Software Tools for the Implementation of Relational Databases Describing the Building Product

J.P. Pantouvakis
Computer Science Department,
University of Nottingham,
Nottingham, NG7 2RD,
U.K.

Abstract
The implementation of relational databases describing the building product has to face difficulties imposed by the very nature of buildings, such as large amounts of data with complex interrelationships which are time and place dependent and dynamically changing integrity constraints. This paper suggests that a succesful implementation should be based on relatively few concepts and a set of powerful techniques by which these concepts can be transformed into a database. Subsequently, these techniques can be used as the basis for the development of software tools. When software tools are available, the creation and/or modification of the database describing the building product can be easily achieved. Two such tools have been identified, designed and implemented at Notting
ham University and are described in this paper. A database design tool, based on state-of-the-art relational database theory and a generalized integrity preservation tool. These tools can be linked together to form an integrated building product modelling environment. The major advantage of this approach is that the dynamic in nature Building Product Model produced is independent from software and hardware systems and is capable of adapting to different working practices.

1. Introduction
In recent years the building industry has started to take advantage of new developments in Information Technology [Bjork, 1985 and CICA, 1987]. These developments have been assisted by the availability of personal computers and an upsurge of research interest covering a wide range of application areas which resulted in cheap off-the shelf software.

Such proliferation of hardware and software systems is not, however, without problems. The lack of standards resulted in incompatible systems [Fisher et al, 1985] which in turn hampered the unification of approach of an already highly fragmented industry.

It would be desirable [CICA, 1979] to produce an integrated information system describing the building product. This system should be based on relational database theory which since the presentation of the original paper by E.F. Codd [Codd, 1970] has grown to a subject in its own right. Relational database theory offers a set of methods for the structuring of information which is far more efficient than any other database model in existense [Date, 1986].

Unfortunately, integrated building product information systems based on relational database concepts are not yet available. A research program on the subject is just starting in Finland [Bjork, 1988]. Before such integrated systems emerge, however, there is a lot to be done in conceptualizing the methods and producing tools to aid the development. This paper presents and evaluates such software tools designed and implemented at Nottingham University.

2. Building Product Models
A Building Product Model (B.P.M.) is a conceptualization of some aspects of a real building. Such a model is required by many functions of the Design & Construction (D & C) process. The aspects of a real building that need to be modelled, depend upon the function served. For example, a bill of quantities may serve as such a model for the estimating function.

Different functions require different models of the same building (Fig. 1(a)). These models may overlap (i.e. more than one contain the same piece of information) or contain information gaps (i.e. a model being incomplete when the information missing could be derived from another model). The operation of such a collection of models has a number of major disadvantages, such as update anomalies and integrity preservation anomalies.

An alternative approach is the development of an integrated B.P.M. containing all needed information and capable of producing different "views" according to the function served. Fig.

1(b) depicts such an integrated B.P.M.. Such a model as a product of a logical process, will be independent of current hardware and software technology.

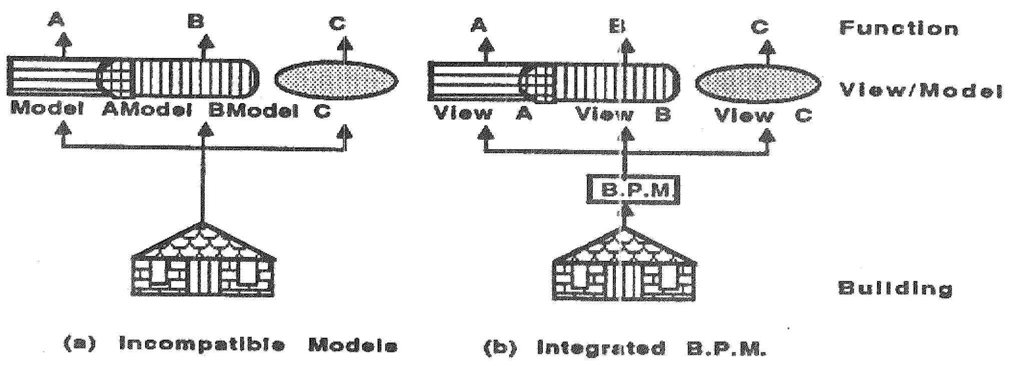

Fig.1 Traditionally different functions are served by incompatible models. When a basic conceptual model is introduced, the major disadvantages of the incompatible models will be overcome and at the same time the end user practices will not be affected.

