What Is An Elementary Fact?

Dr. Terry Halpin
Northface University

This is a slightly edited version of a paper originally published in Proceedings of First NIAM-ISDM Conference*, eds G.M. Nijssen & J. Sharp, Utrecht, (Sep, 1993), and is reprinted here by permission. For historical reasons, the paper has been retained essentially in its original form. However some of the CASE tools mentioned no longer exist (e.g. RIDL* and CD), and InfoDesigner subsequently evolved into InfoModeler, and later VisioModeler, before being replaced by the ORM Source Model solution in Microsoft Visio for Enterprise Architects. Moreover, the term " lazy entity" has been renamed as "independent entity".*

Database schemas are best designed by mapping from a higher level, conceptual schema expressed in human-oriented concepts. While conceptual schemas are often specified using entity-relationship modeling (ER), a more natural and expressive formulation can usually be specified using a version of Object Role modeling (ORM), such as NIAM. This approach views the world in terms of objects playing roles, and traditionally expresses all information in terms of elementary facts, constraints and derivation rules. Although verbalization in terms of elementary facts clearly has many practical and theoretical advantages, it is difficult to define the notion precisely. This paper examines various awkward but practical cases that challenge the traditional definition. In so doing, it aims to clarify what elementary facts are and how they can be best expressed.

Keywords: *conceptual schema, database design, elementary fact, information modeling*

Introduction

It is now widely accepted that logical database schemas (e.g. relational or network schemas) are too far removed from human concepts (such as objects) to facilitate direct modeling of applications. Instead, an application model should be specified at the conceptual level, in formalized natural language, before mapping down to the logical and internal structures supported by the implementation database system (ISO 1982). Existing CASE tools typically allow the modeler to input the main features of a conceptual schema in diagram form, with some finer details specified textually. The conceptual schema is then mapped, with varying degrees of automation and completeness, to internal and perhaps external levels (e.g. see De Troyer 1989, McCormack et al. 1993).

The quality of the database schema thus depends critically on the quality of the original conceptual schema. Linguistic studies have revealed that some things may be easier to describe in one natural language rather than another. Similarly, the quality of a conceptual model is often influenced by the conceptual language (graphic or textual) used for its specification. Most conceptual languages for data modeling are based on a version of Entity-Relationship modeling (ER). Good examples of ER languages can be found in Barker (1990), Czejdo et al. (1990), and Hohenstein & Engels (1991).

However a superior conceptual modeling method is provided by *Object Role Modeling* (ORM), of which NIAM (Natural-language Information Analysis Method) is a classic example (Nijssen & Halpin 1989). Basically, ORM models the world in terms of objects that play roles (e.g. Person jogs, Person authors Book); the attribute construct is not used in the original modeling (but can be in abstractions). Its schema diagrams are closer to natural language, are often more expressive, and their use of role boxes allows them to be populated with fact instances for validation purposes. Various versions include BRM (Binary Relationship Modeling), FORM (Formal Object-Role Modeling), MOON (Normalized Object-Oriented Method), NIAM, NORM (Natural Object Relationship Model) and PSM (Predicator Set Model).

Various graphical and textual languages have been developed to specify, and perhaps manipulate, conceptual models in these versions, often with CASE tool support (e.g. IAST (Control Data 1982) and RIDL* (Intellibase 1990)). More recently, other languages have been proposed. LISA-D (Language for Information Structure and Access Descriptions) is based on PSM (ter Hofstede et al. 1992). Our version, called FORML, (Formal Object Role Modeling Language) is supported in the InfoDesigner workbench from Asymetrix (Halpin & Harding 1993). Other ORM-based CASE tools exist, either as commercial products, such as CD (ITI Brisbane) or academic prototypes such as GISD (Shoval et al. 1988).

