The implication problem for Inclusion Dependencies: A graph approach

Rokia Missaoui and Robert Godin
Département de Mathématiques et d'Informatique
Université du Québec à Montréal
C.P. 8888, Succursale "A"
Montréal, Québec
Canada, H3C 3P8
(514)987-7939
Missaoui@MIG750.UQAM.CA

Abstract:
In this paper, we propose a graph theoretic approach to deal with the implication problem for inclusion dependencies. By analogy with functional dependencies, we define and present algorithms for computing the following concepts: the closure of a relation scheme R for X according to a set of inclusion dependencies and the minimal cover for inclusion dependencies.

1. Introduction

Functional and inclusion dependencies are the most frequently encountered dependencies in database applications. However, functional dependencies have been the focus of more research [Arm74], [Bern76], [Aho79b], [Sagi81],[KaYo83]. Recently, inclusion dependencies (INDs) have received more attention, particularly for theoretical studies [Casa82], [Mitc83], [Cosm84]. This lesser interest for inclusion dependencies studies is due, in part, to the fact that these dependencies do not affect the normalization process. However, these dependencies can play an important role in query optimization [Miss87].
The purpose of this paper is to present some algorithms related to the implication problem for INDs based on a graph structure. In the next section, we briefly review the relational model, and some basic definitions. In section 3, we propose a definition for an IND-graph. The concepts of closure of a relation R for X and the minimal cover for INDs are presented in section 4.

2. Background

A relation Ri (A1, ..., An) satisfies the *functional dependency* X-->Y , where X and Y are subsets of {A1, ..., An}, if for every pair of tuples t1 and t2 of Ri, t1[X]=t2[X] ==> t1[Y]=t2[Y]. The notion of inclusion dependency [Fagi81] is a generalization of the referential integrity for the relational model [Codd79], [Date86]. Following [Casa82] and [JoKl82], an inclusion dependency is an assertion of the form R1[X] $\subseteq$ R2[Y] where R1 and R2 are relation schemes (not necessarily distinct), X and Y are ordered lists of attributes of the same length. A database satisfies the inclusion dependency R1[J1,....Jj] $\subseteq$ R2[K1,....Kj] if, for every sub-tuple <a1,....aj> appearing in the columns J1,...Jj of a tuple of R1, there exists a tuple of the relation R2 which contains <a1,....aj> in columns <K1,....Kj>.

For the purposes of illustration within this paper, the following medical database will be used:
- HOSPITAL (HCODE, HNAME, ADDRESS, HCAPACITY)
- UNIT (HCODE, UCODE, UNAME, UCAPACITY, PRCODE) where PRCODE is the code of the paramedic responsible for the care-unit. PRCODE is an alternate key for the relation
- DOCTOR (DCODE, DNAME, SPECIALIZATION, DSALARY), where DNAME, and DSALARY are respectively the name and the salary of a doctor.
- PATIENT (PCODE, PNAME, BIRTH, SEX, ADMIT, HCODE, UCODE), where PNAME and ADMIT are respectively the name of the patient and his date of admission to the hospital.
- PARAMED (PRCODE, PRNAME, FUNCTION, PRSALARY, PBONUS, HCODE, UCODE), where PRNAME, PRSALARY and PBONUS are the name, the salary and the bonus of a paramedic.
- HOSPDOC (HCODE, DCODE), where HOSPDOC is a relation that binds HOSPITAL and DOCTOR together.
- DOCPAT (DCODE, PCODE), where DOCPAT is a relation that binds DOCTOR and PATIENT together.