Such basic "conceptual" models capable of "presenting" themselves in different ways (or views) according to users' requirements are not specific to the building industry. They were introduced [CODASYL, 1971] as an aid to the information system developer forming a data abstraction level at which one specifies *what* data should be stored and *what* are their interrelationships.

3. Properties of the Building Product Model

The main properties of the B.P.M. are :

1. It should be complete, i.e. it should contain all the information needed to the various functions involved in the D & C process.
2. It should be capable of producing different views of information according to the function served.
3. It should be minimal. Redundant information should only be included if it's derivation consumes significant time.
4. It should be consistent. Neither the information contained in it not the deductions made from it, should lead to logical contradictions.
5. It should allow the representation of information at different levels from very broad to very detailed according to managerial needs.
6. It should be able of being implemented in stages (because of the complexity and diversity of the subject, it is unlikely that a B.P.M. can be produced in one step)
7. It should be able to store and manipulate a wide range of data types, such as variables, dates, different methods of construction, planning and costing, pictures e.t.c.
8. It should assist decision making by providing at least as much information as manual systems.
9. It should be an adequate means of communication between the participants of the D & C process.
10. It should not cloud the responsibilities of the participants.
11. It should be easily implemented (to accept wider use)
12. It should be independent from the end user interface.
13. It should preserve the security of its users.
14. It should be suitable for computerized processing.

4. On the Design of the Building Product Model

The design of a B.P.M. shares with other design activities a number of properties. The most important one is that it consists of three stages, namely conceptualization (the designers should know what it is they should design), testing (they should have means of finding out whether or not they have achieved what they set out to achieve) and modification (they should know ways by which an unsatisfactory design can be changed).

Iteration is at the heart of the design process. Design loops around these three stages until a satisfactory result is reached. These iterations, however, do not necessarily lead to better designs in a sequential fashion. Different designs form a tree structure where significant backtracking may be required. [Rich, 1983].

In addition to these difficulties faced by any design activity, further ones are imposed by the very nature of the building product. These can be derived from the properties of the B.P.M. and can be classified under two headings.

a) Difficulties imposed by the large amount of data required to describe the building product which is time and place dependent. For example, as new construction methods replace old ones, their data requirements may not coincide. Consequently, the B.P.M. should be modified to allow for the introduction of these new methods.

b) When the B.P.M. changes due to a), the constraints imposed on data to maintain it in working order also change. Thus, the designer of the B.P.M. should take into account the dynamic nature of integrity constraints.

Such considerations show that it is worth spending some time trying to identify and produce tools that would reduce the time taken for a full design cycle. Two broad categories of tools can be identified:

a) Relational database design tools.

b) Generalized data integrity tools.

In the followings the design and implementation of these tools will be discussed.

5. Relational Databases-Basic Definitions

Attributes are identifiers taken from a finite set $A = \{ a_1, a_2, ..., a_n \}$. Each attribute has associated with it a domain, denoted $DOM(a_i)$, which is the set of possible values for that attribute.

A relation scheme on an attribute set A, is a finite subset of the cartesian product $DOM(a_1)$ X $DOM(a_2)$ X... X $DOM(a_m)$ and is denoted by r_j. The elements of a relation scheme are called tuples.

A key for a relation scheme r_j is a unique identifier for the r_j relation scheme. This means that a key is an attribute (or a set of attributes) from r_j with the property that, at any given time, not two tuples in r_j have the same value for that attribute (or for that set of attributes).

A functional dependency (FD) from an attribute set X to an attribute set Y, denoted X->Y, (X,Y in r_j), exists in r_j, if independent of time, for every X-value that appears in r_j, the corresponding Y-value is unique.

Let X,Y,Z be arbitrary attribute sets in some relation scheme r_j. A multivalued dependency (MVD) from X to Y, denoted X ->-> Y, exists in r_j, if the set of Y-values matching a given (X-value, Z-value) tuple in r_j depends only on the X-value and is independent of the Z-value.

6. A Generalized Database Design Software Tool

The goal of a relational database design methodology is to produce a set of relation schemes containing non-redundant, lossless and easily retrievable information. Two relational database design methodologies are well-known:

a) The decomposition approach which was originally introduced by Codd [Codd, 1971] and was generalized by Zaniolo [Zaniolo, 1976]. A systematic presentation of the method can be found in [Tanaka et al, 1977]. After the introduction of multivalued dependencies [Fagin, 1977a], the method was extended [Fagin, 1977b].