While some of these have added support for complex objects and hence compound fact types, essentially they all agree that the information carried by a conceptual model can be expressed in terms of elementary facts, constraints and derivation rules. Although verbalization in terms of elementary facts has many practical and theoretical advantages, it is difficult to define the notion precisely. This paper examines some practical cases that challenge the traditional definition. In so doing, it aims to clarify what elementary facts are and how they can be best expressed.

The next section states some basic properties of elementary facts, motivates their use in conceptual modeling, and argues for mixfix predicates of arbitrary arity. Three problem cases are then considered which require further decisions on what should be allowed as an elementary fact. These involve nesting of predicates with proper subkeys, compositely identified object types with non-key roles, and “lazy” entity types. The conclusion summarizes the main points and indicates research directions.

Elementary facts: what, why and how?

We begin with a working definition that will be qualified in later sections. An *elementary fact* can be roughly defined as an assertion that an object plays a role, e. g.

The Planet named 'Earth' is inhabited.

or that one or more objects participate in a relationship, e.g.

The Planet named 'Earth' is orbited by the Moon named 'Luna'.

where the fact cannot be split into two or more facts without losing information. For example, the following fact 3 is compound rather than elementary, since it is equivalent to the conjunction of fact 1 and fact 2.

(3) The Planet named 'Earth' is inhabited and is orbited by the Moon named 'Luna'.

In Object Role modeling, information examples about the application are verbalized in terms of elementary facts, and these instances are abstracted to elementary fact types (e.g. Planet is inhabited; Planet is orbited by Moon). To complete the conceptual schema, we add constraints (e.g. **each** Moon orbits **at most one** Planet) and possibly derivation rules. Expressing information as elementary facts is not always easy. So why go to this trouble? Some of the main reasons are:

- By dealing with information in simple units we stand a better chance of getting a correct picture of the application being modeled;
- Constraints are easier to express and check (e.g. all functional dependencies should appear as uniqueness constraints; and because fact types are shorter, the number of possible constraint patterns in each is reduced);
- The conceptual schema is easier to modify, since fact types can be added or deleted one at a time, rather than modifying compound fact types.
- The same conceptual schema can be used to map to different data models (if we group fact types together into compound fact types on the conceptual schema, different groupings may actually be required in some target data models).

The various ORM languages differ somewhat in the way they express schemas and elementary facts. Ideally, a conceptual language should be formal (so the system can process it), expressive (so we can convey whatever we want) and natural (so people can readily understand it). In Halpin (1989), ORM was formalized in terms of first order logic, using the language KL (Knowledge Language). With respect to static constraints, KL satisfies the criteria of formality and expressive power; but it is too symbolic for the average modeler. FORML was designed as a "sugared" version of KL, allowing high level, natural verbalization, and is supported in both textual and graphical forms in InfoDesigner. Here are a few sample facts expressed in FORML:

(4) The Planet named 'Earth' is inhabited.

(5) The Planet named 'Mars' is orbited by / orbits the Moon named 'Phobos'.

(6) The Lecturer named 'Halpin TA' visited the Country named 'USA' in the Year 1992 AD.

Object type names begin with a capital letter. Here we have highlighted the predicates in bold. A predicate is a sentence with holes in it for object-terms. Here the predicates are: "... is inhabited"; "... is orbited by ..."; and "... visited... in". Sentences (4), (5) and (6) respectively express unary, binary and ternary facts. Predicates of any arity are permitted. In general, an elementary fact is expressed as a sentence of the form $Ro_1,..o_n$, where R is a predicate of arity n, and $o_1..o_n$, are n object terms. Note that mixfix (or

distfix) predicates are used– the holes for the object terms may appear anywhere in the predicate.

For binary cases, FORML allows the *inverse predicate* to be stated as well (preceded by "/"). In (5), "orbits" is the inverse of "is orbited by". This is useful mainly for expressing constraints. Some versions of ORM and ER require all relationships to be binary. So long as nesting is permitted, this does not result in any loss of expressive power; however it often prevents modelers from saying things in the way that appears most natural to them. For example, compare (4) with:

(4') The Planet named 'Earth' has HabitedStatus with code 'I'.

and (6) with a nested formulation such as:

(6') The Lecturer named 'Halpin TA' visited the Country named 'USA' :: alias Visit.

