DECOMPOSITION OF A RELATION SCHEME INTO BOYCE-CODD NORMAL FORM

Don-Min Tsou

Computer Science Department
The Pennsylvania State University
University Park, PA 16802

Patrick C. Fischer

Computer Science Department
Vanderbilt University
Nashville, TN 37235

Decomposition into Boyce-Codd Normal Form (BCNF) with a lossless join and preservation of dependencies is desired in the design of a relational database scheme. However, there may be no decomposition of a relation scheme into BCNF that is dependency preserving, and the known algorithms for lossless join decomposition into BCNF require exponential time and space. In this paper we give an efficient algorithm for lossless join decomposition and show that the problem of deciding whether a relation scheme has a dependency-preserving decomposition into BCNF is NP-hard. The algorithm and the proof assume that all data dependencies are functional. We then discuss the extension of our techniques to the case where data dependencies are multivalued.

Key words: relational data base, decomposition, lossless join, dependency-preserving, Boyce-Codd Normal Form.

1. Introduction

Since the relational data model was proposed in the early 1970's by Codd [9], the idea of using semantic information about databases in order to attempt to mechanize portions of the design has received a great deal of attention. The first natural kind of semantic information is the concept of functional dependency (FD) [2]. Here the values of some attributes functionally determine the values of other attributes.

A good database design often means a proper choice of the database scheme, i.e., the collection of relations used. In general, a database designer has to answer the following two questions:

(1) What are the properties of a good database scheme?

(2) How can one find a good database scheme?

A simple answer to the first question is that a good database scheme ought to have both the "lossless-join" property and the "dependency-preserving" property. Furthermore, each relation scheme in a database scheme should be in Boyce-Codd Normal Form (BCNF). However, the answer to the second question is not as easy as the first question since there may be no decomposition of a relation scheme into BCNF that is dependency preserving, [3,5,16]. Thus, obtaining a lossless-join and dependency-preserving decomposition into BCNF is an unrealistic goal. What is actually achievable, therefore, is of interest.

In section 2, we give basic definitions and summarize some known results. In section 3, we will present a polynomial-time algorithm for a lossless-join decomposition into BCNF. This algorithm is a big improvement over the existing algorithms, which require exponential time and space. In section 4, we will show that the problem of deciding whether a relation scheme has a dependency-preserving decomposition into BCNF is NP-hard. Our result strengthens the result of Beeri and Bernstein [3] by eliminating some unnecessary assumptions. In the last section, we will discuss the extension of our techniques to the case where data dependencies are multivalued.

2. Basic Definitions and Known Results

The basic objects studied in this paper are sets of data dependencies which are defined over sets of individual attributes. We will reserve the following

This work was partially supported by NSF Grants MCS-7904129 and MCS-8007706.

letters, with or without subscripts, for the following types of objects:

ABCDE	- individual attributes
UVWXYZ	- sets of attributes
T	- a collection of sets of attributes
fg	- individual functional dependencies
FG	- sets of functional dependencies

All sets discussed are finite.

When context makes clear our intent, we shall abbreviate proper set-theoretic notation by omitting braces when enumerating a set of attributes and using concatenation to denote the union of sets of attributes, e.g., AB means {A,B} and XYZ means X∪Y∪Z. Similarly, F-f means F-{f}.

We now give some basic definitions.

Definition 2.1: A relation r is a table where each column is labeled and represents an attribute and each row represents a record. No two columns can have the same label, and no two rows can be identical (cf.[12]).

There are two relational operations of interest to us: projection and join.

Definition 2.2: Let U be a set of attributes and let r be a relation over U.

(a) For any $X \subset U$, r(X), the projection of r on X, is a relation obtained by taking from each row of r those entries corresponding to the attributes of X and identifying identical rows.

(b) For any family $\{X_1,X_2,\ldots,X_k\}$ of subsets of U such that $X_1 \cup X_2 \cup \ldots \cup X_k = U$, the natural join of the projection of r onto the X_i's is $r(X_1)*r(X_2)*\ldots*r(X_k)$ = {u| for each $1 \le i \le k$, there is a row u_i in $r(X_i)$ such that row u agrees with u_i on the entries of the X-attributes}.

Definition 2.3: Let U be a finite set of attributes. A functional dependency (FD) f over U is an ordered pair (X,Y) of nonempty subsets of U. Following common notation, we shall write f: $X \to Y$ and call X the left side of f, denoted L(f), and Y the right side of f, denoted R(f).