Inclusion dependencies in this DB are:
DOCPAT[PCODE] $\subseteq$ PATIENT[PCODE] (1)
UNIT[HCODE] $\subseteq$ HOSPITAL[HCODE] (2)
DOCPAT[DCODE] $\subseteq$ DOCTOR[DCODE] (3)
HOSPDOC[DCODE] $\subseteq$ DOCTOR[DCODE] (4)
UNIT[PRCODE] $\subseteq$ PARAMED[PRCODE] (5)
PATIENT[HCODE,UCODE] $\subseteq$ UNIT[HCODE, UCODE] (6)
HOSPDOC[HCODE] $\subseteq$ HOSPITAL[HCODE] (7)
PARAMED[HCODE,UCODE] $\subseteq$ UNIT[HCODE, UCODE] (8)
HOSPITAL[HCODE] $\subseteq$ UNIT [HCODE] (9)

Other INDs constraints can be deduced by using inference rules as described by Casanova et al. [Casa82]

and Mitchell [Mitc83]. The inference rules for inclusion dependencies alone are:

- Reflexivity

  $R[X] \subseteq R[X]$

- Permutation and projection

  If $R[A1,\ldots,An] \subseteq S[B1,\ldots,Bn]$

  then $R[Ai_1,\ldots,Ai_k] \subseteq S[Bi_1,\ldots,Bi_k]$ where $1 \le i_j \le n$ and $i_j = i_m \Rightarrow j=m$

Examples:

PATIENT[HCODE,UCODE] $\subseteq$ UNIT[HCODE,UCODE] implies the following three inclusion dependencies:

PATIENT[UCODE,HCODE] $\subseteq$ UNIT[UCODE,HCODE] (permutation)

PATIENT[UCODE] $\subseteq$ UNIT[UCODE] (projection)

PATIENT[HCODE] $\subseteq$ UNIT[HCODE] (projection).

- Transitivity

  If $R[X] \subseteq S[Y]$ and $S[Y] \subseteq T[Z]$

  then $R[X] \subseteq T[Z]$

Example:

PATIENT[HCODE] $\subseteq$ UNIT[HCODE] & UNIT[HCODE] $\subseteq$ HOSPITAL[HCODE]

$\Rightarrow$ PATIENT[HCODE] $\subseteq$ HOSPITAL[HCODE]

Mitchell [Mitc83] also presents three inference rules in which both inclusion and functional dependencies are considered.

3. Inclusion dependency graph

In this section, we introduce a graph representation of the INDs useful for resolving the implication problem. The implication problem [Cosm84] for a set Σ of dependencies and a dependency σ, consists of deciding whether σ holds in every DB extension in which Σ holds. Since this problem is undecidable [Cosm84] and in practical situations, INDs have special structures, many studies were limited to a certain form of IND such as unary IND in which inclusion concerns individual attributes [Kane83], Key-based IND [JoKl82], typed IND [Casa83], [Cosm84] and acyclic IND [Scio83], [Cosm84].

An IND $R1[A1,\ldots,Am] \subseteq R2[B1,\ldots,Bm]$ is said to be typed if $Ai=Bi$ for $1 \le i \le m$ [Cosm84]. A set of INDs is acyclic if there is no IND such that $R[X] \subseteq R[Y]$ and $X \neq Y$ and there are no relations R1, R2, ..Rn ($n>1$) such that $R1[X1] \subseteq R2[Y2]$, $R2[X2] \subseteq R3[Y3]$, ..., $Rn[Xn] \subseteq R1[Y1]$. The implication problem for typed INDs alone is known to be polynomial in time [Casa83]. In this paper we constrain our work to typed IND.

3.1 Definition

Given a set I of INDs on a DB scheme S, the IND-graph for I, denoted by $G_I=(V, E, L)$ is a labelled oriented graph possibly with cycles.The sets V, E and L stand respectively for vertices, edges and labels. Each vertex corresponds to a relation of the DB. If an edge connecting Ri to Rj is labelled by X, it means that $Rj[X] \subseteq Ri[X]$ holds. The number of edges in G_I corresponds to the cardinality of the set I.

Given the set I of inclusion dependencies presented before, the corresponding IND-graph is shown in figure 1.