The Visit ⟨'Halpin TA', 'USA'⟩ occurred in the Year 1992 AD.

Moreover, since (6) does not have just one uniqueness constraint spanning just two of its roles, it seems arbitrary which two objects are paired first in the nesting (why not pair Lecturer with Year first, or even Country with Year?).

Another problem with nesting is that the graphical display takes up more room and is harder to understand when constraints exist between the inner and outer roles of the objectified predicate. For example, consider the fact type *Person is placed in Subject at Position*, where a uniqueness constraint spans the first two roles and another uniqueness constraint spans the last two roles. The flattened version is much easier to follow than the nested version. Diagrams for this situation are depicted in Nijssen and Halpin (1989, pp. 87-8).

Thus, although the binary-only approach leads to a simpler metamodel and fewer language primitives, it can actually prevent the user from expressing information in a simpler and more natural way. For these reasons, we recommend that fact types of any arity be supported.

In FORML, object terms may be simple or composite (e.g. a city might be identified by the combination of having a name and being in a country). Reference schemes may be declared separately up front; then only the object types and values are required for reference. Articles such as "the" are optional. For example, the fact that the employee with employeenr 37 works in the department with name 'Sales' may be specified thus:

Reference schemes: Employee (employeenr);

Department (name)

Fact: Employee 37 **works in** Department 'Sales'.

By supporting ordered, mixfix predicates of any arity, FORML enables the logical deep structure to be expressed in a surface structure in harmony with the ordered, mixfix nature of natural language, independent of the natural language used to express the facts. For example, the previous fact may be expressed in Japanese thus:

Reference schemes: Jugyo in (jugyo in bango);

Ka (namae)

Fact: Jugyo in 37 **wa** Ka 'Eigyo' **ni shozoku suru.**

The infix predicate "... works in ..." corresponds to the mixfix predicate "... wa ... ni shozoku suru". For a detailed discussion of this example, as well as constraint expression in FORML, see Halpin & Harding (1993).

Although we prefer to use ordered, mixfix predicates for naturalness, another approach is to treat a fact as a named set of (object, role) pairs: $F\{(o_1, r_1),\ldots ,(o_n,r_n)\}$. Here each object o_i is paired with the role r_i that it plays in the fact F. For example, "The Person with surname 'Wirth' designed the Language with name 'Pascal'" might be specified as: Design{(The Person with surname 'Wirth', agentive), (The Language with name 'Pascal', objective)}.

Instead of the case-adjectives "agentive" and "objective", other role names could be used (e.g. "designer" and "language", or "designing" and "being designed by"). By pairing objects with their roles, the order in which the pairs are listed is irrelevant. This approach is used in MIML, RIDL and LISA-D. The use of gerundives for role names often makes conceptual queries sound natural (e.g. LIST Employee working-in Department 'Sales'). However the expression of fact types and facts themselves is obviously less natural. Moreover, the technique is suited only to binary cases, and not all natural languages support gerundives.

Each approach has its own advantages and disadvantages. We have opted for naturalness of expression, to simplify the verbalization phase of the modeling process. However, as we will see in the next section, what seems natural is not always desirable.

Nesting of predicates with proper subkeys

The earlier definition of elementary facts leaves it to the modeler to decide whether or not a fact can be split without information loss. Some linguistic guidance can be given. For example, the presence of "and" in a sentence suggests that it might be compound (e.g. sentence (3)). However this is neither necessary, e.g. (3) might be worded:

(3') The inhabited Planet named 'Earth' is orbited by the Moon named 'Luna'.

nor sufficient, e.g. the following sentence is elementary:

(7) The Student with studentnr 12345 enrolled in the Subject with code 'CS114' and obtained a Rating of 7 for it.