Definition 2.4: Let U be a set of attributes and let F be a set of FD's over U. The pair <U,F> is called a dependency system. A relation r is called a valid relation over <U,F> if for every FD f: $X \to Y$ in F, no two rows of r with identical entries in the X-columns have differing entries in the Y-columns.

Definition 2.5: Let F be a set of FD's over U. For any $X \subset U$, we define $CL_F(X)$ (or simply CL(X)), the closure of X under F, as follows [1,8]:

(1) $X \subset CL(X)$
(2) If $Y \subset CL(X)$ and $Y \to Z$ is a FD in F then $Z \subset CL(X)$.
(3) CL(X) is the least set (with respect to set inclusion) satisfying (1) and (2).

A database scheme is a collection of sets of attributes used to define the relations in the database. A relation scheme is a set of attributes used to define a single relation. A decomposition of a relation scheme $R = A_1\ A_2\ \ldots\ A_n$ is its replacement by a database scheme $T = \{R_1,R_2,\ldots,R_k\}$ such that $R_1 \cup R_2 \cup \ldots \cup R_k = R$ (cf.[16]).

Definition 2.6: Let F be a set of FD's over U and let $T = \{X_1,X_2,\ldots,X_k\}$ be a collection of subsets of U.

(a) T is said to be a lossless-join decomposition of U under F if for every valid relation r over <U,F>:

$$r = r(X_1)*r(X_2)*\ldots*r(X_k)$$

that is, r is the natural join of its projections onto X_i's.

(b) For each $1 \le i \le k$, let F_i be a set of FD's over X_i such that $CL_{F_i}(Z) = CL_F(Z) \cap X_i$ for every $Z \subset X_i$. T is said to be a dependency-preserving decomposition of U if for every $Y \subset U$, we have $CL_F(Y) = CL_G(Y)$, where $G = F_1 \cup F_2 \cup \ldots \cup F_k$.

We note the following result of Aho, Beeri and Ullman ([1], Corollary 1).

Remark 2.7: {X,Y} is a lossless-join decomposition of X∪Y if and only if either $Y \subset CL(X \cap Y)$ or $X \subset CL(X \cap Y)$.

The next definition will formalize the concepts of key and Boyce-Codd Normal Form (BCNF).

Definition 2.8: Let F be a set of FD's over U and let $W \subset U$.

(a) A subset X of W is said to be a key of W under F if $W \subset CL(X)$. Otherwise, it is a nonkey of W under F.

(b) W is said to be in BCNF if whenever $A \in CL(X)-X$ for $A \in W$, $X \subset W$, then X is a key of W under F. (In other words, the only nontrivial dependencies in W are those in which a key functionally determines one or more attributes [16].)

Jou and Fischer [14] have shown the following.

Remark 2.9: For a given set U of attributes and a given set F of FD's over U, if for every FD f in F either CL(L(f)) = L(f) or CL(L(f)) = U, then U is in BCNF.

From Remark 2.9, a polynomial-time algorithm can easily be developed for testing if the universal relation scheme is in BCNF. Decomposition is desired only when the universal relation scheme is not in BCNF.

3. An Algorithm for Lossless Join Decomposition into BCNF.

The correctness of the algorithm presented in the second half of this section is based on the following two lemmas. Lemma 3.1 provides a sufficient condition for a decomposition to have a lossless join, and Lemma 3.2 provides a sufficient condition for a set of attributes to be in BCNF.

Lemma 3.1: Let $T = \{Y_1,\ldots,Y_k\}$ be a collection of subsets of U and let $Y_1 \cup \ldots \cup Y_k = U$. If for every $1 \le i < k$ $\{Y_i, Y_{i+1} \cup \ldots \cup Y_k\}$ has a lossless join, then T is a lossless-join decomposition.

Proof. Follows from Definition 2.6(a), Remark 2.7, and the associativity of the natural join.

Lemma 3.2: Let F be a set of FD's over U and let $Y \subset U$.

(a) If $|Y| \le 2$, then Y is in BCNF.

(b) For $|Y| > 2$, if for any pair of distinct attributes $A,B \in Y$, $A \notin CL(Y-AB)$, then Y is in BCNF.

Proof. (a) Immediate from Definition 2.8(b).

(b) Suppose Y is not in BCNF. Then there exist $A \in Y$, $X \subset Y$, $B \in Y$, such that $A \in CL(X)-X$ but $B \notin CL(X)$. Then $A \in CL(Y-AB)$, a contradiction. (The converse of Lemma 3.2(b) is not true. Consider Y = ABC, with the FD $C \rightarrow AB$.)

