

Towards Active Conceptual Modelling for Sudden Events

John F. Roddick, Aaron Ceglar and Denise de Vries

School of Computer Science, Engineering and Mathematics
Flinders University,
PO Box 2100, Adelaide, South Australia 5001,
Email: roddick@csem.flinders.edu.au

Abstract

There are a number of issues for information systems which are required to collect data urgently that are not well accommodated by current conceptual modelling methodologies and as a result the modelling step (and the use of databases) is often omitted. Such issues include the fact that

- the number of instances for each entity are relatively low resulting in data definition taking a disproportionate amount of effort,
- the storage of data and the retrieval of information must take priority over the full definition of a schema describing that data,
- they undergo regular structural change and are thus subject to information loss as a result of changes to the schema's information capacity,
- finally, the structure of the information is likely to be only partially known or for which there are multiple, perhaps contradictory, competing hypotheses as to the underlying structure.

This paper presents the *Low Instance-to-Entity Ratio* (LItER) Model, which attempts to circumvent some of the problems encountered by these types of application and to provide a platform and modelling technique to handle rapidly occurring phenomena. The two-part LItER modelling process possesses an overarching architecture which provides hypothesis, knowledge base and ontology support together with a common conceptual schema. This allows data to be stored immediately and for a more refined conceptual schema to be developed later. LItER modelling also aims to facilitate later translation to EER, ORM and UML models and the use of (a form of) SQL. Moreover, an additional benefit of the model is that it provides a partial solution to a number of outstanding issues in current conceptual modelling systems.

Keywords: LItER Modelling, Rapid Conceptual Modelling.

1 Introduction

Some sudden events require systems capable of rapidly tracking and explaining the phenomenon for a number of reasons:

1. To eliminate, or at least limit, any immediate damage caused by the event,
2. To explain how an event occurred or was allowed to occur, including accommodating alternative hypotheses as to why the event happened,
3. To assist in any subsequent investigations, including the generation of inferences regarding the people who may have been involved,
4. To expose weaknesses in measures designed to prevent such events,
5. To prevent such events happening again.

The role of information systems in such scenarios can vary widely but given that such events are largely unexpected, the rapid development of information systems capable of answering questions is clearly important. Unfortunately, current conceptual modelling techniques are not capable of handling some of the vagaries of such systems (Chen 2006).

In the case of the September 11 attacks, there is still some debate as to whether the US intelligence community did or did not possess the data necessary to infer in advance that the attacks would occur (Lefebvre 2004, Popp et al. 2004). However, whatever the truth, it became necessary to rapidly construct a database to assist afterwards. This database needed to include data from a number of diverse and up to that point, unlinked systems.

Since then there have been a number of attempts to build large scale intelligence systems including the now defunct Terrorism Information Awareness (TIA) project, the abandoned Computer-Assisted Passenger Prescreening System II (CAPPS II), and the Multistate Anti-Terrorism Information Exchange (MATRIX) pilot project (Seifert 2004). Each of these aimed to prevent expected scenarios and thus, in theory at least, the necessary data structures were known in advance.

In the case of unexpected events, the data necessary to assist fall into two categories:

- **Context-specific data** that could not reasonably have been foreseen,
- **Referential or global data** (such as ontologies, classifications, taxonomies, etc) that can be compiled in advance and used as needed.

What is required therefore is a conceptual structure within which contextual data can be loaded and waiting but which also accommodates, rapidly, any other data that might be deemed necessary.

In this paper we outline a new approach – LItER modelling – that lends itself to such systems¹. Both

¹The LItER model, including the example in Section 2.2, was first discussed by the authors elsewhere (Roddick et al. 2007).

context-specific and referential data can be accommodated and, since the model possesses a common schema, the data can be recorded in a (temporal) database environment with all of the advantages that such databases offer, including security, auditing and decision justification. (In the case of decision justification, it may be important that the exact details of the knowledge available at the time of the decision are noted in case they are contradicted by subsequent data).

If required, although with some caveats, the eventual model can be translated to a conventional model (such as EER) once the aspects unable to be accommodated by conventional models have dissipated.

