Journal of Computer and System Sciences **56**, 299–309 (1998)
Article No. SS981570

Universal-Relation Databases

Francesco Mario Malvestuto and Marina Moscarini*

Dipartimento di Scienze dell'Informazione, Via Salaria 113, I-00198 Roma, Italy

Received April 21, 1995; revised October 17, 1997

The method of the canonical connection introduced by Maier and Ullman provides an optimal procedure for query processing in universal-relation databases. We present an algorithm for computing canonical connections in a database scheme which is more efficient than the classical algorithm based on tableau reduction. Moreover, with a slight modification of the algorithm we obtain a join plan which succeeds in controlling the nonmonotonicity of cyclic canonical connections. © 1998 Academic Press

1. INTRODUCTION

In the design of user friendly interfaces to relational databases, the concept of logical independence has been introduced in order to characterize an approach to the problem of query answering in which the user specifies a query in terms of attributes, without being aware of the actual aggregation of attributes in relation schemes. In other words, the user sees the database as constituted of a unique relation aggregating the universe of attributes; for this reason such interfaces are referred to as universal-relation systems. In these systems the queries are processed in two steps:

— a relation having as scheme the set X of the attributes involved in the query is computed; such a relation is called the *window* for X;

— the query is evaluated as if it applied to the above relation.

The window for a given set of attributes is computed by applying to the actual database a relational-algebra (possibly modified to treat null values) expression; a *window function* associates to every set of attributes the corresponding expression. A number of window functions have been proposed in literature (see [11, 18] for a deep discussion of the philosophy, history, and theory of window functions), based on different possible meanings of the universal relation concept. Most initial work on window functions assumes that the database is *globally consistent* [4]; i.e., the relations $r_1, r_2, ..., r_m$ stored in the database are all projections of a single (universal) relation I over the universe of attributes. Under this assumption, known as universal instance assumption or pure universal relation assumption, the answer to a query X is $\pi_X(I)$. Now, if $\{r_1, r_2, ..., r_m\}$ is a globally consistent database with scheme $\{R_1, R_2, ..., R_m\}$, we have that $r_i = \pi_{R_i}(r_1 \bowtie r_2 \bowtie \cdots \bowtie r_m)$, and, hence, $r_1 \bowtie r_2 \bowtie \cdots \bowtie r_m$ is a candidate to be the instance I from which the answer to a query X can be computed by projection on X; consequently, with every set of attributes X the window function associates the expression $\pi_X(R_1 \bowtie R_2 \bowtie \cdots \bowtie R_m)$. Since computing $r_1 \bowtie r_2 \bowtie \cdots \bowtie r_m$ is an expensive task, the following optimization problem arises:

> *Given a database scheme* $\{R_1, R_2, ..., R_m\}$ *and a set of attributes* $X \subseteq \bigcup_{i=1}^{m} R_i$, *find a minimum* (*with respect to set inclusion*) *subset* $\{R_{i_1}, R_{i_2}, ..., R_{i_k}\}$ *of* $\{R_1, R_2, ..., R_m\}$ *such that*
>
> $$\pi_X(R_{i_1} \bowtie R_{i_2} \bowtie \cdots \bowtie R_{i_k}) \equiv \pi_X(R_1 \bowtie R_2 \bowtie \cdots \bowtie R_m), \quad (1)$$
>
> *i.e.,* $\pi_X(r_{i_1} \bowtie r_{i_2} \bowtie \cdots \bowtie r_{i_k}) = \pi_X(r_1 \bowtie r_2 \bowtie \cdots \bowtie r_m)$, *for every globally consistent database* $\{r_1, r_2, ..., r_m\}$ *on* $\{R_1, R_2, ..., R_m\}$.

A subset $\{R_{i_1}, R_{i_2}, ..., R_{i_k}\}$ of $\{R_1, R_2, ..., R_m\}$ that satisfies (1) is here called an *X-subscheme*.

In what follows, since a database scheme can be pictured by a hypergraph (where nodes represent attributes and edges represent relation schemes), we shall make use of a notation borrowed from the hypergraph theory [5] and denote a database scheme and the universe of attributes by H and $V(H)$, respectively. Moreover, we assume that in every database scheme no relation scheme is a subset of another, so that the corresponding hypergraph is reduced [4].