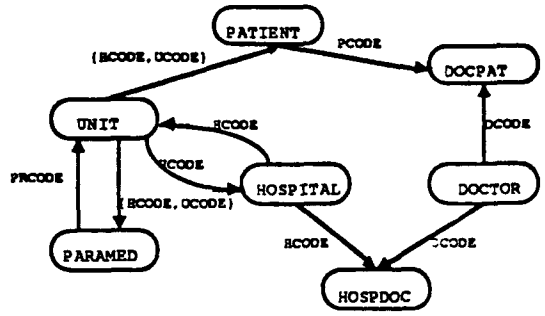

Figure 1. the IND-graph

The IND-graph can be used to answer the following questions:

a) what is the set of relations {Rk |$1 \le k \le n$} such that $Rk[X] \subseteq Ri[X]$ holds? (see closure $R^+(X)$ in section 4.1)

b) does $Rk[X] \subseteq Ri[X]$ hold?(see implication in section 4.2)

c) what is a possible minimal set of non-redundant INDs? (see minimum cover for IND in section 4.3).

Using the IND-graph, we can tell how to obtain the elements cited before:

a) determine the set of vertices in the oriented path (R_i, R_k) such that the visited edges have a label $Z \supseteq X$

b) from vertex R_i, find a directed path to R_k in which the visited edges have some sub-common part X in their label

c) test each edge (Ri, Rj) for redundancy by applying b) to $G_I - (R_i, R_j)$.

3.2 Building an IND-graph

Algorithm 1

Input: a set I of inclusion dependencies

Output: G_I

Begin
For each dependency say $R_j[X] \subseteq R_i[X]$ in I **do**
If there is no vertex R_i or R_j **then**
Begin
Create the corresponding vertex
Create a directed edge (R_i, R_j) labelled by X
End
EndIf
EndFor
End

Analysis:
Finding a node can be done in constant time using a table indexed by the relation index i. Creating an edge can be done in constant time and finding a node is done twice for each dependency, so the worst case time complexity is O(III).

4. The implication problem for inclusion dependencies

By analogy with functional dependencies, we define and present algorithms for computing the following concepts: **closure** of a relation scheme R for X according to a set of inclusion dependencies and **minimal cover** for inclusion dependencies.

4.1 The closure $R^+(X)$ according to a set of IND

The closure $R^+(X)$ of a relation R for X according to a set of INDs is a set of relations R_k such that $R_k[X] \subseteq R[X]$. Such a set can be derived using the inference axioms. This closure includes R, by virtue of the reflexivity axiom.

Algorithm 2
{Given a directed labelled graph $G_I=(V, E, L)$ and a set X of attributes, compute $R^+(X)$ }
Procedure Closure (vertexR_i,X)
If vertex Ri is not visited **then**
Begin
Add R_i to R^+
Mark vertexR_i as visited
For each edge (R_i,R_j) labelled by Y where $X \subseteq Y$
do
Closure(vertexR_j,X)
End
EndIf
End Closure

Initialize R^+ to the empty set before calling Closure(vertexR,X).

Analysis:
Since there are III edges in G and each edge is visited at most one time, the worst case time complexity is O(III).

Example:
The sequence numbers appearing on the edges of figure 2 show the order of graph traversal for computing the closure HOSPITAL$^+$(HCODE). The collected vertices will be {HOSPITAL, UNIT, PARAMED, PATIENT, HOSPDOC}.