The following algorithm offers a significant improvement over the previously known lossless-join decomposition algorithms:

(Algorithm given on following page.)

Algorithm 3.3: Lossless Join Decomposition into BCNF.

Input: A set U of attributes and a set F of FD's over U.
Output: A decomposition of U with a lossless join, such that every set in the decomposition is in BCNF under F.

Procedure:

```
       begin
 1.      X ← U ;
 2.      Y ← Ø ;
      ** after initialization, we start the main loop **
 3.      while X ≠ Y do
          begin
 4.         Y ← X ;
 5.         Z ← Ø ;
 6. repeat: if |Y|>2 then do
           begin
 7.          for each attribute A ∈ Y do
 8.            for each attribute B ∈ Y-A do
                 begin
           ** check if Y-AB → A is logically implied by F **
 9.                 if A ∈ CL(Y-AB) then do
                       begin
                          ** remove B from Y **
10.                       Y ← Y-B ;
11.                       Z ← A ;
12.                       go to repeat;
                       end;
                  end;
            end;
          ** remove Z from X and output a relation scheme Y **
13.       X ← X-Z ;
14.       write Y ;
          end;
       end;
```

Theorem 3.4: Let U be a set of attributes and let F be a set of FD's over U.

(a) Algorithm 3.3 terminates in time $O(|U|^5.|F|)$.

(b) Algorithm 3.3 yields a lossless-join decomposition into BCNF.

Proof.

(a) First let us count the number of times each statement will be executed in each iteration of the main loop. Statements 4, 5, 13, and 14 will be executed only once. Since one attribute is removed from Y (statement 10) before the execution of the go to statement (statement 12), statements 6, 10, 11, and 12 will be executed at most $O(|U|)$ times. Each time when statement 6 is true, statement 7 will be executed at most $O(|U|)$ times and statements 8 and 9 will be executed at most $O(|U|^2)$ times. Hence in each iteration of the main loop, statement 7 will be executed at most $O(|U|^2)$ times and statements 8 and 9 will be executed at most $O(|U|^3)$ times.

Now we consider the main loop. Since one attribute is removed from X at statement 13 in each iteration with the exception of the last one, the main loop iterates at most $O(|U|)$ times. It is therefore clear that statements 8 and 9 will be executed at most $O(|U|^4)$ times. Since the execution of statement 9 is the most time-consuming execution, and it takes time $O(|U|.|F|)$ (for the computation of closure [3]), the total running time of Algorithm 3.3 is $O(|U|^5.|F|)$.

(b) Suppose that Algorithm 3.3 terminates after k iterations of the main loop. For $1\le i\le k$, let X_i, Y_i, and Z_i represent, respectively, the contents of X, Y, and Z at the end of the ith iteration of the main loop (after executing statement 14).

From the construction of the main loop, in each iteration with the exception of the last one, statements 10-12 must be executed at least once and, hence, $Z_i \subset CL(Y_i-Z_i)$ for each $1\le i<k$. Therefore, $Y_i \subset CL(Y_i-Z_i)$. Furthermore, $Y_i-Z_i = Y_i \cap X_i$. Thus, from Remark 2.7, $\{Y_i,X_i\}$ has a lossless join for each $1\le i<k$. One may observe that $Y_k = X_k$, and for every $1\le i<k$, $X_i = Y_{i+1}\cup\ldots\cup Y_k$. Also, $Y_1\cup\ldots\cup Y_k = U$, and from Lemma 3.1 we conclude that $\{Y_1,Y_2,\ldots,Y_k\}$ has a lossless join. Furthermore, for each $1\le i\le k$, we have either $|Y_i|\le 2$ or $A \notin CL(Y_i-AB)$ for any pair of distinct attributes $A,B \in Y_i$. From Lemma 3.2, Y_i is in BCNF for every $1\le i\le k$.

4. The Complexity of Dependency Preserving Decomposition into BCNF.

We first give a lemma which provides a necessary condition for a decomposition to be dependency-preserving.

Lemma 4.1: Let F be a set of FD's over U and let $T = \{X_1,X_2,\ldots,X_k\}$ be a dependency-preserving decomposition over U. Let $f': W \to A$ be a FD in F such that $A \notin W$. If $A \in L(f)$ for every $f \in F-f'$, then $WA \subset X_i$ for some $X_i \in T$.