An X-subscheme of H can be characterized by the notion of an X-mapping [12], this being defined as follows. An *X-mapping* of H is a mapping f from H to H such that:

* Corresponding author. E-mail: moscarini@dsi.uniroma1.it.

0022-0000/98 $25.00

(a) for each relation scheme $e \in f(H)$, $f(e) = e$;

(b) if v is an attribute belonging to two or more relation schemes in H, then

— for each relation scheme e containing v, $f(e)$ contains v, or

— for every two relation schemes e and e' containing v, $f(e) = f(e')$;

(c) if a relation scheme e contains an attribute v belonging to X, then v also belongs to $f(e)$.

By making use of results of tableau theory [1, 2, 12], we have that H' is an X-subscheme of H if and only if H' is the image of an X-mapping of H; furthermore, H' is a *minimal* X-subscheme of H if the unique X-mapping of H' is the identity. As suggested in [12], a minimal X-subscheme of H can be computed by resorting to the algorithm for reducing tableaux [1, 2] which runs in $O(m^4 \times n)$ time, where $m = |H|$ and $n = |V(H)|$.

Suppose that a minimal X-subscheme $H' = \{e_{i_1}, e_{i_2}, ..., e_{i_k}\}$ of H has been computed. A further optimization step to compute the answer to the query X can be done by considering the set $\{e'_1, e'_2, ..., e'_k\}$, where e'_h $(h = 1, ..., k)$ is the subset of e_{i_h} obtained by deleting the attributes that are not in X and that in H' appear only in e_{i_h}. In fact, it is well known [18] that

$$\pi_X(e_{i_1} \bowtie e_{i_2} \bowtie \cdots \bowtie e_{i_k}) \equiv \pi_X(\pi_{e'_1}(e_{i_1}) \bowtie \pi_{e'_2}(e_{i_2}) \bowtie \cdots \bowtie \pi_{e'_k}(e_{i_k})) \qquad (2)$$

and that the size of the join in the right-hand side is less or equal to the size of the join in the left-hand side. Let us denote $\{e'_1, e'_2, ..., e'_k\}$ by (H', X). It can be shown [9] that, for every two minimal X-subschemes H' and H'' of H, one has $(H', X) = (H'', X)$. This unique (reduced) hypergraph is called the *canonical connection for X in H*, denoted by $CC(H, X)$ [12].

In this paper we present an algorithm which, given H and X, computes the node set $V(CC(H, X))$ of $CC(H, X)$ in $O(m \times n)$ time, so that, since $CC(H, X)$ is an induced subhypergraph of H (see Lemma 3.6 in [12]), $CC(H, X)$ can be computed in $O(m^2 \times n)$ time in the following way:

— compute the set family H' containing the intersections of all edges of H with $V(CC(H, X))$

— remove a set from H' if it is contained in another set in H'.

In what follows the node set of $CC(H, X)$ will be called the *canonical closure* of X in H. At the basis of our algorithm there is the concept (introduced in [15] and recalled in the next section) of *compaction* of a reduced hypergraph. The compaction of a hypergraph H is a reduced hypergraph, denoted by $\hat{H}$, with the relevant property [15] of being *acyclic*.

Acyclic hypergraphs have been extensively studied in literature and several graph theoretic characterizations of such hypergraphs have been provided [6–8, 12, 17]. Furthermore, it has been shown that relational databases with acyclic schemes enjoy several desirable properties related to consistency checking, dependency equivalence and query optimization [3, 4, 6, 9, 16]. Among these characterizations and properties, the following two will be used in our approach to the query optimization problem in universal-relation databases:

(a) [12] if H is acyclic then $CC(H, X)$ coincides with the *Graham reduction* of H *with X sacred*, this being obtained by recursively applying the following two operations to H until they can be applied no more:

(G1) If an attribute v not in X appears in only one relation scheme, delete v from that relation scheme.

(G2) Delete one relation scheme e from H if there is another relation scheme e' for which $e \subseteq e'$.