In Section 2 we discuss a number of conceptual modelling issues and issues related to the late binding of the conceptual model over the database data is discussed in Section 3. The LtER model itself is introduced in Section 4. This is followed by some discussion of the issues and an outline of future work.

2 Rapid Conceptual Modelling

2.1 Systems Issues in Conceptual Modelling

Conceptual schema development is an important aspect of an information system's design. In this phase the structure of the system in terms of the relationships between objects, their attributes and constraints are established to the agreement of user and designer. Modelling techniques, such as ER/EER (Chen 1976, Thalheim 2000), NIAM/ORM (Halpin 1998, Verheijen & van Bekkum 1982) and UML (Booch et al. 2005), have been developed, extended and deployed effectively for this purpose over a number of years. Such techniques fit well into the common software engineering frameworks that establish a firm design for the database before any data is stored (Sommerville 2006).

As discussed in other work (Roddick et al. 2007) some classes of information system, however, are not well served by these common techniques. These include:

- systems in which the immediate storage of data is the priority with the organisation of that data (that is, the development of a conceptual model) a secondary issue. For some systems, a mechanism to collect and store data is required more rapidly than the database design phase will permit. This includes systems designed rapidly in response to an immediate need (Chen 2006).
- systems for which the number of entity-types is large in comparison with the number of instances stored. For these systems, the overhead of conceptual modelling can be high and can lead to short-cuts such as the aggregation of inappropriate entity-types.
- systems that undergo substantial structural change. While schema conversion, even those in which the schema's information capacity changes, does not always result in a loss of information², systems that regularly undergo change commonly lose information. Such systems include those used for hypothesis creation such as scientific databases (Shoshani & Wong 1985), criminological systems (Chen et al. 2003) and *ad hoc* models established to track evolving phenomena.

²As discussed in other work (Roddick & de Vries 2006) the limits for *practical* schema versioning in a database \mathcal{D} are that $S_1 \stackrel{p}{\equiv} S_2$ iff $I'(\mathcal{D}|S_1) \rightarrow I'(\mathcal{D}|S_2)$ is bijective where $I'(\mathcal{D}|S_n)$ is the set of all instances of S_n inferable from \mathcal{D} given the constraints of S_n .

- situations where the structure of the information is only partially known or where there are multiple, sometimes competing (although equally valid) models of the same data. While XML can handle semi-structured schema in which instances may possess varying structure, the overall schema is still largely formalised. However, we deal here with systems where the existence of different entity-types and the attributes they possess are largely unknown or where there is no agreement on the structure. Such systems include those that aim to handle empirical evidence in which the overall structure may be changed as ideas are developed and the evidence may still be in the process of being discovered. For these sorts of system, in any conflict between data and schema, it cannot be assumed that the schema is correct and that the data is therefore wrong. Sometimes it is the schema which is the component requiring change.

All of these aspects are exhibited, to a greater or lesser extent, by systems set up to handle sudden events and/or rapidly changing systems. Thus, while, with effort, the traditional forms of conceptual modelling can handle these types of system, the overhead and side effects of doing so are often excessively high. In practice, the conceptual modelling step (and as a consequence the use of a DBMS and the valuable facilities that it can provide) is often side-stepped because of the overhead involved.

2.2 Data Issues in Conceptual Modelling

As well as the classes of system outlined in Section 2.1, there are other problems common in the modelling and implementation of even conventional systems. Some of these are also apparent in systems set up quickly as some occur as a result of ambiguity in the object world. Others, such as the example below, are simply situations not well handled by conventional modelling techniques.

In order to provide some illustration of this, we provide here a motivating example based around the part-subpart and the supplier-part-project problems.

The ABC Company manufactures three types of widget - widayes, which are always blue regardless of who makes them, widebes, which the ABC Company paints blue, and wideas which are by default black but which can be painted according to the project on which they are used. The XYZ Company manufactures widayes, green widebes and has recently made a test batch of red widebes that are as yet unused. widayes are not only sold by themselves but are also used to make wideas. widayes are also known as ayewids.