If a population is significant with respect to elementarity, then elementarity can be determined by looking for spurious tuples when the fact is split and then joined (Nijssen & Halpin 1989 §5.3). But in practice this check is almost worthless, since one rarely has a significant population, and to know that it is significant begs the question.

The only formal way to check that a predicate is elementary is to make use of known constraints. A sufficient but not necessary condition for splittability is the presence of at least two roles in the predicate that are outside the predicate key(s). For example, consider fact (8):

(8) The Moon 'Deimos' orbits the Planet 'Mars' in the Period 30 days.

Suppose we schematize this as the ternary shown in Figure 1. For the reader unfamiliar with the conceptual schema diagram notation, we note some basic conventions. Object types are shown as named ellipses, and predicates are named beside their first role. An arrowed bar over a role or role sequence indicates an internal uniqueness constraint, and a circled "u" denotes an external uniqueness constraint. Roles that are mandatory for their fact type population are marked by a dot "•" at the end of their role connector. With this example, all entity types have simple primary reference schemes, shown in parenthesis.

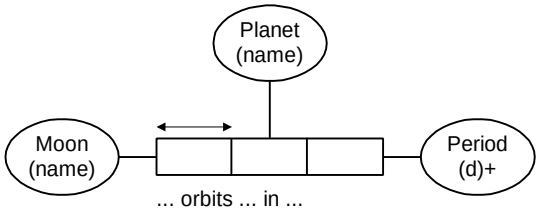

Figure 1: This fact type is not elementary

Because each moon orbits only one planet, the role played by Moon in Figure 1 has a simple uniqueness constraint, and hence functionally determines the other roles. Hence the ternary splits on the first role into two binaries. So fact (8) is compound, being equivalent to the conjunction of (9) and (10).

(9) The Moon 'Deimos' orbits the Planet 'Mars'.

The Moon 'Deimos' orbits in the Period 30 days.

Now suppose that fact (8) was verbalized in a nested way, e.g.

(8') The Moon 'Deimos' orbits the Planet 'Mars'.

This orbit has a Period of 30 days.

This leads to the nested fact type shown in Figure 2. Because a moon orbits at most one planet, the objectified predicate has a uniqueness constraint over only one of its roles. So the orbital period depends only on the moon, not the planet. Hence the nesting can be split into the same two binaries as before. Indeed, the nested version of the splittability check is that all roles of an objectified predicate be spanned by the same uniqueness constraint.

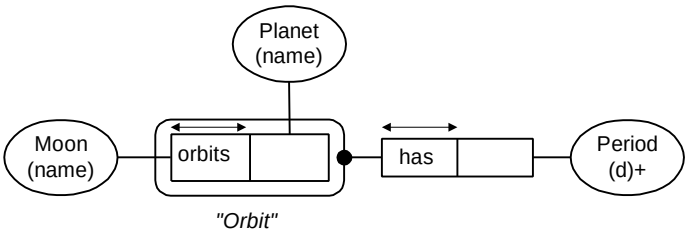

Figure 2: The outer fact type is not elementary

Now this is old news (Nijssen & Halpin 1989, p. 100). So why bring it up again? Well, one can run into cases where it seems quite unnatural not to nest them, even though they are compound according to our current definition. The most extreme cases of this involve 1:1 predicates. For example, suppose we are maintaining a database about the current marriage of famous couples. Consider the following information (the value for marriage year is just a guess):

(11) The Person 'Bill Clinton' is husband of / is wife of the Person 'Hillary Clinton'.

(12) This marriage occurred in the Year 1970.

To avoid problems with symmetric predicates we have chosen "is husband of" instead of "is married to". We have also included the inverse predicate "is wife of". If we schematize this as a nested fact type, we obtain the schema shown in Figure 3.

