RELATIONAL DECOMPOSITION PRESERVING FUNCTIONAL DEPENDENCE

by

W. Chang

Memorandum No. UCB/ERL M78/47

10 July 1978

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

RELATIONAL DECOMPOSITION PRESERVING FUNCTIONAL DEPENDENCE

Wanyen Chang

Berkeley, May 1978

Abstract

This paper presents a procedure which can be automated for generating a complete third normal form decomposition of any relational schema. Complete third normal form decomposition is defined to be a decomposition in third normal form relations which represent all the functional dependences embodied in the original relational schema. Such a decomposition facilitates data base integrity checks, hence is an important tool for relational data base design.

This research is sponsored by National Science Foundation Grant ENG76-84522.

RELATIONAL DECOMPOSITION PRESERVING FUNCTIONAL DEPENDENCE

Wanyen Chang

Berkeley, May 1978

1. Introduction

Problems associated with decomposition of relations have been studied by many researchers [Cod71, Hea71, Del73, Fag77, Ris73]. The motivations seem to be (1) the quest for a theory of logical files, and (2) application of decomposition techniques to relational database design [Fag77].

When decomposing a large file into smaller components, some conditons are often imposed on the components. Two such conditions are

(1) the components together must represent all the information contained in the original relation

(2) the components satisfy certain restrictions.

Several such restrictions have been proposed: Third normal form by Codd, an modification to third normal form by Boyce and Codd, referred to as Boyce-Codd normal form [Cod76], and Fourth normal form by Fagin [Fag77].

This paper presents an approach to relational decomposition which

can be automated. The decomposition thus produced consists of a collection of projections of the given relation, all in third normal form with the property that it preserves all the functional dependences represented by the original relation.

2. Analysis

2.1 Related Concepts And Theorem

The operation of Natural-Join and Heath Theorem are used in decomposition theory to re-compose the original relation from the components. These and other related concepts are discussed here.

Natural-Join. If relation R1 is defined over a set of variables (attributes) A, relation R2 over a set B, and A ^ B = C, a non-empty set. Then the natural join of R1 and R2 on C is defined to be

$$J(A \vee B) = R1(A) * R2(B)$$
$$= \{(a\text{-}c, c, b\text{-}c) \mid a \text{ in } R1 \text{ and } b \text{ in } R2\}.$$

Figure 1 shows an example of natural join. Note that A v B is the set-theoretic union. Thus, each label in C appears exactly once in J. Also, the definition requires that A and B have identical variable labels (e.g. C) in order to be joined. This is clearly not necessary, but is required here to avoid the issue of relabelling the variables in J.

FIGURE 1

R1(A,C) = { (a1,c1), (a2,c1) }

R2(B,C) = { (b1,c1), (b2,c1) }

J(A,B,C) = { (a1,b1,c1), (a1,b2,c1), (a2,b1,c1), (a2,b2,c1) }

Decomposition. Decomposition of a relation R is defined to be a set of projections of R such that the projections can be natural-joined back to R.

Functional Dependence (FD). Suppose V is the variable set of a relation R, and A, B are two subsets.

B is functionally dependent on A, A > B, if at any instant in time any value A assumes in R is associated with exactly one value of B in R. If B is a subset of A, then A > B is a trivil functional dependence.

Heath Theorem.

R(A,B,C) = P1(A,B) * P2(A,C)

if A > B and P1, P2 are projections of R.

The theorem can be put in a slightly more general form as follows:

R(A,B,C,D) = P1(A,B,C) * P2(A,B,D)

if A>BC. Note this equality holds, because A > BC implies AB > C, and by the Heath theorem. The only difference is that the latter

explicitely allows FD among the attributes on the left-hand side of the given FD. Interested reader may check that this generalization does not affect the validity of the original proof given in [Hea71]. In this paper, this form of Heath theorem will be used.

2.2 Observations

From here on, the discussion of relation is mainly concerned with its intension (or schema) rather than its extension (i.e. the elements or tuples in a relation). In a relational database management system, a definition of relational schema consists of the followins

(i) for each relation, the variable set and the underlying domains (data types, ranse of values, etc.)

(ii) all the functional dependences among the variables in the schema.

Observation 1. A decomposition does not necessarily represent all the functional dependences embodied in the relation beins decomposed.