Some of the problems illustrated by this example and that currently cause some concern include the following:

- Collections of objects must sometimes be treated in the same manner as the objects themselves, often transitively, sometimes recursively. For example, if a batch of widayes are found to be defective then there may also be some wideas that also need to be recalled. This is particularly the case where groups are referred to in place of individuals (either through metonyms, holonyms or hypernyms).
- Attribute values are often provided to the system in ways that are not directly comparable despite conforming to the domain's type definition.

For example, *widayes* may be described as *blue*, *dark blue*, *x3333cc*, *royal blue*, *PMS286* and so on. Synonyms, such as *widayes* and *ayewids* in the example, despite being relatively common, are not well accommodated. While data coercion is sometimes possible, this is not always a solution as the provenance and integrity of the original data may need to be maintained.

- In many conventional modelling techniques (such as EER) a relation formed from an n -ary or binary many-to-many relationship must, for reasons of entity integrity, have a stored instance for all its associated entities. In some cases this is either not possible or not desirable and the common practice is either to deform the model to suit a small fraction of the instances or, more commonly, to create *dummy* or default codes to circumvent this constraint. For instance, in our example, since *widayes* only come in blue there may be little need to record the supplier or the project but for *widbees* and *widseas* the project must be recorded if the colour is needed. Moreover, what about the sample batch of red *widdees*? How is the colour to be stored? Classical modelling would require colour to be recorded in two (or more) places in the model, for example, in a schema such as that depicted in Figure 1.

3 Late Binding the Conceptual Modelling

The storage of data followed by conceptual model creation requires a different position to be adopted in that a generic or common conceptual model must exist for the initial data storage in the absence of the more specialised model. However, having established a common conceptual model, specialisations to that model can be developed incrementally through the testing and imposition of constraints.

For example, consider a scenario in which a University’s student and faculty data is stored in a common storage structure (ignore for the moment the details of that storage structure). In the absence of any specialised schema, reference to entities and attributes must be phrased in terms of the structures provided by the common data model.

Over time, constraints could be tested and added providing more specialisation and eventually providing a level of structure consistent with a conventional conceptual schema. For example, we might test through the discovery of induced dependencies (Roddick et al. 1996), whether a doctoral student has exactly one supervisor and if so, and if such a constraint is considered sensible, it could be added. Labels could also be added so that conventional query languages such as SQL can function.

There are however, a few additional advantages to this approach:

- multiple, perhaps conflicting, structures can be

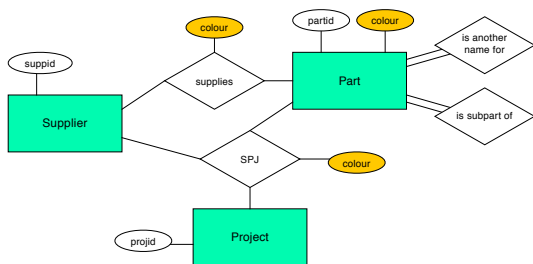


Figure 1: Example Schema for Parts Example

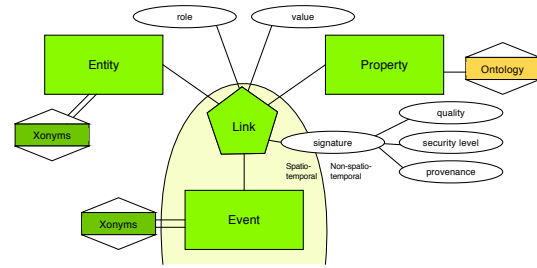


Figure 2: LitER Schema

held allowing, for example, transition between structures or the development of hierarchies of schema.

- multiple modelling paradigms can be used. For example, an EER model can be superimposed to provide a schema view while graph-oriented structures could be tested between data elements.
- having a common conceptual schema provides the ability to construct general utilities as well as facilitating schema integration.
- a common conceptual model lends itself to data mining as an *a priori* defined structure will not mask hidden associations.

The issue is to define a common schema which is flexible enough to hold all data but simple enough not to incur the overhead experienced when creating a full conceptual model. We argue that the proposed LitER modelling method provides a common structural model capable of accommodating data derived from a variety of situations but which remains sufficiently structured to retain the semantics of the data.

