The Relational Model contra Entity Relationship?

H.W.Buff, Swiss Reinsurance Company, 8022 Zürich, SWITZERLAND

Introduction

Entity relationship (ER) languages are usually presented as a means to design conceptual data structures that are independent of the type of the target DBMS (relational, hierarchical or network; see for instance Bat92). Therefore ER claims to be a ("semantic") data model per se.

On the other hand, proponents of the relational model tend to neglect the usefulness of ER (see for instance Dat92).

This note contains a few remarks on how to grasp the best of both worlds.

First: It is both unrealistic and unnecessary to design data structures independent of the type of the target DBMS (unrealistic because of problems of physical implementation, unnecessary because DBMS's in companies usually live longer than conceptual data structures). In this sense it is a good idea to consider ER as a "thin layer on top of the relational model" (Dat92).

But second: If one chooses the right dialect of ER, this thin layer can give very useful help in the following traditional problem domains of the relational model:

- normalisation
- referential integrity
- NULLs
- semantics support by the DBMS.

The Problems

Normalisation

Traditional dependency theory is too difficult, sometimes even for theoreticians (we continue to take notice of articles that describe corrections to published normalisation algorithms). Textbooks for beginners often fail to distinguish sharply between a syntactical side (world of "time independent" functional dependencies and normalisation algorithms) and a semantical side (world of relations which contain dynamically changing data). According to those textbooks an empty relation of three or more attributes would be highly unnormalised (because every functional dependency is valid in such a relation).

In general, only third normal form is achievable, but the target normal form should be Boyce Codd (BCNF), because only BCNF can be supported by the typical relational DBMS.

The designer should not have to occupy himself with these data representation normalisation problems. He seldomly thinks in terms of functional dependencies and does not like "universal relation assumptions" of any kind. A further problem is that the language for expressing functional dependencies does not distinguish different kinds of FD's (for instance we can have an FD between EMPNUM and PROJNUM for "WORKS IN", and another one for "SUPERVISES").

It is a bad idea to apply normalisation algorithms **after** mapping the conceptual design into relation schemes, because it is natural for analysts, programmers and users to think in terms of relations, which should therefore mirror the design. Normalisation should be "information normalisation" and should not be separated from the conceptual design.

Referential integrity

Dependency theory in the presence of inclusion dependencies is even more difficult. It is well known that the question whether a (functional or inclusion) dependency follows from a set of functional and inclusion dependencies is in general undecidable (Var85). Therefore fully automatic tools that properly handle those dependencies cannot be expected. But practical problems arise at an even more elementary level: What to do if a normalisation algorithm demands a foreign key consisting of more than one attribute to be separated?

Almost impossible for the practitioner to handle is the presence of NULLs in foreign keys. A tuple usually is defined as NULL if one of the components is NULL (and the system checks referential integrity for non NULL foreign keys only). Therefore transaction logic can become very complicated (if a user adds the last missing value of a foreign key tupel, he might only today recognize that he made a mistake with the other values a few weeks ago).

Prominent relational DBMS have implemented a "record level semantics" with referential integrity, which is a clear deviation of the philosophy of set processing within relational systems. This means that the developer has to occupy himself with lots of cases that cause difficulties and unexpected results at definition or at run time.

And there is not enough time for every developer to read dozens of pages on referential integrity. All of those problems call for guidelines.

NULLs

The principal problem with NULLs is that everybody (theoretician, DBMS manifacturer and DBMS user) should "think" in threevalued logic, which is very difficult. There are theoretical papers that make propositions for algorithms in the context of NULLs that turn out to be unimplementable because of undecidability problems (Gra77 describes such an algorithm). There are prominent relational systems with semantical faults concerning NULLs. And the user typically looses values and has to write complicated queries in the presence of NULLs.

Therefore NULLs should be avoided (the fact that an information is missing should map to the absence of a tupel in a relation).

Semantics support by the DBMS

Data design should not be separated from operations and constraints design. This can be seen even without ideas of object orientation in mind (with trivial examples like the following: Suppose we have data with periods of validity. If we take attributes "FROMTIME" and "TOTIME" for validity periods, then typically the operations are trivial and the constraints complicated, whereas if we only take "FROMTIME", then operations tend to become complicated and constraints are trivial). Constraints in general include more than only data key and referential integrity conditions, and it is not a trivial task to write them down.

Therefore it is important to have a clear separation of those constraints that can be guaranteed by the DBMS from those that must be guaranteed by the programmer (or by suitable data administration). In other words, a language for conceptual data design should admit only expressibility of constraints that can be supported by the DBMS.

Clear definition of the general case

Another problem encountered with conceptual data design languages (which is not a problem of the relational model per se) is that the semantics definitions often are limited to a few example cases. The same situation can be observed with the mapping of conceptual design constructs into relation schemes.

It is clear that general definitions must be recursive in nature (semantics and mapping), but it is equally clear that these definitions must be very simple (two pages of mathematical definitions are interesting but not practical).

A possible solution

It is not difficult to define a dialect of ER that addresses the abovementioned problems, for instance along the lines of Mar87 (with slightly modified semantics definitions, such that the "canonical mapping" delivers BCNF).

Buf91 presents such a dialect for the practitioner, together with a very easily understandable definition of the notion of "correct ER diagram" (a syntactical notion) and ("canonical") mapping into relation schemes. The mapping delivers BCNF (exactly every box will be a relation). Referential integrity constraints correspond exactly to the arrows (the connections between boxes are always arrows), which means that (for correct diagrams) only undangereous cases of referential integrity will have to be implemented. There is no need for NULLs. They are not completely forbidden, but they do not appear (in keys and) in foreign keys, because there are no binary cardinality constraints. The whole of the semantics expressible in the (correct) diagrams is guaranteed by the (relational) DBMS (if keys and referential integrity constraints are definable in the system). Semantics and mapping are only defined for correct diagrams (this is the reason why the whole thing is both understandable by everyone and mathematically formalisable).

Entity relationship, properly defined, is not a competition for the relational model, but an added help that can simplify matters a lot.

References

Bat92 C.Batini,S.Ceri,S.B.Navathe, Conceptual Database Design, An entity relationship approach, Benjamin Cummings 1992

Buf91 H.W.Buff, Entity relationship for relational database, Swiss Re 1991

Dat92 C.J.Date,H.Darwen, Relational Database, Writings 1989-1991, Addison Wesley 1992

Gra77 J.Grant, Null values in a relational database, Inf. Proc. Letters, vol.6, no.5, Oct.1977

Mar87 V.M.Markowitz, Entity relationship consistency for the relational model of databases, Thesis Technion Haifa Israel 1987

Var85 L.K.Chandra,M.Y.Vardi, The implication problem of functional and inclusion dependencies is undecidable, SIAM J. COMPUT., vol.14, no.3, Aug.1985