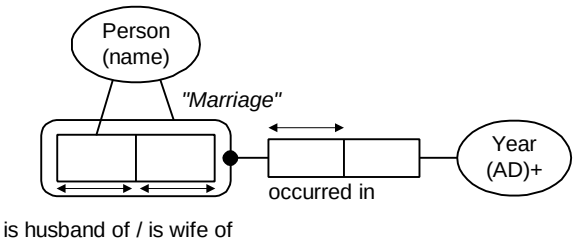

Figure 3: Should this be allowed?

Because of our splittability rule that an objectified predicate must have a single uniqueness constraint spanning all its roles, the schema in Figure 3 must be split into two binaries. Fact 11 can remain as it is, but what about fact 12? Do we record the marriage year for the husband or for the wife? If we do this for the husband, we must create a subtype for Husband, and replace (12) by:

(12') The Husband 'Bill Clinton' was married in the Year 1970.

Or should we create a subtype for Wife, and replace (12) by:

(12") The Wife 'Hillary Clinton' was married in the Year 1970.

The reader is invited to draw the schema diagrams for each choice. Whichever choice we take, we may be accused of sexist bias! Moreover, it seems quite unnatural to be forced to make this kind of choice at the conceptual level.

Perhaps then, in rare cases like this we should relax the nested version of the splittability rule to allow nesting of 1:1 cases where the modeler feels it is unnatural to make a splitting choice. What do you think? Note however, that when the conceptual schema is mapped down to a logical schema some choice may have to be made as to how to store the information. For example, if no other functional roles (i.e. roles with a simple uniqueness constraint) are played by Person, we might map the nested conceptual schema to the relational schema:

Marriage (husband, wife, marriageyear)

or to:

Marriage (husband, wife, marriageyear)

where the double underline indicates choice of primary key. Apart from our present discussion, the mapping of 1:1 predicates can in general require careful thought. For a detailed discussion of 1:1 mapping alternatives, see Ritson & Halpin (1993).

It should be pointed out that nested 1:1 cases can often be split naturally. For example, consider a database about current professors in a University. Each professor holds exactly one chair, and vice versa. How should we represent the information in sentence (13)?

(13) Professor 'Maria Orlowska' was appointed to the Chair 'Information systems' in Year 1990.

One might consider nesting this as: (Professor holds Chair) was appointed in Year. But (13) splits naturally to:

(14) Professor 'Maria Orlowska' holds the Chair 'Information systems'.

(15) Professor 'Maria Orlowska' was appointed in Year 1990.

In such cases we feel that the two binaries are preferred to the nested solution.

What about *n*:l or 1:*n* binaries? Should we ever allow these to be nested? Some versions of ORM do (e.g. PSM), and those ER versions that support nesting at all, typically do allow such cases. When justification is given for allowing such compound fact types in a conceptual schema it usually amounts to a claim that this is the "natural" way to do things. In general we feel it is very dangerous to allow such cases. We find beginning students tend to do this in their early efforts, and in almost all cases it is just bad modeling. Having a rule to stop nesting of *n*:l and 1:*n* cases at least prevents a lot of students from doing some very silly things.

Moreover, if we allow this, the mapping algorithms become more complicated. For example, consider the schema in Figure 4 (reference schemes omitted for simplicity).

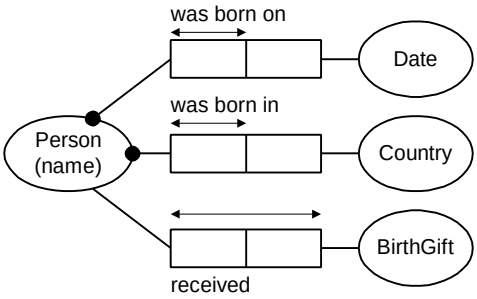

Figure 4

Suppose that instead of restricting ourselves to elementary fact types, we nest this as shown in Figure 5. One often finds this kind of modeling in ER books. For example, Batini et al. (1992, p. 34) model birthdate as an attribute of (Person is born in City). We feel this is unfortunate. To begin with, why bring in a place to talk about a date or gift? Secondly, the mapping is now complicated. Before passing the schema to the normal relational mapping algorithm, it will have to be pre-processed to the elementary version shown above; otherwise country will appear in the table for persons and their gifts. Unlike the 1:1 case, this pre-processing can be done automatically, since only one choice is possible. Note that if a role played by an objectified functional predicate is optional, it may be necessary to include in the pre-processing an appropriate subset or subtype constraint.