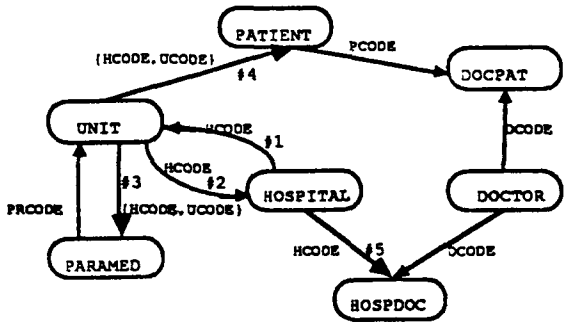

Figure 2

4.2 Implication problem for INDs alone

The IND-graph can be used to deal with the implication problem because the inclusion dependency $S[X] \subseteq R[X]$ holds if and only if $S \in R^+(X)$. However, we can directly search for the particular vertex S as shown in algorithm 3 below. Starting with node R, this algorithm searches G_I using the edges labelled by $Y \supseteq X$ and stops the search otherwise. If S is reached then return True, otherwise return False.
Algorithme 3: Implication of $S[X] \subseteq R[X]$ using G_I
Input: R, S, X, G_I
Output: T or F
Procedure *Implication* (vertexT, X, S)
Begin
If vertexT is not visited **then**
begin
Mark vertexT as visited
If vertexT is labelled S **then** return (T)
For each descendant edge Y such that $X \subseteq Y$ **do**
Implication (DescendantVertex, X, S)
Endif
End
Return (F)
End

Initialize VertexT to vertexR before calling Implication.

Analysis:
The worst case time complexity is the same as that of algorithm 2, i.e. O(III).

Example:

The inclusion dependency PARAMED [HCODE] ⊆ HOSPITAL [HCODE] holds since PARAMED ∈ HOSPITAL$^+$(HCODE). This can be easily verified using the procedure Implication which returns the value TRUE after the fourth call. The following parameters are used for the four calls respectively:
- (VertexHOSPITAL, HCODEH, PARAMED)
- (VertexUNIT, HCODE, PARAMED)
- (VertexHOSPITAL, HCODE, PARAMED)
- (VertexPARAMED, HCODE, PARAMED).

4.3.Minimal cover for a set I of inclusion dependencies

By analogy with FDs, it would be interesting to compute some minimal cover for INDs. By limiting the number of INDs, this reduces the number of edges in the IND-graph, improves the performance of the graph traversal and saves space.

Definition
A minimal cover M_I for a set I of INDs is a set of INDs such that for no $S[X] \subseteq R[X]$ in M_I is M_I - {$S[X] \subseteq R[X]$} equivalent to M_I.

As in the case of FDs, the minimal cover for INDs may not be unique. Since the IND-graph is used in the preceding algorithms, we give an algorithm for building Gm_I, a graph for a minimal cover M_I.

Algorithm 4: Building Gm_I
$Gm_I := G_I$
Begin
For each edge (R, S) labelled by X in G_I **do**
  Begin
    Remove (R, S) from Gm_I
    If Implication(VertexR, X, S) is false **then** replace (R, S) in Gm_I
    EndIf
  End
EndFor
End

Analysis:
Since the procedure Implication is O(|I|) and is called |I| times, once for each iteration of the loop, the worst case time complexity is $O(|I|^2)$.

Example:
Let G_I be the following IND-graph:

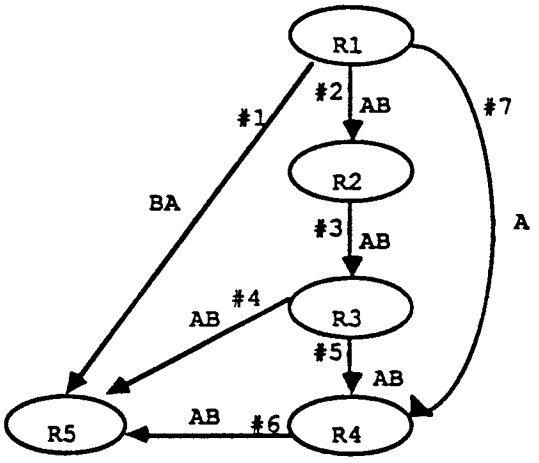

Figure 3

Suppose that the edges are considered in order of the sequence numbers appearing in figure 3. First, the edge #1 will be eliminated because after removing it, Implication(vertexR_1, BA, R_5) will return T. The IND #1 can be obtained from the path corresponding to the INDs #2, #3 and #4 by the permutation and transitivity axioms. Similarly, edges #4 and #7 will be removed. So, the graph Gm_I for a minimal cover related to G_I (figure 3) is the graph G_I from which the edges #1, #4 and #7 are removed.

