A NOTE ON DECOMPOSITIONS OF RELATIONAL DATABASES

by
C. Beeri and M. Y. Vardi
Department of Computer Science
The Hebrew University of Jerusalem
Jerusalem, Israel

January 1981

INTRODUCTION

One of the main concerns of the theory of relational databases has been the study of data dependencies and their effect on schema construction. The types of dependencies that were studied include functional, multivalued and join dependencies (abbr. fd, mvd and jd, respectively), and recently even more genral types of dependencies. A central idea of schema construction has been the notion of a decomposition; i.e., a schema is decomposed, using its dependencies, into a collection of smaller "independent" schemes. The dependencies are used to ensure that no information is lost in the decomposition process. A formalization of the desirable properties of decompositions led to the definition of the normal forms, in particular the 4'th normal form (4NF) and the project-join normal form (PJNF), which are the ultimate decompositions when mvds or jds, respectively, are given.

However, these normal forms do not solve the problems of schema design. It is known in the folklore of database theory that for some dependency structures no 4NF (or PJNF) decomposition can be constructed without loss of information. Put another way, there are structures that have several 4NF decompositions; choosing any one of these may result in the loss of the information related to the others. This and similar problems motivated some recent reasearch concerned with the application of database theory to practical situations. E.g., several researchers have tried to characterize database schemas that have a "good" 4NF decomposition. The characterization is that the specified set of mvd's is equivalent to exactly one jd [BFMMUY, BG, FMU, Li2, Sc2]. Furthermore, it has been claimed ([Sc2]) that all "real world" database schemas have this property. Other directions of research include the introduction of concepts like objects ([Sc3]) and maximal objects ([MU]) which can be used to bridge the gap between real world semantics and formal database schemes.

The purpose of this note is to present a "real world"

database schema that has three 4NF decompositions, and in our opinion there is no sense in decomposing this schema, except possibly for space considerations. It is also shown that the customary method of normalization by stepwise decomposition ([Fa,Li1]) misses some 4NF decompositions, at least, when applied naively. We intend our example to be for 4NF theory what the example {{CITY,ADDRESS}->ZIP, ZIP->CITY} is for BCNF theory.

THE EXAMPLE

Consider the relation defined on the attributes {E,P,L} with the intended meaning of employee, project and location, respectively, where a tuple <e,p,l> means that employee e works in location l for project p. Assume that an employee should be associated with all locations associated with a project for which he works, and a project should be associated with all locations associated with an employee that works for it. That is, the schema consist of the attribute set {E,P,L} and the set of mvd's {E->->P|L, P->->E|L}.

Now this schema has three 4NF decompositions: {EP,EL}, {EP,PL}, and {EP,EL,PL} (any other 4NF decomposition is equivalent to one of those). However, it is easily verified that there is no single jd equivalent to the given set of mvd's, which means that, no matter which decomposition we choose, we must add an interrelational dependency to ensure the semantic integrity of the database. This renders meaningless the essential rationale of decomposition, which is the ability to maintain the integrity of the database by maintaining separately the integrity of each of the relations resulting from the decomposition. (This problem is called the 'Split Key' problem in [Li2], but it has nothing to do with split keys as is demonstrated by our example).

If we add a fourth attribute C, for children, then the above semantics is captured by the mvd's {E->->P|LC, P->->L|EC}. Consider now the normalization process for this example. We start with the relation scheme {EPLC}. We can apply first the mvd E->->P|LC to get the 4NF decomposition {EP,ELC} or we can apply first the mvd P->->L|EC and then the mvd E->->P|LC to get the 4NF decomposition {EP,EC,PL}. In no way can we obtain {EP,EL,EC} which is also a 4NF decomposition, by using only the given mvd's. A better approach is probably that of [Li2] which takes into account the dependency basis of each of the left hands sides of the given mvd's. In this case, the mvd E->->P|L|C is derivable from the given mvd's. However, even this approach misses the 4NF decomposition

{EP, EL, EC, PL}.

As an aside, this example also shows that the claim in [Sc2] that 'transitivity of mvd's is meaningless' is ,at best, misleading. It is possible that we need the transitivity rule only to compute the dependency bases for the given left sides and after that it is not needed any more. This still has to be justified and proved. However, we cannot do without using this rule.