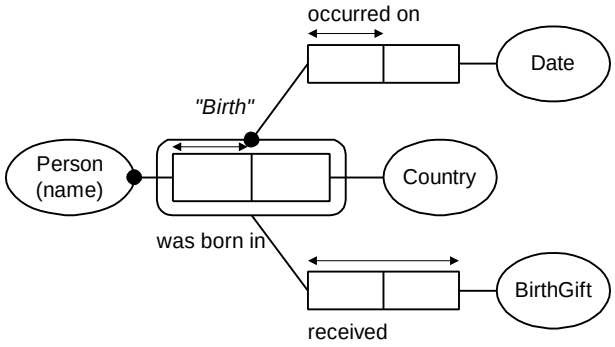

Figure 5: A bad way to model the UoD of Figure 4

Even if we arrange for pre-processing, the nested version of the conceptual schema is often undesirable since it is often harder for the application developer to relate the logical schema back to the conceptual schema. Another disadvantage of nesting predicates with proper subkeys is that conceptual schema transformations (e.g. see Halpin 1992) now become more complicated. Preprocessing to the elementary version should normally be undertaken before performing the transformations.

With all these disadvantages, should we ever allow nesting of 1:*n or n*:1 binaries? For the novice modeler, the answer appears to be "No". However, one may occasionally run into a case where it seems much more natural to allow nesting– we have encountered

such cases in meta-modeling. We recommend that an expert modeler should be given the option of over-riding the default restriction on nesting, provided pre-processing to the elementary version is guaranteed before mapping or transforming the schema.

Compositely defined object types with nonkey roles

A compidot (compositely identified object type) is either an objectified predicate or an entity type with an explicit, composite reference scheme. When the only fact role attached to a compidot is not part of a key, the notion of splittability needs further clarification. For example, consider the schema in Figure 6.

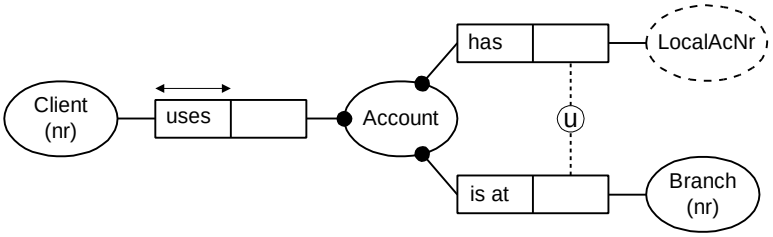

Figure 6: Is this fact type elementary?

Here we have one fact type: Client uses Account. The entity type Account has a composite reference scheme. In principle, one might consider replacing this schema with two fact types: Client has account with LocalAcNr; Client has account at Branch. Is this a case of splitting a compound fact type into two elementary fact types? We answer "No" to this question. We give the same answer if Account were modeled as an objectified predicate (though we feel this is less natural). To allow free use of compidots by the modeler, we refined the notion of Elementarity to require that splitting must preserve object types. Since the object type Account would be lost in the replacement entertained earlier, this does not count as splitting. The fact type in the original schema is elementary, not compound. For further discussion on this issue, see Halpin & Orlowska (1991).

Lazy entities

In traditional NIAM, each entity in a conceptual model must play a role in some fact. We felt this rule was too restrictive, as in many practical applications one needs to assert the existence of an entity before one knows anything else about it. In our formalization (Halpin 1989) we decided to allow *lazy entities* (entities that exist but need not participate in any facts). For example, suppose we wish to maintain a database about the current countries and, if they competed in the last Olympics, how many gold medals they won. This could be schematized as shown in Figure 7 (for simplicity, reference schemes are omitted). The data here are fictitious. Here Country is a lazy entity type, as shown by the " !" appended to its name.