4 LitER Modelling

The LitER approach has been developed as a consequence of a practical industry need to generate systems rapidly where the nature of the system is not known in advance but where some preparation (for example, the collection of mesodata and/or ontologies) can be undertaken. Such areas include national security, natural disaster and large-scale incidents where the details are unlikely to be known (at least not in detail) in advance. The LitER approach is to use a common schema discussed in Section 4.1 and shown in Figure 2, embedded within an overarching architecture discussed in Section 4.2 and shown in Figure 3.

4.1 The LitER Schema

The schema consists of three primary meta-entities, Entities, Properties and Events, together with a ternary Link (a form of polymorphic (overloaded) relationship-type). Events and Links are temporally referenced. Specifically, the model consists of the following components:

Entities: These represent objects in the model. This includes not only elementary objects such as those that might be represented by strong and weak entities in an EER Model but also components and aggregations of entities. If data is obtained from multiple sources, the same entity may also be recorded more than once (linked by a synonym link). The only attribute directly recorded is the entity’s identifier (which might

be user or system supplied); all other attributes being recorded through reference to a property.

Properties: These allow the description of properties that can be associated with either an entity or an event. A property may be associated with an ontology which can be used by the constraint manager/hypothesis checker or by the query language as appropriate.

Events: These allow the recording of spatio-temporally referenced happenings. Once again, the only attribute directly recorded is the event's identifier.

Links: The link allows entities, properties and events to be combined. In LitER, we use an overloading form of polymorphism to specify that a link can occur between all or any of Entities, Events or Properties. Moreover, more than one of each can participate in a link with the sole requirement being that a link must include at least two identifying instances. Thus a link can allow:

- one or more entities to be associated with a describing property, optionally with a *value* of that property. Note that the value may also be provided as a simple formula which may include a variable (such as > 25 or $P(\text{Age}).E(\text{Christopher}) + 2$). This allows facts such as *... is over 25* or *... is 2 years older than Christopher* to be recorded³.
- one or more events to be associated with a describing property, optionally with a *value* for that property.
- a link between an entity (or entities) and an event such that an entity's *role* in the event can be recorded.
- a relationship between two entities.

Links have a number of optional predefined attributes as follows:

Value - providing a qualification for the relationship being described. For example, Entity Mary linked to Property Nationality might have the value Australian.

Role - providing a qualification for the relationship being described. For example, Entity Luke linked to Event Phone.Call might have the role Caller.

Quality - provides a measure of confidence (such as a probability) to the link;

Security level - provides a mechanism for restricting access to data;

Provenance - provides a mechanism to record the owner or source of the data.

Ontologies and Xonyms : These are an integral part of the model.

Ontologies: Full ontologies (which for this purpose we define as complex domain structures) are associated with properties. Such ontologies are similar to the mesodata concept discussed by de Vries *et al.* (de Vries *et al.* 2004, de Vries 2006).

³Specifically, we allow first order formulae over constants or variables participating in the link plus those accessible through graph traversal. For example, $P(\text{Age}).E(\text{Christopher}) + 2$ references the value associated with link between the entity with entity identifier *Christopher* and the Property *Age* while $\max(P(\text{Age}).E(*).P(\text{InDept}[\text{Consultant}].\text{Sales}))$ returns the maximum *Age* of all entities with a link to Property *InDept* with a value of *Sales* and a role of *Consultant*.

Xonyms: There are a variety of common binary references that are used and understood widely. Xonyms allow such common linkages between objects to be recorded more simply. For example, a Person can be found to *be the same as* another. A Meeting *is a form of* Communication and so on. In these cases *synonym* and *hypernym* references would be created resp.

Other Xonyms include acronyms, holonyms (and meronyms), hypernyms (and hyponyms), metonyms, pseudonyms, and synonyms. The alternative, merging Entities, would result in a loss of both information and provenance, particularly if the reason for the merge was later discredited.

While in many cases it is not difficult to extend most query languages and data mining routines to be able to understand the semantics of ontologies and Xonyms, by making them part of the model it is possible that some systems need not have to do so. For example, association mining routines might be given their input data with all synonyms resolved.

4.2 The LitER Architecture

While the LitER model is independently useful, an overarching architecture has been developed which maximises the benefits of the model (shown in Figure 3). Some of the important points are discussed below.