Example R(A,B,C) with the functional dependence structure,

$$S = \{A > B,\ B > C,\ A > C\}.$$

Then by Heath theorem,

$$R = P1(A,B) * P2(A,C).$$

Although P1 and P2 constitute a decomposition of R, and P1

'embodies' A > B and P2 'embodies' A > C, the functional dependence B > C is not 'represented' by the decomposition. Note, however, that

R = P1(A,B) * P2(B,C)

is a decomposition representing all the functional dependences.

Observation 2. A set of projections representing all the functional dependences does not necessarily constitutes a decomposition.

Example R(A,B,C), S = {A > C, B > C}.
Consider, at some instant in time, the extension of R consists of (a1,b1,c1) and (a2,b2,c1). Then the projections P1(A,C) and P2(B,C) does not constitute a decomposition.

Observation 3. Given R, there may be no decomposition in Boyce-Codd normal form that represents all the functional dependences of R.

Example Similar example appeared in [Dat77] and [Ris77].

R(A,B,C), S = {AB > C, C > B}

Boyce-Codd normal form is defined in [Cod74] as a relation in first normal form and for every attribute collection C of R, if any attribute not in C is functionally dependent on C, then all attributes in R are functionally dependent on C. Thus, the only decomposition in Boyce-Codd normal form is the projections P1(C,B) and P2(A,C). The functional dependence AB > C is not

represented by this decomposition.

Observation 4. As we have seen, Heath theorem is the only tool used in third normal form decomposition. (Fagin has proved a stronger theorem and used it in Fourth Normal Form decomposition. (See [Fag77].) But there are cases where Heath theorem alone is not sufficient for obtaining a decomposition representing all functional dependences.

Example R(A,B,C,D), S = {AB > C, CD > B}.

Applying Heath theorem, two decompositions are possible:

R = P1(A,B,C) * P2(A,B,D), and

R = P3(C,D,B) * P3(C,D,A).

Neither represents the entire S.

2.3 Defining The Problem And Its Significance

The issue at hand is database integrity. It is well-known that it is hard and costly to maintain integrity of a database. If the operation to check whether a functional dependence is truely embodied in a relation is very expensive, then clearly it would be much more expensive for the system to check the same on a natural- joined relation, because the natural-join itself is an expensive operation.

Define a complete third normal form (3NF) decomposition of R as a collection of 3NF projections of R which is a decomposition and

represents all the functional dependences (FD) embodied in R. A complete decomposition has a desirable property, i.e. if the integrity of all the FD's embodied in a decomposition is ensured, then the integrity of all FD's represented by the decomposition is automatically ensured. Thus, given a relation R, the problem of determining the existence of a complete 3NF decomposition of R, and constructing it if it exists, is one of theorectical and practical interest.

Note that the notion of completeness of a decomposition is the same as that of independent relations defined in [Ris77]. In fact, it can be readily shown, using Heath theorem and theorem 2 in [Ris77], that the two definitions are equivalent. However, the pairwise decomposition process suggested by Rissanen in that same paper for generating independent relations is not sufficient in general as illustrated in Observation 4 above.

3. Complete 3NF Decomposition

3.1 3NF Synthesis

In section 2, we have shown that Heath theorem alone is not sufficient to generatee a complete 3NF decomposition. Bernstein, in [Ber76], presents an algorithm which, given a set S of functional dependences, produces 3NF relations representing the entire S. The algorithm is outlined in Fisure 2. We will refer to it as

the Synthesizer.

FIGURE 2

1. Eliminate extraneous attributes.
2. Find a nonredundant covering of the given set of FD's.
3. Partition the covering into groups with identical left sides.
4. Merge equivalent keys.
5. Eliminate transitive dependences.
6. Construct relation.

The synthesizer operates under an assumption, which Bernstein calls the Uniqueness Assumption. The example below explains what it is.

Example Let V be the set of variables, S the set of FD's over V. Specifically,

V = {LOC(location), PART(parts names), WARE(warehouse), VOL(volume)}

S = {f1 : LOC > WARE, (a location has one warehouse),

f2 : WARE > LOC, (a warehouse has a location),

f3 : WARE,PART > VOL, (a part in a warehouse has a volume),

f4 : LOC,PART > VOL, (a part at a location has a volume)}

The first two steps of the synthesizer tries to determine the redundant FD's in S, i.e. FD's derivable from the rest. In this example, either f3 or f4 is redundant. Suppose we drop f2 from the input S, assuming that it is permissible to have more than one warehouse at a location. Then f3 and f4 may have different meaning, namely, f3 gives the volume of a given part in a given warehouse, while f4 the volume of a given part at a given location. However, the synthesizer can not tell the difference from the formal definitions and simply concludes f4 is redundant as a derivation of f1 and f3. Clearly, had we use VOL1 in f3 and VOL2 in f4, the synthesizer would reach the correct conclusion. The problem here is a classical one: how to capture enough semantics with proper syntax. The solution is also classic: each formal variable stands for exacty one meaning.