(b) [4] H is acyclic if and only if it has a *monotone sequential join expression*, i.e., there exists an ordering $e_1, e_2, ..., e_m$ of the relation schemes in H such that, for every globally consistent database $\{r_1, r_2, ..., r_m\}$ on H, in the evaluation of the expression $((\cdots(e_1 \bowtie e_2) \bowtie \cdots) \bowtie e_m)$ we have that

$$|((\cdots(r_1 \bowtie r_2) \bowtie \cdots) \bowtie r_i)| \leqslant |((\cdots(r_1 \bowtie r_2) \bowtie \cdots) \bowtie r_{i+1})|.$$

Property (a) will be used in our algorithm; in fact, this algorithm consists of two steps: at the first one $CC(\hat{H}, X)$ is computed and at the second one $V(CC(H, X))$ is derived from $CC(\hat{H}, X)$. Due to the acyclicity of $\hat{H}$ the computation of $CC(\hat{H}, X)$ can be done in $O(m \times n)$ time [17]; moreover, as shown later on, the second step requires $O(m \times n)$ time so that the overall complexity of our algorithm is $O(m \times n)$.

In Section 6 we discuss how to exploit property (b) enjoyed by $\hat{H}$ in order to obtain a sequential join expression, equivalent to the right-hand side of (2), that in some sense controls the size growth of the intermediate results due to the existence of cyclic subhypergraphs in $CC(H, X)$.

The paper is organized as follows. In Section 2 we recall basic definitions from the hypergraph theory and introduce the notion of the compaction of a hypergraph. Section 3 contains the proposed algorithm which is proved to correctly compute canonical closures in Section 4, and to run in $O(m \times n)$ time in Section 5. In Section 6 we propose a join plan based on that algorithm which has the merit of controlling the "non-monotonicity" [4] of cyclic canonical connections.

2. BASIC DEFINITIONS AND PRELIMINARY RESULTS

In this section we recall some basic definitions for hypergraphs and some results recently stated by the authors [15], which will be used in the following sections.

According to the terminology used in [5], a *hypergraph* H is a set family; the members of H are called *edges* of H and the elements of edges are called *nodes* of H. The node set of H is denoted by $V(H)$. A hypergraph is *reduced* [4] (or "simple" [5]) if no edge is a subset of another edge. A *partial edge* of hypergraph H is any nonempty subset of some edge of H. The *degree* of a node of H is the number of edges of H containing it. A *path* (or a "connection" [15]) in H from node u to node v is a sequence of edges $(e_1, ..., e_k)$, $k \geqslant 1$, such that $u \in e_1$, $v \in e_k$, and, if $k > 1$, $e_h \cap e_{h+1} \neq \varnothing$ for $h = 1, ..., k-1$. Furthermore, we say that this path *passes through* a subset X of $V(H)$ if $e_h \cap e_{h+1}$ is a subset of X for some $h < k$. H is *connected* if for every two nodes u and v there is a path from u to v. In what follows we assume that H is a reduced and connected hypergraph.

A hypergraph H is *acyclic* if the Graham reduction of H with no sacred nodes is the empty hypergraph. Let H be a hypergraph and X a subset of $V(H)$; by $\langle X \rangle_H$ (we omit H when there is no ambiguity) we denote the *subhypergraph of H induced by X*, that is, the reduced (not necessarily connected) hypergraph with node set X whose edges are the maximal (with respect to set inclusion) members of the set family $\{e \cap X \mid e \in H\}$. An *X-component* of H is a connected component of $\langle V(H) \backslash X \rangle$.

The following definitions and results appear in [15]. In order to illustrate them, in our running example we will refer to the hypergraph H in Fig. 1.

Let H be a hypergraph and X a proper subset of $V(H)$. If C is an X-component of H, the *boundary of C in H*, denoted by $\partial_H C$ (we omit H when there is no ambiguity) is the set of nodes in X that in H are adjacent to some node of C; the *closure of C in H*, denoted by $[C]_H$ (we omit H when there is no ambiguity), is the subhypergraph of H induced by $V(C) \cup \partial_H C$.