Analysis Routines. Four sets of routine are made available to the user:

Predicate Definition and Data Dictionary.

These provide a resource to allow easy reference to data items.

Query Languages. It is possible to provide a form of SQL which resolves the terms provided by reference to the Predicate Definitions and Data Dictionary. For example, the query:

```
SELECT EmpNAME
FROM EMPLOYEE
WHERE EmplAGE < 25;
```

might (depending on how the data was organised) be resolved by the following definitions:

```
EMPLOYEE ::= {E(*) . P(WorkFor)}
EmpNAME ::= P.(HasName) . EMPLOYEE
P.HasAge ::= V.(WasBorn) . E(*)
            - TODAY()
EmplAGE ::= P.(HasAge) . EMPLOYEE
```

The first statement creates a set of instances of Entity. The second returns the value for the link to the Property *HasName*. The third creates a virtual property of *HasAge* which exists for all instances of Entity with a link to an Event of type *WasBorn*. The last returns that value for all instances of *EMPLOYEE*.

Constraint Manager / Hypothesis Checker.

This allows the creation of structures that are either used to constrain the data or as putative hypotheses that can be checked against the data.

Data Mining Routines. One of the drawbacks of many data mining systems is the lack of reuseability caused, in part, by

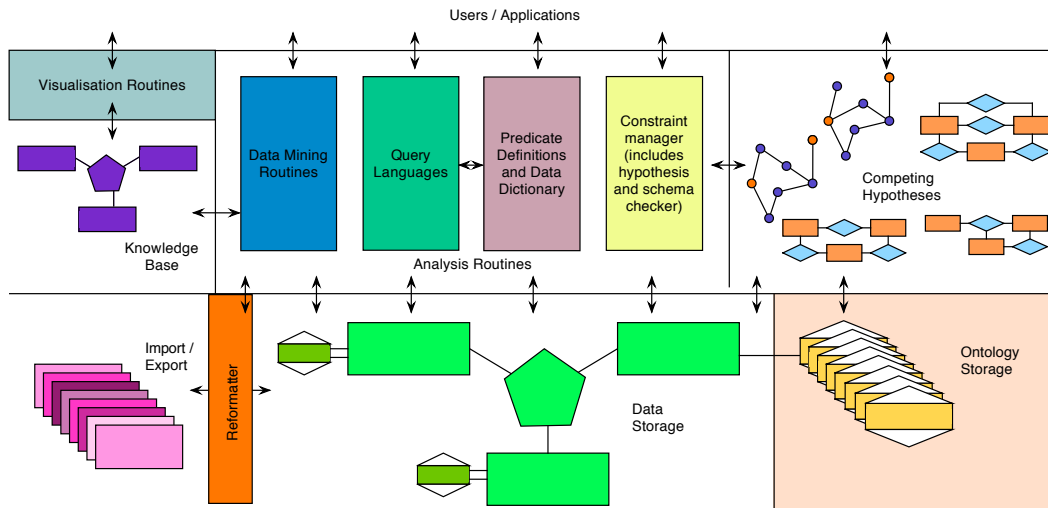


Figure 3: LtER Architecture

changes in the manner in which data are stored. LtER accommodates data mining routines which now have access to a common schema. In particular, graph mining (Chakrabarti & Faloutsos 2006) and association mining (Ceglar & Roddick 2006) have been found to be useful. Importantly, these routines are generally generic and independent of the data. For example, a graph mining routine that seeks to characterise modes of communication between actors can also be applied, given the same LtER schema, to any other graph in which the entities are linked through some event, for example, the transmission of infection.

A knowledge base. In most cases the knowledge base uses the same LtER architecture. Importantly, visualisation routines can operate over the knowledge base and this can take advantage of multiple runs and different forms of data mining.

A general ontology storage area. This can be populated independently of the data storage (ie. it can be made ready in advance of any intended use). To date, the ontologies used have been restricted to complex structures of attribute values à la the mesodata concept (de Vries et al. 2004)⁴.

5 Discussion

The model has a number of important modelling characteristics.