Proof. For every $1\le i\le k$, let F_i be a set of FD's over X_i such that $CL_{F_i}(Z) = CL_F(Z)\cap X_i$ for every $Z \subset X_i$. If $A \in L(f)$ for every $f \in F-f'$, then by Definition 2.5 $A \notin CL_F(Y-A)$ for any subset Y of U such that $W \not\subset Y$. Suppose that $WA \not\subset X_i$ for every $X_i \in T$. If $A \in X_i$, then $W \not\subset X_i$. Otherwise $A \notin X_i$. In either case, $A \notin CL_{F_i}(X_i-A)$ for every $1\le i\le k$. Thus $A \notin R(g)$ for any $g \in F_1\cup\ldots\cup F_k = G$. By Definition 2.5, $A \notin CL_G(W)$. But rule f' in F implies $A \in CL_F(W)$; thus by Definition 2.6(b) T is not dependency preserving, a contradiction.

Beeri and Bernstein [3] proved that the problem "Is there a BCNF scheme that represents F?" is NP-hard under the assumption that no two relation schemes have equivalent keys. However, we will show in Theorem 4.3 below that this assumption is unnecessary. Furthermore, we present an example of a relation scheme for which the only existing dependency-preserving decomposition into BCNF does not satisfy their assumption.

Example 4.2: Let U = ABCD and let F = {AB → CD, CD → B}. Then {ABC, ABD, BCD} is the only dependency-preserving decomposition into BCNF, where AB is a key for both ABC and ABD.

Theorem 4.3: Let F be a set of FD's over U. The problem of deciding whether there is a dependency-preserving decomposition over U into BCNF is NP-hard.

Proof. To prove NP-hardness, we present a polynomial-time reduction from the hitting set problem, which is NP-complete [15], and can be formulated as follows:

Given a family $\{V_1,V_2,\ldots,V_n\}$ of subsets of a set S, one is to decide if there exists a subset W of S such that W contains exactly one element of each V_i. Such a set W is called a hitting set. Without loss of generality, we assume that $n>1$ and $|V_i|>1$ for some $1\le i\le n$.

Now let $U = S \cup \{B_1,\ldots,B_n,A,C\}$ where $B_1,\ldots,B_n$, A, and C are n+2 new symbols not in S. We shall construct a set F of FD's over U as follows:

(a) $AD \rightarrow B_i$ for each $1 \le i \le n$ and for each $D \in V_i$;

(b) $ADE \rightarrow S-DE$ for each pair of distinct elements $D,E \in V_i$ and for each $1 \le i \le n$;

(c) $AB_1\ldots B_n \rightarrow C$;

(d) $CS \rightarrow A$.

This construction can easily be performed in polynomial time. We claim that there is a hitting set $W \subset S$ if and only if there is no dependency-preserving decomposition over U into BCNF.

Suppose that there is a hitting set $W \subset S$. It is clear that $CL(AW) = AWB_1\ldots B_nC$. Since $|V_i|>1$ for some $1 \le i \le n$, $S \ne W$ follows from the definition of hitting set, and AW is a nonkey.

Then from Definition 2.8(b), CSA is not in BCNF. Since $A \in L(f)$ for every $f \in F-\{CS \rightarrow A\}$, by Lemma 4.1, for any dependency-preserving decomposition T, we have $CSA \subset X$ for some $X \in T$. Since CSA is not in BCNF, any such X is not in BCNF. Thus there is no dependency-preserving decomposition.

Now suppose that there is no hitting set. First we list the following computations of closure, which follow immediately from the construction of F.

(1) $CL(A) = A$;

(2) $CL(B_i) = B_i$, for each $1 \le i \le n$;

(3) $CL(D) = D$, for each $D \in S$;

(4) $CL(DE) = DE$, for any $D,E \in S$;

(5) $E \notin CL(AD)$, for any pair of distinct attributes $D,E \in S$.

A dependency-preserving decomposition into BCNF follows from the construction of F, and we will show that each relation scheme is in BCNF.

(a) From (1), (2), and (3), ADB_i is in BCNF for each $1 \le i \le n$ and for each $D \in V_i$;

(b) From (1), (3), (4), and (5), ADEB is in BCNF for each pair of distinct elements $D,E \in V_i$, for each $B \in S-DE$, and for each $1 \le i \le n$;

(c) Since L(f) contains an element from S for every f in $F-\{AB_1\ldots B_n \rightarrow C\}$, $AB_1\ldots B_nC$ is in BCNF;

(d) For each proper subset Y of CS, since $A \in L(f)$ for every f in $F-\{CS \rightarrow A\}$, we have $CL(Y) = Y$. For any subset $W \subset S$, either $CL(ACW) \cap CSA = ACW$ or, if any of the rules in (b) can be applied, $CL(ACW) \cap CSA = CSA$. Finally, for any subset $W \subset S$, since there is no hitting set, we have either $CL(AW) \cap CSA = AW$ or $CL(AW) \cap CSA = CSA$. Thus, CSA is in BCNF.