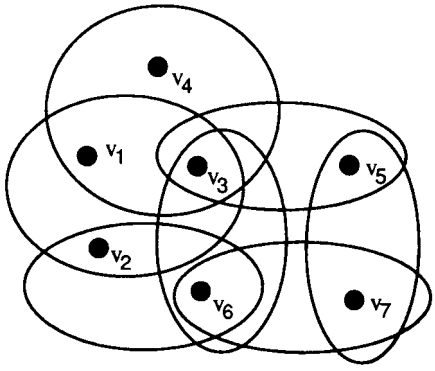

FIG. 1. A hypergraph H.

EXAMPLE. Let $X = \{v_1, v_2, v_3\}$. In Fig. 2 the X-components of H and their closures are shown. ▮

Fact 1 [15, Lemma 1]. Let H be a hypergraph and X a proper subset of $V(H)$. If C is an X-component of H, then ∂C is a subset of X.

Let H be a hypergraph. A proper subset X of $V(H)$ is a *separator* of H if $\langle V(H) \backslash X \rangle$ is not connected. If X is a separator of H, then two nodes of distinct X-components of H are said to be *separated in H by X*. Equivalently, two nodes u and v of H that are not in X are separated by X if they are not adjacent in H and every path from u to v passes through X.

Fact 2 [15, Lemma 5]. Let H be a hypergraph, X a partial edge of H, and C an X-component of H. Two nodes of $[C]$ are separated by Y in H and Y is a partial edge of H, if and only if they are separated by Y in $[C]$ and Y is a partial edge of $[C]$.

Let H be a hypergraph. A *partial-edge separator* of H is a partial edge which is also a separator. A partial-edge separator X of H is a *divider* of H if there exist two nodes of H which are separated by X and by no proper subset of X. We say that two nodes of H are *tightly connected* in H if they are separated in H by no partial edge of H. Moreover, by a *compact* of H we mean a set of pairwise tightly connected nodes of H.

EXAMPLE (Continued). The set of nodes $\{v_1, v_2, v_3\}$ is a partial-edge separator but not a divider of H. The set of nodes $\{v_1, v_3\}$ is a divider of H. The nodes v_3 and v_7 are tightly connected in H. The set of nodes $\{v_3, v_5, v_6\}$ is a compact of H. ▮

Fact 3 [15, Corollary 4]. Let H be a hypergraph, X a proper subset of $V(H)$ and C an X-component of H. If H

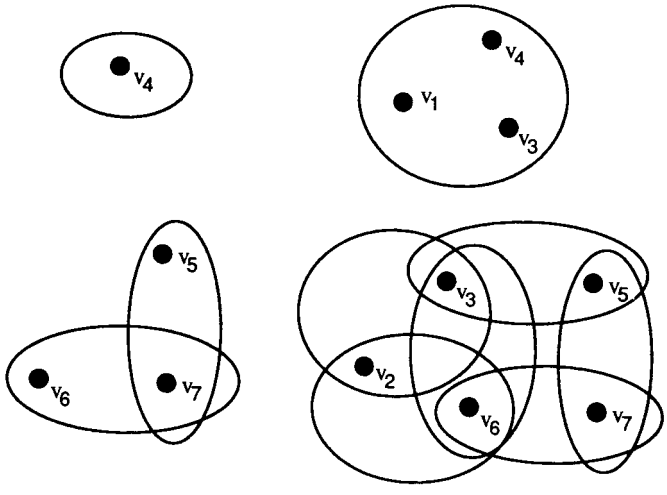

FIG. 2. (Left) The X-components of H. (Right) The closure of the X-components of H.

is acyclic and ∂C is a partial-edge separator of H, then ∂C is a divider of H.

Fact 4 [15, Lemma 9]. Let H and K be two hypergraphs such that each edge of H is a partial edge of K. If each edge of K is a compact of H, then any two nodes that are separated in H by a partial edge X of H are separated by X in K too.

Let H be a hypergraph. The *compaction* of H, denoted by $\hat{H}$, is the hypergraph having as its edges the maximal (with respect to set inclusion) compacts of H.

EXAMPLE (Continued). The compaction of H is shown in Fig. 3. ▮

The compaction of a hypergraph H has a number of nice properties [15] such as:

(i) each edge of H is a partial edge of $\hat{H}$;

(ii) $\hat{H}$ is acyclic;

(iii) H coincides with $\hat{H}$ if and only if H is acyclic;

(iv) every divider of H is a divider of $\hat{H}$, and vice versa.