The methodology accepts as input an initial set of relation schemes, along with a set of functional and multivalued dependencies. By using the information contained in the dependencies, the initial set of relation schemes is decomposed a number of times until a satisfactory set of relation schemes is achieved.

The major drawback of this approach is that the output of the method is severely limited by its input. In fact, the final design cannot be very different from the initial one, because the decomposition can only go "down", not "up" or "sideways" [Fagin, 1977b].

b) The synthetic approach was first discussed by [Delobel et al, 1973] and [Wang et al, 1975]. [Bernstein et al, 1975] and [Bernstein, 1976] presented systematic design algorithms. The method was later extended by [Fagin, 1977b] to cope with multivalued dependencies.

The design process begins with a set of functional and multivalued dependencies. This set is subsequently refined to obtain an equivalent more desirable set of dependencies which in turn is used to synthesize the set of relation schemes.

The major advantage of this method is that it does not require an initial design of the set of relation schemes and thus is not limited by its accuracy.

Such considerations show that it is advisable to follow the synthetic approach to relational database design. In fact, the software produced at Nottingham synthesizes relation schemes from a set of dependencies. The user provides the program with a dependency matrix. The program then produces a set of relation schemes (Fig. 2)

Σ = { 1->2,3   (1,4)->5,6   (2,3)->1   (2,3,4)->5,6 }
Σs= { 1->2,3 }
Σc= { (1,4)->5,6   (2,3)->1   (2,3,4)->5,6 }
n = 6
k = 3

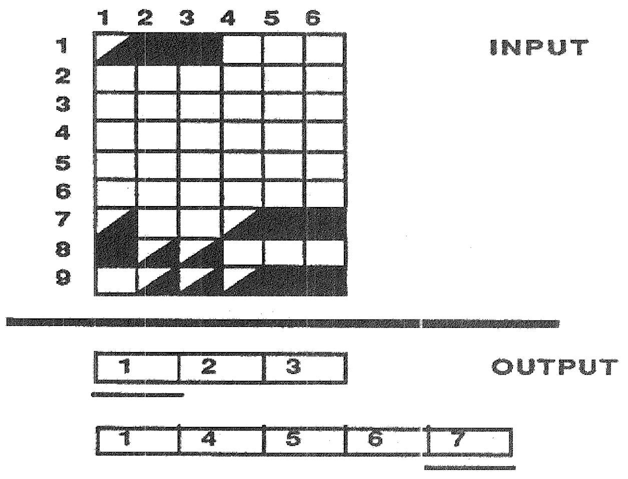

Fig.2 Let Σ be a set of FDs on a set of attributes A={1,2,...,n}. Let Σs be the set containing the FDs in Σ whose left subset contains one attribute, Σc be the difference of Σ and Σs and k be the number of FDs in Σc. Σ is converted to a [(n+k) X n] dependency matrix (DM). A dependency i->j in Σs is represented by the i^{th} row of the DM. A dependency in Σc is represented by one of the n+1, n+2, ..., n+k rows of the DM. The attributes in the left subset of a dependency are represented by a black triangle where those in the right subset by a right rectangle in the appropriate row and in the corresponding column. In the example above, a [(6+3) X 6] DM is needed. The 1st row of this DM represents the first dependency in Σ, where the 2^{nd}, 3^{rd} and 4^{th} dependencies in Σ are represented by rows 7, 8 and 9 respectively. This DM is provided as input to the program. The program then produces a set of relation schemes, indicating the (candidate) keys for each relation (shown here underlined).

A minor complication faced by the designer is the fact that some database designs although more logically correct are less desirable due to performance requirements ([Fagin, 1977b] and [McFadden et al, 1985]). Taking this fact into acount, a program was developed that produces a set of relation schemes as soon as possible. Any subsequent designs are then stored for future reference. At the end, the designer is provided with a set of alternative designs each one of which is logically better than the previous one. The designer can then select the set of relation schemes that suits his needs best.

Before any of the designs is adopted, a final test should be performed. This test will reveal whether the design in question possesses the lossless-join property, since without it information will be lost. A tableau method for testing this property has been described in [Aho et al, 1979]. A program based on this tableau method has been produced. The program accepts as input a set of relation schemes and a set of functional dependencies and returns either lossless or lossy. The time savings from using the program are important, since the method takes $O(s^4)$ time, where s is the space needed to write down the set of relation schemes and the dependencies.

If the set of relation schemes is lossless then it can be adopted. If it is lossy, the set of attributes and dependencies should be modified [Fagin et al, 1982] and the whole design process should start again. This fact alone could justify the assumption that a database design tool is needed.