The uniqueness assumption poses a difficult problem for relatioal database designers using the synthesizer. When specifying FD's at design time, it is hard to see where a new label is called for. Putting it in relational terminology, the difficulty is to determine how many different roles each attribute plays in the entire database.

A second difficulty posed to the designer is due to the fact that the synthesizer only handles functional relationships among variables. Thus nonfunctional relationships (e.g. complex mappings between variables) have to be defined with artificial

attributes, e.g. AB > $, if A and B have a nonfunctional relationship.

A third problem is that for a database system to provide inference capability, there must be some linkage between attribue names. One may suggest the use of base name with prefix or suffix as commonly done in COBOL programming to indicate the different roles and their underlying attributes. But this will ineviably introduce complications for detecting redundant FD's which may not be tractable at all.

In contrast, the decomposition process begins with a given relation and a set of FD's embodied in it. The uniqueness assumption always holds in a relation, and if nonfunctional relationships exist, they are already embodied in the relation. Thus applying the synthesizer to decomposition does not have the above-mentioned difficulties. However, data base design using decomposition can not avoid completely the issue of unique attribute name. Rather, the approach may be a more natural way to deal with it. Since this is not the subject matter of this paper, we will not discuss it further. Interested readers are referred to [Fag77-2], in which Fagin compared the two approaches to data base design.

3.2 Existence Theorem

For any relational schema R, the set of projections,

D = {Pi's, Pb},

where {Pb} is the output of the synthesizer, and

Pb, referred to as the base projection of D, is the output of

the Base Reduction Algorithm (shown below),

constitutes a complete third normal form decomposition of R.

The proof of the theorem will be given in section 3.4. In the followings, we present the base reduction algorithm.

Base Reduction Algorithm

Notation. Let f denotes an FD and F a set of FD's. Then,

L(f) is the set of variables appearing on the left side of f,

R(f) is the set of variables appearing on the right side of f,

L(F) is the union of L(f) for all f in F,

R(F) is the union of R(f) for all f in F,

V(F) is the set of variables appearing in F.

Input. R - a set of variables,

E - a set of FD's over R (or its subset) such that for any f1, f2 in E,

if L(f1) > L(f2) is in the closure of E,

then L(f2) > L(f1) is not.

BEGIN

X = R - V(E)

```
Y = R(E)
F = E
FOR every f in F,
    IF L(f) contains more than one variable, or L(f) is not in Y,
       THEN X = X + L(f)
    delete f.
Pb = X
FOR every varible x in Pb,
    K = Pb - x
    F = E
    FLAG = 'down'
    WHILE FLAG = 'down' and F non-empty,
          IF L(f) < K and R(f) = x,
             THEN FLAG = 'up'
          IF L(f) < K and R(f) not = x,
             THEN K = K + R(f)
     IF FLAG = 'up', THEN delete x from Pb.
END
```

Note-1.

If each FD in E is represented by a pair of nodes labelled as L(f) and R(f) respectively, and a directed arc between them. And introduce an auxiliary arc between any two nodes, ni, nJ, if ni < nJ. Then E is represented by an acyclic directed grash. A node is a source if it has no in-arc. The set X in the base reduction

algorithm corresponds to the set of all source nodes in E.

Note-2.

Let C be the non-redundant covering output from the synthesizer. If we replce all the keys equivatent to each other with a new label and keep a separate list of definitions of these labels. Then C satisfies the input restriction on E. For example, if A > B nd B > A are in C, then after relabelling, they all become, say, Z > Z, which is trivil FD and is deleted. Here, we assume that the relabelling is done by the synthesizer. Thus, the output of the synthesizer can be used as the input of the base reduction algorithm.

Note-3.

Since E is acyclic, X contains at least one variable.

Note-4.

If K is empty, the WHILE loop does nothing. Therefore, Pb contains at least one variable. Also, by construction, no FD exists among variables in Pb. This implies that Pb is in third normal form.

3.3 Decomposition Algorithm

We summarize the steps for decomposing a given relation R embodying a set S of FD's.