Finally, the subhypergraphs of a hypergraph H induced by maximal compacts of H (i.e., by edges of $\hat{H}$) are here called the *compact components* of H. A compact component of H is *simple* if it has one edge (that is. if it is induced by one edge of $\hat{H}$ which is also an edge of H), and *compound* otherwise. We will denote by $\mathscr{H}$ the set of the compact components of H.

EXAMPLE (Continued). The compact components of H are shown in Fig. 4. Out of the four compact components of H, two are simple and two are compound. ▮

The notion of "compact components" originates from [13, 14] and is closely related to the database-theoretic notions of an "extension" of a database scheme [16] and of a "hinge-tree" [10].

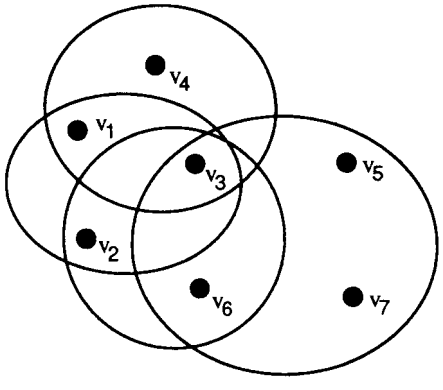

FIG. 3. The compaction of H.

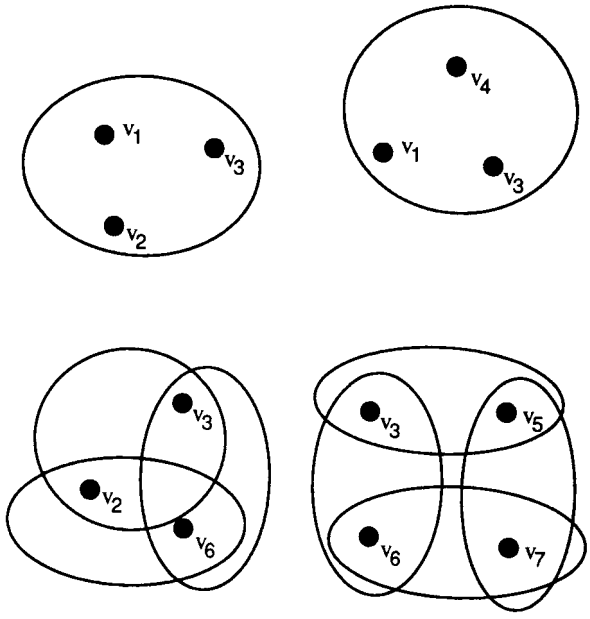

FIG. 4. The compact components of H.

3. THE ALGORITHM

In this section we present an algorithm to compute $V(CC(H, X))$. It starts by performing the Graham reduction of $\hat{H}$ with X sacred; since $\hat{H}$ is acyclic, the result gives the canonical connection for X in $\hat{H}$ [12]. The Graham reduction of an acyclic hypergraph can be computed using the algorithm by Tarjan and Yannakakis [17], which with each edge e^* of $CC(\hat{H}, X)$ associates an edge of $\hat{H}$ containing e^*, we denote by $source(e^*)$.

At this point, the node set of $CC(H, X)$ is computed in the following way. Each edge e^* of $CC(\hat{H}, X)$ is tested for inclusion in some edge of $\langle source(e^*)\rangle_H$; if e^* turns out to be a partial edge of $\langle source(e^*)\rangle_H$, then we add the nodes in e^* to $V(CC(H, X))$; otherwise we add all nodes in $source(e^*)$ to $V(CC(H, X))$.