8. A Generalized Integrity Preservation Software Tool

The term integrity refers to the accuracy or correctness of the data in the database. There has not been much work done so far on data integrity as such. Only a few texts devote a considerable part to data integrity (e.g. [Fernandez et al, 1981]). Other good discussions can be found in [Wiederhold, 1983], [Ullman, 1982] and [Date, 1986]. The main reason seems to be that most databases consist of a small number of relation schemes with few time-independent integrity constraints.

As far as systems are concerned, only a few provide general data integrity handling facilities ([Stonebraker, 1986], [Zloof, 1978], [IBM, 1982]). Most integrity checking is done by user written procedural code. The major disadvantage of this approach is that when a modification to a relation scheme or a change to an integrity constraint is introduced, the integrity checking code needs to be ammended or even re-written from scratch in some cases.

It would obviously be preferable to be able to specify integrity constraints in a declarative fashion and let the system perform the checking. In this case, an integrity constraint would be expressed as a predicate that all correct tuples of the database should satisfy. A simple example of such a predicate might be that a field in a relation scheme should be positive. If the user attempts to violate a constraint, the system should reject the operation.

Five types of integrity constraints can be identified:

a) Key Constraints, i.e. the key fields should be unique and not null.

b) Matching Fields or Foreign Key Constraints, where a field f_1 of a relation scheme r_1 should match a field f_2 of a relation scheme r_2, where r_1, r_2 are not necessarily distinct.

c) Domain Constraints, which are implied by the fact that a particular field is defined on a particular domain. For example, such a constraint could be that a field containing time units can only accept values from the set {sec, min, hour, day, week, month, year}.

d) Range Constraints, where a (numeric) field should only take values within specified limits.

e) Format Constraints, where the format of the values of a field should conform to a specific format. For example, for a date field, this could be [day/month/year].

A generalized system to assist integrity preservation could consist of two parts:

a) A relation scheme holding

1. The kind of operations that the integrity constraints apply. (because there may be constraints which should be enforced only in special circumstances).
2. The specific "weight" of the integrity constraint (i.e whether its violation is fatal or a correcting action can be taken).
3. The integrity constraints themselves stored as predicates.

b) A generalized program triggered every time a change in the set of relation schemes is about to happen. This program should parse and evaluate the appropriate tuples from the integrity relation scheme. Depending upon the result of this evaluation, the program should either accept the operation or abandon it.

If a modification is introduced, the user could then query the integrity relation scheme and modify the appropriate tuples as required. No re-programming would be necessary.

The system described so far, has been implemented using a well-known commercial product, dbase iii plus, running on an IBM PC/AT compatible machine. The structure of the relation scheme occupies 0.15K (which means that thousands of itegrity constraints can be stored on a single diskette) and the compiled version of the program about 40K.

9. Conclusions

The development of a B.P.M. serving all functions of the D & C process is a highly complex operation due to the large amount of data and interralationships involved. The number of different practices in existense and the ever changing technology introduce one additional difficulty: The B.P.M. should be flexible to adapt.

The traditional approach to this requirement is the development of a "huge" model capable of "anticipating" the future and incorporating any changes. This paper suggests an alternative approach; the development of a dynamic B.P.M. Due to the complexity involved, a computer solution is justified. To build such a dynamic B.P.M., a B.P.M. development software environment is proposed.

This environment should consist of a set of relation schemes and a set of integrated software tools where the latter should assist the design and maintenance of the former. More precisely, the set of tools should assist:

a) Addition /deletion of attributes in a relation scheme.
b) Addition /deletion of relation schemes.
c) Addition /deletion of integrity constraints.

The major advantages of this approach are :

1. The dynamic B.P.M. is upwards compatible with developments of hardware and software systems. A particular case in point is the end user presentation of the system. The system can use graphical interfaces and take advantage of shortened access times as dictated by the state-of-the-art computer technology.

2. The dynamic B.P.M. is upwards compatible with more complicated conceptual models. The degree of complexity incorporated in the model can reflect the currently accepted level of sophistication (by the building industry).

3. The expressive power of the B.P.M. is not compensated against the potential integration it offers. All functions of the D & C process are served by the same consistent B.P.M. via different views.

As should have become apperent to the initiated reader, this paper suggests the application of new techniques to the well-known integration problem of the building industry for the development of a new system. As with the development of any new system, it is important not to freeze the methods of the past into the systems of the future. This paper sets out to open a little of what lies behind the application of such new techniques.