1. Because the objects in a system can play many different roles, entities are not directly typed. Instead their types are recorded by virtue of the properties (and perhaps the property-value pairs) they hold. Thus property owners may be identified by being linked to the property *is an owner* (cf. the category concept of Elmasri et al. (Elmasri et al. 1985)), Australians by having the property-value pair of *having nationality* with value *Australian*. Significantly, this allows

⁴Mesodata aims to add semantic capability by providing greater semantics to the domain of an attribute by allowing attributes to be defined over complex domain structures. For example, while the code for a disease might be defined as CHAR(5), disease codes exist within an agreed international classification (such as ICD10), a tree-structure that relates diseases and other observations by group.

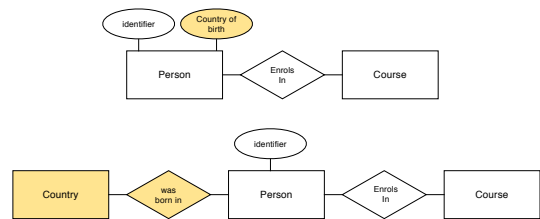


Figure 4: Two design choices

the creation of heterogeneous sets - being *Australian* is, of course, a property that could be assigned to more than just people.

2. As the data stored in such a system is often used for hypothesis creation, the model must also allow for temporal auditing and probabilistic reasoning. Moreover, such systems often obtain data from various sources and therefore not only must the provenance of the data be recorded but also, as far as practicable, the format and content of the data must be maintained (and thus data matching is an important component of the system).
3. Relationships, and their cardinalities, are induced rather than explicitly stored. In many cases, the choice to make a property of an object an attribute rather than a relationship to an entity-type representing the concept is largely dependent on the data available (qv. the semantic ambiguity as discussed by Wand, Storey and Weber (Wand et al. 1999)). Consider the example in Figure 4 in which either an attribute or a relationship-type and entity-type are used. In this respect, the LtER model mirrors the bottom-up approach used by ORM.
4. Unlike the EER Model, relationships (renamed here as links to avoid confusion) are polymorphic; a variable number of Entities can be required to provide the key for a link. For example, the property *has colour* may be specified with a *part number* and the *project* and/or with just a *part number*. In LtER, this polymorphic use of relationships is allowed subject to there not being a constraint forbidding it.
5. The recording of constraints used to gradually refine the model can be used to either validate

data (either before or after DBMS commit) or to validate the schema, and to determine contradictory information. Moreover, constraints can take the form of hypotheses, in which ideas can be tested and the extent of missing information ascertained.

Note that the systems for which the LtER model is most suited do not necessarily have low volumes of data. What distinguishes the model for these systems is that the information held is diverse with some coming from large databases but with the structure of other information being more or less specific to one or a few entities only.

6 Conclusions and Further Work

The LtER model and architecture is being developed as a result of genuine industry requirements and is being applied in both a defence and health environment. Interestingly, as has been noticed in some information analysis research (e.g. Spencer 2001), there is a cascade effect in which the more data is added, the greater the overall connectivity between data objects is experienced.

There has been considerable discussion as to the role of Reiter's closed world assumption (Reiter 1978) and whether it can be assumed to hold in the context of some systems. This, and the accommodation of negative information is currently the subject of further investigation.

While the LtER model does not fulfil all conceptual modelling needs, (we restrict ontologies to conform to the concepts of mesodata domains for example), its development is generating substantial interest. In particular, its ability to rapidly provide areas for subsequent investigation and its ability to quickly bring together data from a variety of data sources has been of substantial interest and it is in this domain that future development work is being directed.