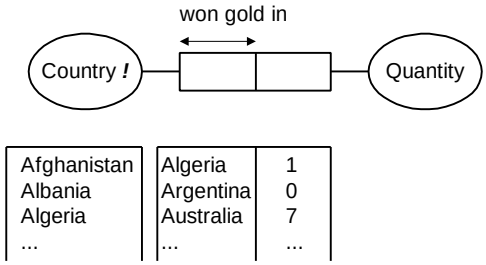

Figure 7: Country is a "lazy" entity type

A reference table for Country lists all the countries. For example, the first row says that a country named "Afghanistan" exists. Rather than introduce another kind of fact (e.g. existential fact as opposed to elementary fact), we treat this as a reference rather than a fact. Because the reference list is large and unstable, it is impractical to include it as a value constraint (i.e. a listing of allowed values which may reference objects of that type).

Lazy object types require special treatment in mapping. The sample schema maps to a table with two colunms, one for the country name, and for the quantity (optional). If a lazy object type has no functional fact roles, its references map to a single column table. For example, suppose the only fact type for Country is: Country borders Country. Moreover suppose we wish to know which countries have no bordering countries (e.g. consider Australia). This maps to two tables: one table to list all the countries, and one table to list the border facts. See Ritson & Halpin (1992) for more details.

Conclusion

This paper examined the notion of elementary fact type, as used in Object Role modeling. After providing some motivation and explanation for this notion, we suggested that elementary facts are most naturally expressed by using mixfix predicates of arbitrary arity, and structured object terms. This approach has been adopted in the FORML language supported by InfoDesigner. Although the definition of an elementary fact precludes the use of objectified predicates without spanning uniqueness constraints, in rare cases this restriction may obviate natural modeling-in such cases, the restriction might be relaxed for expert modelers provided preprocessing to the elementary version is ensured before mapping or transforming the schema. To cater for compositely defined object types, the notion of fact splitting was strengthened to require object type preservation. Finally the notion of lazy entities was briefly discussed.

Limited support for complex objects has recently been added to the FORM method (as well as other versions such as NORM and PSM). Since these effectively allow compound facts in the conceptual model, considerable care is needed to avoid their abuse. Current research efforts include the specification of a design discipline to control their use, and the specification and realization of associated mapping procedures.

References and bibliography

1. Barker, R. 1990, *CASE* Method: Entity Relationship Modelling,* Addison-Wesley, Wokingham, England.

2. Batini, C., Ceri, S. & Navathe, S.B. 1992, *Conceptual Database Design: an entity-relationship approach*, Benjamin / Cummings, Redwood City CA.

3. Control Data 1982, *IAST.- Information Analysis Support Tools,* Reference Manual (Control Data Publication no. 60484610).

4. Czejdo, B., Elmasri, R., Rusinkiewicz, M. & Embley, D.W. 1990, 'A graphical data manipulation language for an extended entity-relationship model', *IEEE Computer,* March 1990, pp. 26-37.

5. De Troyer, 0. 1989, 'RIDL*: A tool for the computer-assisted engineering of large databases in the presence of integrity constraints', *Proc. ACMSIGMOD Int. Conf. on Management of Data*, Oregon.

6. De Troyer, 0. 1991, 'The 00-Binary Relationship Model: a truly object-oriented conceptual model', *Advanced Information Systems Engineering: Proc. CAiSE-91,* Springer Lecture Notes in Computer Science, no. 498, Trondheim.

7. Halpin, T.A. 1989, 'A logical analysis of information systems: static aspects of the data-oriented perspective', doctoral thesis, University of Queensland.

8. Halpin, T.A. 1991, 'A fact-oriented approach to schema transformation', *Proc. MFDBS-91,* Springer Lecture Notes in Computer Science, no. 495, Rostock.