Acknowledgements

It is a great pleasure to acknowledge my debt to Dr. M. O'Brien (University of Nottingham) for his continuous help, support and encouragement. I would also like to thank Dr. M. Wybrow (Transfer Technology Ltd., Birmingham) whose useful comments have benefited enormously this work.

References

1. A.V. Aho, C. Berri, J.D. Ullman, 1979, The Theory of Joins in Relational Databases, ACM TODS, 4, 3, September 1979, pp.297-314.
2. P.A. Bernstein, J.R. Swenson, D.C. Tsichritzis, 1975, A Unified Approach to Functional Dependencies and Relations, Proc. ACM SIGMOD, W.F. King (ed), San Jose, California, May 1975, pp.237-245.

3. P.A. Bernstein, 1976, Synthesizing Third Normal Form Relations from Functional Dependencies, ACM TODS, 1, 4, December 1976, pp.277-298.
4. B.C. Bjork, 1985, Computers in the British Construction Industry, Technical Research Centre of Finland.
5. B.C. Bjork, 1988, RATAS-A Proposed Building Product Model, presented at the Seminar on Knowledge Based Design in Architecture, Otaniemi, Finland, August 1988.
6. C.I.C.A., 1979, Computer Programs for Construction Management, User Report No.4
7. C.I.C.A., 1987, Building on IT-A Survey of Information Technology Trends and Needs in the Construction Industry, Peat Marwick McLintock.
8. CODASYL Data Base Task Group April 71 Report, 1971, ACM, New York.
9. E.F. Codd, 1970, A Relational Model of Data for Large Shared Data Banks, Communications ACM, 13, 6, pp. 377-387.
10. E.F. Codd, 1971, Further Normalization of the Data Base Relational Model, Courant Computer Science Symposium 6, Data Base Systems, Prentice-Hall, New York, May 1971, pp.65-98.
11. C.J. Date, 1986, An Introduction to Database Systems, Vol 1, 4th edition, Addison-Wesley.
12. C. Delobel, R.G. Casey, 1973, Decomposition of a Data Base and the Theory of Boolean Switching Functions, IBM Journal of Res & Dev, 17, 5, September 1973, pp.441-446.
13. R. Fagin, 1977a, Multivalued Dependencies and a New Normal Form for Relational Databases, ACM TODS, 2, 3, September 1977, pp.262-268.
14. R. Fagin, 1977b, The Decomposition versus the Synthetic Approach to Relational Database Design, Proc. 3rd International Conference on VLDB, October 1977, pp.441-446.
15. R. Fagin, A.O. Mendelzon, J.D. Ullman, 1982, A Simplified Universal Assumption and its Properties, ACM TODS, 7, 3, September 1982, pp.343-360.
16. E.B. Fernandez, R.C. Summers, C. Wood, 1981, Database Security and Integrity, Addison-Wesley.
17. G. N. Fisher, B.L. Atkin, 1985, A Construction Industry Computer Workstation-Towards an Integrated Management Information System, in P. Brandon (ed), Information Systems in Construction Management, Mitchell's Professional Library.
18. IBM Corporation, 1982, SQL/Data System Terminal Users' Guide, IBM Form Number SH20-2078-0.
19. F. McFadden, J. Hoffer, 1985, Data Base Management, Benjamin/Cummings.
20. E. Rich, 1983, Artificial Intelligence, McGraw-Hill, New York.
21. M.R. Stonebraker (ed), 1986, The Ingres Papers, Addison-Wesley, Reading, Massachusetts.
22. Y. Tanaka, T. Tsuda, 1977, Decomposition and Composition of a Relational Data Base, Proc. 3rd International Conference on VLDB, October 1977, pp. 451-461.
23. J.D. Ullman, 1982, Principles of Database Systems, 2nd edition, Computer Science Press, Rockville, Maryland.
24. C.P. Wang, H. Wedekind, 1975, Segment Synthesis in Logical Data Base Design, IBM Journal of Res & Dev., 19, 1, January 1975, pp.71-77.
25. G. Wiederhold , 1983, Database Design, 2nd edition, McGraw-Hill, New York.
26. C. Zaniolo, 1976, Analysis and Design of Relational Schemata for Database Systems, Computer Methodology Group Report, Computer Science Dept., U.C.L.A., U.C.L.A.-ENG-7669, June 1976.
27. M.M. Zloof, 1978, Security and Integrity within Query-By-Example Data Base Management Language, IBM Research Report RC6982, IBM T.J. Watson Research Centre, New York.