Let us consider what might be the 'objects' in this example. One possibility is that the sets EP, EL, PL are the objects. However, the jd *[EP, EL, PL] fails to capture the semantics of the situation. Another possibility (which we feel is preferable) is to consider EPL as an object. In this case, we have an object with non trivial dependencies. This is not to say that concepts like 'objects' are useless. On the contrary, this direction of research is very promising. However, one should consider carefully (and perhaps suspiciously?) assumptions like "there are no dependencies on objects".

One proposed method for trouble-shooting faulty database designs is the splitting of attributes [Be, FMU, Sc1]. Thus, one may claim that the attribute L is overloaded with meaning, and that it should be split to L_OF_E and L_OF_P (meaning obviously, location of employee and location of project, respectively). A possible decomposition would be {R(E, L_OF_E), S(P, L_OF_P), T(E, P)}. However, we still have to express and enforce the same semantic constraints. These constraints now take the form of non-typed tuple generating dependencies [BV] as follows:
(1) T(e, p)/\S(p, l)=>R(e, l),
(2) T(e, p)/\R(e, l)=>S(p, l).
It is not at all clear that that makes the problem of maintaining the integrity of the database easier. (Not to mention the recursive unsolvability of the implication problem for binary non-typed dependencies [BV]).

CONCLUSIONS

The history of normalization theory is abundant with naive assumptions that, basically, claim that the world is nice and well behaved. It has taken a series of counterexamples and negative results to uproot some of these assumptions. In this note we presented one more example with the purpose of casting some doubts on a few more of these assumptions. In particular, this example presents additional evidence against the simplistic approach that normailzation solves database design problems by "normalizing", i.e., decomposing. It would perhaps be best to

replace "normalization theory" with "database design theory" since the goal of the theory is not to tell us how to normalize but rather to develop rules and directives for the proper design and use of database schemes.

New directions of research, e.g., the development of concepts like objects and maximal objects seem to be a step in the right direction, namely, a development of a general theory of schema design. However, since 'the true test of the theory is demonstrating its effectiveness in solving day to day database design problems' ([BBG]), non-well-behaved cases can not be avoided, and the theory should supply effective solutions, possibly not so elegant as 4NF, to those cases.

REFERENCES

[BBG] Beeri, C., Bernstein, P.A., Goodman, N.: A sophisticate's introduction to database normalization theory. Proc. 4th ACM Int'l Conf. on Very Large Databases, 1978, pp. 113-124.

[Be] Bernstein, P.A.: Synthesizing third normal form relations from functional dependencies. ACM Trans. on Database Systems 1:4(1976), pp. 277-298.

[BFMMUY] Beeri, C., Fagin, R., Maier, D.A., Mendelzon, A.O., Ullman, J.D., Yannakakis, M.: Properties of acyclic database schemes. To be presented in the 13th Ann. ACM Symp. on Theory of Computing.

[BG] Bernstein, P.A., Goodman, N.: The theory of semijoins. To appear.

[BV] Beeri, C., Vardi, M.Y.: The implication problem for data dependencies. To be presented in the 8th Int'l Coll. on Automata, Languages and Programming. Also, Research Report, Dept. of Computer Science, The Hebrew University of Jerusalem, May 1980.

[Fa] Fagin, R.: Multivalued dependencies and a new normal form for relational databases. ACM Trans. on Database Systems 2:3(1977), pp. 262-278.

[FMU] Fagin, R., Mendelzon, A.O., Ullman, J.D.: A simplified universal relation assumption and its properties. IBM Research Report RJ 2900, Nov. 1980.

[Li] Lien, Y.E.: Hierarchical Schemata for relational databases. To appear in ACM Trans. on Database systems.

[Li2] Lien, Y.E.: On the equivalence of database models. Bell Labs Technical Report, Holmdel, 1980.

[MU] Maier, D.A., Ullman, J.D.: Maximal objects and the semantics of universal relations. Technical Report, SUNY at Stony Brook, 1980.

[Sc1] Sciore, E.: Improving semantic specification in the database relational model. Proc. ACM-SIGMOD Int'l Conf. on Management of Data, 1979, pp. 170-178.

[Sc2] Sciore, E.: Some observations on real world data dependencies. Proc. XP1 Workshop on Relational Database Theory, June 1980.

[Sc3] Sciore, E.: The universal instance and database design. Ph.D. Thesis, TR-#271, Dept. of EECS, Princeton University, June 1980.