(1) Input (R,S) to the synthesizer, yielding {Pi} and E.

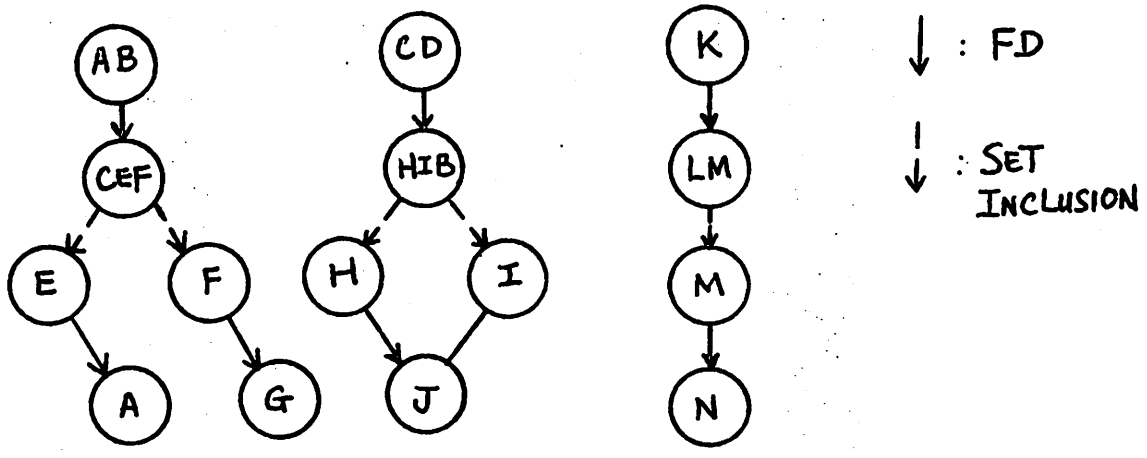
(2) Input (R,E) to the base reduction algorithm, yielding Pb.

(3) Merge PJ,Pk in {Pi,Pb}, if PJ < Pk or Pk < PJ. (This implies the corresponding sets of FD's are also merged.)

The existence theorem asserts that {Pi} from step 1 and Pb from step 2 forms a complete 3NF decomposition. Step 3 only reduces the size of the decomposition. The following two examples illustrate the process.

Example-1. Let R = (A,B,...,M,N) embodying the FD-set S whose graph is shown in Figure 3.

FIGURE 3



After step 1, we have

E = S

{Pi} = {P1(AB,CEF), P2(E,A), P3(F,G)
P4(CD,HIB), P5(H,GJ), P6(I,J),
P7(K,LM), P8(M,N)}

After step 2, we have

X = (A,B,C,D,K)

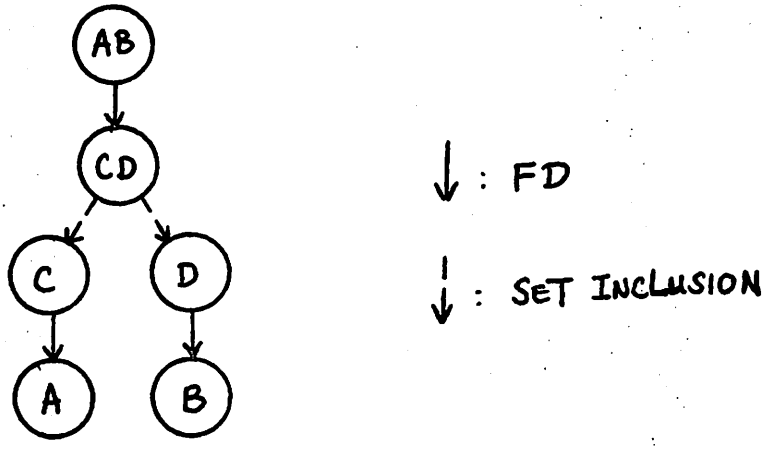
Pb = (ABDK) (or ACDK depending on the order of deletion)

After step 3, we have

D = {Pb, P1, P2, P4,..., P8}. P3 is merged into P1.

Example-2. R = (A,B,C,D) with graph shown in Figure 4.

FIGURE 4



Note R is already in 3NF.

After step 1, we have

E = S

{Pi} = {P1(AB,CD), P2(C,A), P3(D,B)}

After step 2, we have

X = (A,B)

Pb = (AB)

After step 3, we have

D = P1 = R. P2, P3, and Pb are absorbed into P1.