ALGORITHM Canonical Closure.
input: the compaction $\hat{H}$ of a hypergraph H, the compact components of H, and a subset X of $V(H)$;
output: $V(CC(H, X))$;
begin
Initialization:
$V(CC(H, X)) := \varnothing$;
Step 1:
Compute $CC(\hat{H}, X)$;
Step 2:
for every edge e^* of $CC(\hat{H}, X)$ **do**
begin
if e^* is a partial edge of the compact component $\langle source(e^*)\rangle_H$

then
$V(CC(H, X)) := V(CC(H, X)) \cup e^*$
else $V(CC(H, X)) := V(CC(H, X)) \cup$ *source*(e^*)
end
end

EXAMPLE (Continued). Apply the algorithm above to compute the canonical closure of $X = \{v_1, v_7\}$ in the hypergraph H shown in Fig. 1. The Graham reduction of the hypergraph $\hat{H}$ (shown in Fig. 3) with X sacred is $CC(\hat{H}, X) = \{\{v_1, v_2, v_3\}, \{v_2, v_3, v_6\}, \{v_3, v_6, v_7\}\}$. Notice that $\{v_1, v_2, v_3\}$ and $\{v_2, v_3, v_6\}$ are the sources of themselves, and the edge $\{v_3, v_5, v_6, v_7\}$ of $\hat{H}$ is the source of $\{v_3, v_6, v_7\}$. Now, only the edge $\{v_1, v_2, v_3\}$ of $CC(\hat{H}, X)$ is a partial edge of the compact component of H induced by its source; so, we take

$$V(CC(H, X)) = \{v_1, v_2, v_3\} \cup \{v_2, v_3, v_6\} \cup \{v_3, v_5, v_6, v_7\}. \quad \blacksquare$$

In Sections 4 and 5 we prove, respectively, that our algorithm correctly computes the canonical closure of X in H and runs in $O(m \times n)$. It should be noted that computing $\hat{H}$ and $\mathscr{H}$, which occur in the input of the algorithm, can be done once and for all at the creation time of the database scheme.

4. THE CORRECTNESS

In this section we prove that the algorithm stated in the previous section determines the canonical closure of X in H. First, notice that if H is acyclic, then $\hat{H} = H$ (see property (iii) of $\hat{H}$); thus, the Graham reduction of $\hat{H}$ with X sacred is $CC(H, X)$; moreover, it is easily seen that Step 2 computes the node set of $CC(H, X)$.

To prove the correctness in the general case, we need some preliminary technical lemmas.

LEMMA 1. *Let H be a hypergraph, X a subset of $V(H)$, and C an X-component of H. If C' is an X-component of $\hat{H}$ such that $V(C) \cap V(C') \neq \varnothing$, then $V(C) = V(C')$.*

Proof. Let v be a node in $V(C) \cap V(C')$. Let u be a node in $V(C)$ and let $(e_1, ..., e_k)$ be a path from u to v in C. Since $\bigcup_{h=1}^{k} e_h$ and X are disjoint and every edge of H is a partial edge of $\hat{H}$, u is in the X-component of $\hat{H}$ containing v; therefore $V(C) \subseteq V(C')$.

To prove that $V(C) \supseteq V(C')$, let us assume by contradiction that there is a node w in $V(C') \backslash V(C)$. Since w does not belong to X, the fact that w is not a node of C implies that v and w are separated by X in H and, hence, by Fact 4, they are separated by X in $\hat{H}$; but this contradicts the fact that v and w are in the same X-component of $\hat{H}$. $\blacksquare$

Let H be a hypergraph, X a subset of $V(H)$ and v a node of H; we say that in H a partial edge Y of H *separates v from X* if there exists a Y-component C of H such that v is a node of C and $V(C) \cap X = \varnothing$. In other words, Y separates v from X if v does not belong to Y and, for each u in $X \backslash Y$, u and v are separated by Y. (Notice that, if there is an edge e of H such that $X \subseteq Y \subseteq e$ and v is in $e \backslash Y$, then Y separates v from X in H.)

LEMMA 2. *Let H be a hypergraph, X a subset of $V(H)$ and v a node of H. If v is not a node of $CC(H, X)$, then there exists an edge of $CC(H, X)$ that separates v from X in H.*

Proof. Let v be a node of H but not a node of $CC(H, X)$. Since $X \subseteq V(CC(H, X))$, v is not in X. Let f be an X-mapping of H such that $f(H)$ is a minimal X-subscheme of H. Two cases arise:

(a) v is a node of $f(H)$;

(b) v is not a node of $f(H)$.