References

- Booch, G., Jacobson, I. & Rumbaugh, J. (2005), *Unified modelling language user guide*, 2 edn, Addison Wesley Professional.
- Ceglar, A. & Roddick, J. F. (2006), 'Association mining', *ACM Computing Surveys* **38**(2).
- Chakrabarti, D. & Faloutsos, C. (2006), 'Graph mining: Laws, generators, and algorithms', *ACM Computing Surveys* **38**(1).
- Chen, H., Zeng, D., Atabakhsh, H., Wyzga, W. & Schroeder, J. (2003), 'COPLINK: managing law enforcement data and knowledge', *Communications of the ACM* **46**(1), 28–34.
- Chen, P. P.-S. (1976), 'The entity-relationship model - toward a unified view of data', *ACM Transactions on Database Systems* **1**(1), 9–36.
- Chen, P. P.-S. (2006), Suggested research directions for a new frontier - active conceptual modeling, in D. Embley, A. Olivé & S. Ram, eds, '25th International Conference on Conceptual Modeling (ER 2006)', Vol. 4215 of *LNCS*, Springer, Tucson, AZ, pp. 1–4.
- de Vries, D. (2006), *Mesodata : Engineering Domains for Attribute Evolution and Data Integration*, PhD thesis, Flinders University.
- de Vries, D., Rice, S. & Roddick, J. F. (2004), In support of mesodata in database management systems, in F. Galindo, M. Takizawa & R. Traunmüller, eds, '15th International Conference on Database and Expert Systems', Vol. 3180 of *LNCS*, Springer, Zaragoza, Spain, pp. 663–674.
- Elmasri, R., Weeldreyer, J. A. & Hevner, A. R. (1985), 'The category concept: an extension to the entity-relationship model', *Data and Knowledge Engineering* **1**(1), 75–116.
- Halpin, T. (1998), Object-role modeling (ORM/NIAM), in P. Bernus, K. Mertins & G. Schmidt, eds, 'Handbook on Architectures of Information Systems.', Springer-Verlag, Berlin, pp. 81–101.
- Lefebvre, S. (2004), 'A look at intelligence analysis', *International Journal of Intelligence and Counter-Intelligence* **17**(2), 231–264.
- Popp, R., Armour, T., Senator, T. & Numrych, K. (2004), 'Countering terrorism through information technology', *Communications of the ACM* **47**(3), 36–43.
- Reiter, R. (1978), On closed world databases, in H. Gallaire & J. Minker, eds, 'Logic and Databases', Plenum Press, New York, pp. 55–76. Reprinted in *Artificial Intelligence and Databases*. J. Mylopoulos and M.L. Brodie (eds.), Morgan Kaufmann, 248–258.
- Roddick, J. F., Ceglar, A., de Vries, D. & La-Ongsri, S. (2007), Postponing schema definition : Low instance-to-entity ratio (LtER) modelling, in P. Chen & L. Wong, eds, 'ACM-L workshop proceedings', Vol. 4512 of *LNCS*, Springer, Tucson, AZ, USA, pp. 206–216.
- Roddick, J. F., Craske, N. G. & Richards, T. J. (1996), 'Handling discovered structure in database systems', *IEEE Transactions on Knowledge and Data Engineering* **8**(2), 227–240.
- Roddick, J. F. & de Vries, D. (2006), Reduce, reuse, recycle: Practical approaches to schema integration, evolution and versioning, invited keynote address, in F. Grandi, ed., '4th International Workshop on Evolution and Change in Data Management (ECDM 2006)', Vol. 4231 of *LNCS*, Springer, Tucson, Arizona, pp. 209–216.
- Seifert, J. W. (2004), 'Data mining and the search for security: Challenges for connecting the dots and databases', *Government Information Quarterly* **21**(4), 461–480.
- Shoshani, A. & Wong, H. K. T. (1985), 'Statistical and scientific database issues', *IEEE Transactions on Software Engineering* **11**(10), 1040–1047.
- Sommerville, I. (2006), *Software Engineering*, 8th edn, Addison-Wesley, Boston, MA, USA.
- Spencer, J. (2001), *The Strange Logic of Random Graphs*, Springer.
- Thalheim, B. (2000), *Entity-Relationship Modeling: Foundations of Database Technology*, Springer, Berlin.
- Verheijen, G. & van Bekkum, J. (1982), NIAM: an information analysis method, in 'IFIP WG8.I Working Conference', Information Systems Design Methodologies: a comparative review, North Holland Publishing, Netherlands.

Wand, Y., Storey, V. C. & Weber, R. (1999), 'An ontological analysis of the relationship construct in conceptual modeling', *ACM Transactions on Database Systems* **24**(4), 494–518.