3.4 Proof of The Existence Theorem

We already show that Pb is in 3NF. Also, by the synthesis algorithm, {Pi} are in 3NF and together they represent S, the set of FD's embodied in R. So it remains to show that the generalized Heath theorem can be repeatedly applied to D to yield R.

It should be obvious that any set of projections whose graphical representation is a tree can be joined into a single relation, using the Heath theorem (HT). Let P be a projection whose variable set contains X, the variable set of all the source nodes in E. It follows that P and {Pi} can be joined into a relation using HT.

We now show that Pb can be joined with some Pi's in {Pi} to yield such a P. Let Z = X - Pb = z1,z2,...,zm, and zm is the last variable deleted by the base reduction algorithm to form Pb. Then f : Pb > zm is in the closure of E.

Case 1. If f is in E.

Then the Pi embodying f has the property

$$key(Pi) = Pb$$

Case 2. If f is not in E.

Then there is a derivation Ed < E for f such that any FD h in Ed

$$L(h) < Pb.$$

Hence the Pi embodying h has the property

$$key(Pi) < Pb.$$

Therefore,

$$Pb.i = Pb * Pi,$$

where Pb.i is the projection of R over the variable set {Pb,Pi}. The same argument applies to the next z in Z, etc.. Eventually, we have

$$Pb.i1.i2...ik = Pb * Pi1 * ... * Pik$$

and X < Pb.i1...ik.

Thus Pb.i1...ik can be joined with the remaining projections in {Pi} to yield a single relation R'. Since R' contains all the variables in R, R' = R, by Heath theorem. Q.E.D.

4. Conclusion

We have presented a procedure for generating complete third normal form decomposions. The first step of the procedure, the synthesizer, can be implemented with a time bound of O(L), where L is the length of the string encoding the set S of FD's [Ber76]. The second step can be implemented with a time bound of O(NL'), where N is the number of variables in R and L' is the length of the string encoding the set E of FD'S. The third step is optional. Since in most cases the output and the input of this step would be identical, i.e. no merge can be performed, it would be hard to justify the operational cost of this step. As shown by the examples in section 3.3, a manual scan of the projections is both sufficient and effctive in practices.

5. Acknowledgements

The author would like to express his sincere gratitude to Prof. L.A. Zadeh for his guidance and encouragement. He also wishes to thank C.L. Chang, J. Rissanen of IBM for their helpful comments. R. Fagin of IBM pointed out several errors in the first draft, helped clarify a number of points in regard to natural-join and completeness, and brought to the author's attention a Ph.D. thesis by S.L. Osborn [Osb78], in which an algorithm using directed graph was presented for generating the complete 3NF decomposition (referred to by Osborn as Covering Third Normal Form). His generous help is gratefully acknowledged.

References

Ber76 P.A.Bernsein, "Synthesizing third normal form relations from functional dependencies," Trans. on Database Systems, vol 1, no 4 Dec 1976.

Cod71 E.F.Codd, "Further normalization of the data base relational model," Courant Computer Science Symposium 6, Data Base Systems, Prentice-Hall, N. Y., May 1971, pp. 65-98.

Cod74 E.F.Codd, "Recent investigations in relational data base systems, " IFIP Conf. Proc., North-Holland Publishing Company, 1974, pp. 1017-1021.

Dat77 C.J. Date, Introduction to Database Systems, IBM Systems Programming Series, 1977 Addison-Wesley Publishing Co. Inc., Jacob-Way, Reading, Ma 01867

Del73 C.Delobel, and R.G.Casey, "Decomposition of a data base and the theory of Boolean switching functions," IBM J. Res. Develop. 17, 5 Sept 1973.

Fag77 R.Fagin, "Multivalued dependencies and a new normal form for relational database," Trans. on Database Systems, vol 2, no 3, Sept 1977.

-, "The decomposition versus the synthetic approach to relational database design," Conf. Proc. 1977 on Very Large Data Bases, Tokyo, Japan.

Hea71 I.J.Heath, "Unacceptable file operations in a relational data base, " Proc. 1971 ACM-SIGFIDET Workshop on Data Description, Access and Control.

Osb78 S.L.Osborn, 'Normal forms for relational data bases,' Research report CS-78-06, Department of Computer Science, Univ. of Waterloo, Waterloo, Ontario, Canada, January 1978.

Ris73 J.Rissanen, "Independent Component of relations," IBM RJ1899 (#27239), Oct 1977.