9. Halpin, T.A. 1992, 'Fact-oriented schema optimization', *Proc. CISMOD-92*, Bangalore, India, July 1992.

10. Halpin, T.A. 1993, 'Object-Oriented Databases: is this the future?', *Proc. 1993 IPT Conf.*, QSITE, Brisbane.

11. Halpin, T.A. & Harding, J. 1993, 'Automated support for verbalization of conceptual schemas', *Proc. 4th European Workshop on Next Generation CASE Tools,* Paris (available as Tech. Report from Comp. Sc.Dept Univ. Twente).

12. Halpin, T.A. & McCormack, J. 1992, 'Automated validation of conceptual schema constraints', Advanced Inf. Systems Engineering: *Proc. CAiSE'92,* ed. P. Loucopoulos, Springer Lecture Notes in Computer Science, no. 593, pp. 445-62.

13. Halpin, T.A. & Oei, J.L.H. 1992, 'A framework for comparing conceptual modelling languages', *Tech. Report 92-29,* Dept. of Informatics, Uni. of Nijmegen, Nov. 92.

14. Halpin, T.A. & Orlowska, M.E. 1991, 'Fact-oriented modelling for data analysis', *Journal of Information Systems,* vol. 2, no. 2, Blackwell Scientific, Oxford.

15. Halpin, T.A. & Ritson, P.R. 1992, 'Fact-oriented modelling and null values', *Research and Practical Issues in Databases: Proc. 3rd Australian Database Conf.*, eds B. Srinivasan & J. Zeleznikov, World Scientific, Singapore.

16. ter Hofstede, A.H.M., Proper, H.A. & van der Weide, Th.P. 1992, 'Formal definition of a conceptual language for the description and manipulation of information models', *Tech. report 92-16*, Dept of Informatics, Uni. of Nijmegen.

17. Hohenstein, U. & Engels, G. 1991, 'Formal semantics of an entity-relationship-based query language', *Entity-Relationship Approach: the core of conceptual modelling (Proc. 9th ER conf.)*, ed. H. Kangassalo, Elsevier Science Pub., Amsterdam.

18. Intellibase, 1990, *RIDL-M User's Guide*, Intellibase N.V., Belgium.

19. ISO 1982, *Concepts and Terminology for the Conceptual Schema and the Information Base*, ed. J.J. van Griethuysen, ISO TC97/SC5/WG3, Eindhoven.

20. McCormack, J.1., Halpin, T.A. & Ritson, P.R. 1993, 'Automated mapping of conceptual schemas to relational schemas', *Advanced Info. SystemsEngineering: Proc. CAiSE-93., eds* C. Rolland, F. Bodart & C. Cauvet, Springer LNCS vol. 685, pp. 432-48, Paris (June).

21. Nijssen, G.M. & Halpin, T.A. 1989, *Conceptual Schema and Relational Database Design*, Prentice Hall, Sydney.

22. Olle, T.W., Hagelstein, J., Macdonald, I.G., Rolland, C., Sol, H.G., Van Assche, F.J.M. & Verrijn-Stuart, A.A. 1991, *Information Systems Methodologies - A Framework for Understanding, 2nd edn.*, Addison-Wesley, Wokingham, England.

23. Ritson, P.R. & Halpin, T.A. 1992, 'Mapping conceptual constraints to a relational schema', *Tech. Report 223*, Dept of Computer Science, University of Queensland.

24. Ritson, P.R. & Halpin, T.A. 1993, 'Mapping one-to-one predicates to a relational schema', *Advances in Database Research: Proc. 4th Australian Database Conf.*, eds M.E. Orlowska & M. Papazoglou, World Scientific, Singapore, pp. 68-84.

25. Shoval, P., Gudes, E. & Goldstein, M. 1988, 'GISD: a graphical interactive system for conceptual database design', *Infonn. Systems*, vol. 13, no. 1, pp. 81-95.