This completes the proof.

5. Conclusions and Future Work

The polynomial-time algorithm for a lossless-join decomposition into BCNF and the NP-hardness of the problem of deciding whether there is a dependency-preserving decomposition into BCNF come from the analysis of the lossless-join property, the dependency-preserving property, and the definition of BCNF. The techniques developed in sections 3 and 4 can easily be extended to solve the problems concerning the decomposition into BCNF in the presence of multivalued dependencies (MVD's). Since a FD is a special case of a MVD [11], by using the same construction as in the proof of Theorem 4.3, we can prove that the problem of deciding whether there is a dependency-preserving decomposition into BCNF under both FD's and MVD's is also NP-hard. Furthermore, if we modify statement 9 of Algorithm 3.3 to "if Y-AB → A is logically implied by the given set of FD's and MVD's then do", then we obtain a lossless-join decomposition into BCNF under both FD's and MVD's. Since the membership problem (testing whether a FD can be logically implied by a set of FD's and MVD's) can be solved in polynomial time [4,13], the modified algorithm still runs in polynomial time.

The next natural question is "Is there a polynomial-time algorithm for lossless-join decomposition into 4NF?[10, 11]". However, we know of no simple sufficient condition for testing 4NF analogous to our condition in Lemma 3.2 for the testing of BCNF, and this question is open.

References

1. A. V. Aho, C. Beeri, and J. D. Ullman. The theory of joins in relational databases. ACM Trans. Database Syst. 4, 3 (Sept., 1979), pp. 297-314.

2. W. W. Armstrong. Dependency structures of data base relationships. Information Processing 74, North Holland Pub. Co., Amsterdam, 1974, pp. 580-583.

3. C. Beeri and P. A. Bernstein. Computational problems related to the design of normal form relational schemes. ACM Trans. Database Syst. 4, 1 (March, 1979), pp. 30-59.

4. C. Beeri. On the membership problem for multivalued dependencies in relational databases. TR-229, Dept. of EECS, Princeton University, Sept., 1977.

5. C. Beeri. On the role of data dependencies in the construction of relational database schemes. TR-43, Dept. of CS, The Hebrew University of Jerusalem, Jan., 1979.

6. P. A. Bernstein. Synthesizing third normal form relations from functional dependencies. ACM Trans. Database Syst. 1, 4 (Dec., 1976), pp. 277-298.

7. C. Beeri, R. Fagin, J. Howard. A complete axiomatization for functional and multivalued dependencies. Proc. ACM SIGMOD Conf., Toronto, 1977, pp. 47-51.

8. J. Biskup, U. Dayal, P. A. Bernstein. Synthesizing independent database schemes. Proc. ACM SIGMOD Conf., Boston, 1979, pp. 143-151.

9. E. F. Codd. Recent investigations in relational data base systems. Information Processing 74, North Holland Pub. Co., Amsterdam, 1974, pp. 1017-1021.

10. R. Y. Fadous. Decomposition of a relation into fourth normal form. Proc. Computer Software & Applications Conf., Chicago, 1979, pp. 404-408.

11. R. Fagin. Multivalued dependencies and a new normal form for relational databases. ACM Trans. Database Syst. 2, 3 (Sept., 1977), pp. 262-278.

12. W. L. Gewirtz. The universal relation assumption and decomposition strategies for scheme design. Proc. Computer Software & Applications Conf., Chicago, 1979, pp. 136-140.

13. K. Hagihara, M. Ito, K. Taniguchi. Decision problems for multivalued dependencies in relational databases. SIAM J. Computing 8, 2 (May, 1979), pp. 247-264.

14. J. Jou, P. Fischer. A new view of functional dependency structures and normal forms. Technical Report CS-80-6, Dept. of CS, Pennsylvania State University, Jan., 1980.

15. R. M. Karp. Reducibility among combinatorial problems. In Complexity of Computer Computations, Plenum Press, New York, 1972, pp. 85-104.

16. J. D. Ullman. Principles of Database Systems. Computer Science Press, Potomac, Maryland, 1979.