5.Conclusion

In this paper, a graph structure to deal with the implication problem for typed inclusion dependencies has been presented. The concepts of closure of a relation scheme R for X according to a set of inclusion dependencies and of minimal cover for inclusion dependencies were defined and algorithms for computing them were provided. These ideas can be generalized to less restrictive classes of inclusion dependencies and are the object of our current research.

Acknowledgement
This work was supported in part by NSERC of Canada under grants OGP0041899 and OGP0009184.

Bibliography

[Aho79b] AHO A.V., SAGIV Y. & ULLMAN J. D."Equivalences among relational expressions".*SIAM Journal on Computing*, vol.8, n.2, May 1979, pp.218-246.

[Arms74] ARMSTRONG W.W."Dependency structures of database relationship", *Proc. IFIP'74*, 1974, Geneva, Switzerland, pp.580-583.

[Bern76] BERNSTEIN P.A. "Synthesizing third normal form relations from functional dependencies", *ACM Trans. on Database Systems*, vol.1, no.4, pp.277-298.

[Casa82] CASANOVA M.A., FAGIN R. & PAPADIMITRIOU C.H. "Inclusion dependencies and their interaction with functional dependencies", Proc. *ACM SIGACT-SIGMOD* Conf. On Principles of Database Systems, Los Angeles, Calif. March 29-31 1982, ACM, New York, pp. 171-176.

[Codd70] CODD E.F., "A relational model of data for large shared data banks"; *Communication of ACM*, vol. 13, n. 6, June 1970, pp. 377-387.

[Cosm84] COSMADAKIS S. S., KANELLAKIS P. C., "Functional and inclusion dependencies, a graph theoretic approach", Proceedings *ACM Symposium On Principles of Database Systems*, Waterloo, Ont., 1984, pp. 29-37.

[Date86] DATE C.J., *An introduction to database systems*. Volume I, 4th edition , Addison-Wesley, 1986.

[Fagi77] FAGIN R."Multivalued dependencies and a new normal form for relational databases", *ACM TODS*, vol.2, n.3, pp. 262-278.

[Fagi81] FAGIN R."A normal form for relational databases that is based on domains and keys", *ACM TODS*, vol. 6, 1981, pp. 387-415.

[JoKl82] JOHNSON D.S. & KLUG A.,"Testing containment of conjunctive queries under functional and inclusion dependencies".Proc. ACM *SIGACT-SIGMOD* Conf. On Principles of Database Systems, 1982, pp. 164-169.

[Kane83] KANELLAKIS P.C., COSMADAKIS S. S., VARDI M.Y., "Unary Inclusion Dependencies have Polynomial Time Inference Problems", Proc. 15th *ACM STOC*, 1983.

[KaYo83] KAMBAYASHI Y. & YOSHIKAWA M."Query processing utilizing dependencies and horizontal decomposition". *ACM SIGMOD* Conf., San Jose, May 1983, pp.55-67.

[Miss87] MISSAOUI R, Optimisation des performances d'un système de bases de données relationnel: une approche par système expert. *Thèse de Ph.D.*, Université de Montréal, décembre 1987.

[MiG089b] MISSAOUI R., GODIN R., "Semantic Query Optimization Using Generalized Functional Dependencies and Inclusion Dependencies", Technical Report #98, Département de mathématiques et d'informatique, September 1989, Université du Québec à Montréal, 35 pages.

[Mitc83] MITCHELL J.C.,"Inferences rules for functional and inclusion dependencies".,Proc. *ACM SIGACT-SIGMOD* Conf. On Principles of Database Systems, 1983, pp. 58-69.

[Scio83] SCIORE E. "Inclusion Dependencies and the Universal Instance", Proc. *ACM SIGACT-SIGMOD*, Conf. On Principles of Database Systems, 1983, pp. 48-57.