursive relationship. (Cf Thalheim [11]) The case against narrowing the interpretation down to fixpoint semantics is that we do not want to harm the expressiveness. After all, if the mini-world we are dealing with does involve several such 'solutions', it should be possible to retain them in the conceptual model. On the other hand, when faced with questions of *identification*, some could prefer to pick a distinguished member from the set of solutions that a recursive relationship might produce. So, here lies a debate but we shall not enter into it.

5 On multiple identifying relationship types

As presented above, a weak entity type could have several identifying relationship types, possibly representing alternative ways of identifying a weak entity. This point is rarely debated.

However, sometimes it is argued that relationship types should be binary. This, it is claimed, is unproblematic since relationship type of higher order can be transformed into a collection of binary ones. Sometimes this transformation introduces weak entity-types having *several* binary identifying relationship types which *collectively* define a unique ownership of the weak entity type [8]. If it is mandatory that all identifying roles are taken to be pulling together to form a single way of identification, the possibility for having alternative keys is lost. This would be an obvious weakness, but even worse is that commonly used notation would be ambiguous. In the diagram shown below



there are two weak entity-types and three identifying relationship types. Hence, one of the weak entity types has two owners. The trouble is, the notation does not reveal which one.

It should be remarked that the notation of Atzeni et al [2] avoids such problems.

6 Conclusion

This has been an attempt to extract the mainstream of entity relationship modelling to form the basis for the formalization of ER-diagrams. The result is an account of ER-syntax with a higher level of detail and rigor than is usually found in the literature. We have discussed the consequences of relaxing some of the requirements involved, and identified possible new features to the ER-model, such as complementing individual entities with *groups* of entities as a basic notion.

References

[1] S. Aronoff, *Geographic information systems: A mangagement perspective*, WDL Publications

[2] P. Atzeni, S. Ceri, S Paraboschi, R. Torlone, *Database systems. Concepts, languages and architectures*, McGraw Hill 1999

[3] M. Balaban, P. Shoval, Resolving the 'weak status' of weak entity types in entity relationship schemas. In Akoka, Bouzeghoub, Comyn-Wattiau, Metais (Eds) *Conceptual Modeling - ER'99: proceedings of the 18th international conference on conceptual modeling*, Lecture Notes in Computer Science 1728, Springer Verlag 1999

[4] M. Balaban, P. Shoval, Enhancing the ER model with integrity methods, *Journal of Database Management*, Vol 10, No 4, 1999.

[5] C. Batini, S. Ceri, S. Navathe, *Conceptual database design. An entity relationship approach*, Benjamin/Cummings 1992

[6] P. Chen, The entity-relationship model – Toward a unified view of data., *ACM transactions on database systems* Vol1, No. 1, 1976

[7] P. Chen P, Database design based on entity and relationship, In Yao (ed), *Principles of database design* Volume I, Prentice Hall 1985

[8] R. Elmasri, S. Navathe, *Fundamentals of database systems*, Addison Wesley 2003

[9] J. Grant, T. W. Ling, M. L. Lee, ERL: Logic for entity-relationship databases,*Journal of intelligent information systems* 2, 1993

[10] G. M. Nijssen, D. J. Duke, S. M. Twine, The entity-relationship data model considered harmful., *Proc. 6th symposium on empirical foundations of information and software sciences*, Atlanta, Ga. 1988

[11] B. Thalheim, *Entity-Relationship Modeling. Foundations of Database Technology*, Springer Verlag 2000