Case (a). Since v is a node of $f(H)$ and is not a node of $CC(H, X)$ (which is obtained from $f(H)$ by removing nodes of degree one), v is a node of degree one of $f(H)$ and v does not belong to X. Let e be the edge of $f(H)$ that contains v and let e' be the edge of $CC(H, X)$ obtained from e by eliminating the nodes of degree one that are not in X. Let C be the e'-component of H containing v; we will show that if $u \in V(C)$ then $u \notin V(CC(H, X))$ (from which the disjointness of $V(C)$ and X will follow).

The statement above is trivially true whenever u belongs to e because every node in $V(C) \cap e$ does not belong to X and has degree one in $f(H)$ and, hence, is not a node of $CC(H, X)$. Now, suppose that u belongs to $V(C) \backslash e$; let $(e'_1, ..., e'_k)$ be a path in C from v to u such that $e'_h \cap e = \varnothing$ $(h = 2, ..., k)$ and $e'_h \cap e'_{h'} = \varnothing$ for $|h - h'| \neq 1$ (it is easily seen that such a path always exists). For each h, $1 \leqslant h < k$, let v_h be in $e'_h \cap e'_{h+1}$ (see Fig. 5). Since the edges of C are all partial edges of H, there exists a path $(e_1, ..., e_k)$ in H from v to u such that for each h, $1 \leqslant h < k$, $e'_h \subseteq e_h$. Notice that:

— $e_1 \neq e$ (since v_1 is in e_1 but not in e),

— $f(e) = e$ (by condition (a) in the definition of an X-mapping);

— e_1 is not an edge of $f(H)$ (since v is a node of $f(H)$ of degree one).

It follows from condition (b) in the definition of an X-mapping that $f(e_1) = e$. Now, since v_1 is not in e, by condition (b) in the definition of an X-mapping, one has also that $f(e_2) = e$ and, by repeating similar arguments, $f(e_h) = e$ for $h = 1, ..., k$. Since u is in e_k and not in e, by condition (b) in the definition of an X-mapping every edge containing u is mapped by f to e; therefore, u is not in $V(f(H))$ and, hence, is not a node of $CC(H, X)$).

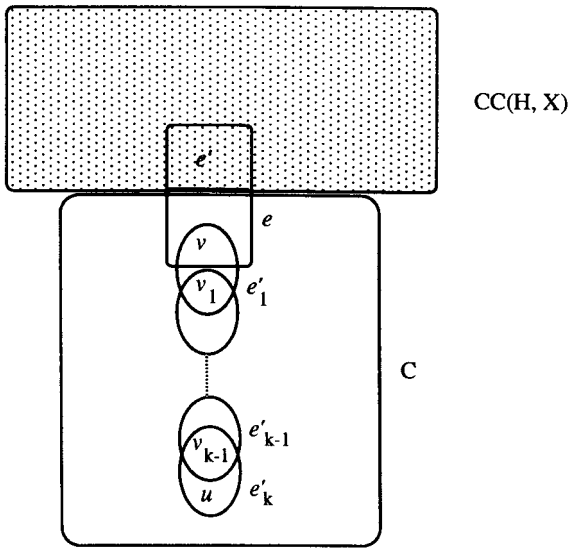

FIGURE 5

Case (b). By condition (b) in the definition of an X-mapping, all edges of H containing v are mapped by f to one edge e of $f(H)$. Let e' be an edge of $CC(H, X)$ containing the nodes in e that in $f(H)$ have degree two or more, and let C be the e'-component of H containing v. It is easy to show with arguments analogous to those employed in Case (a) that no node of C is a node of $CC(H, X)$. ▮

LEMMA 3. *Let H be a hypergraph, X a subset of $V(H)$, and v a node of H. If there exists a partial edge of H that separates v from X in H, then v is not a node of $CC(H, X)$.*

Proof. Let Y be a partial edge of H that separates v from X in H, and let C be the Y-component of H containing v. Let e be an edge of H that includes Y. Consider the mapping from H to H defined as

$$f(e') = \begin{cases} e, & \text{if } e' \text{ is an edge of } [C] \\ e', & \text{otherwise.} \end{cases}$$

We will show that f is an X-mapping of H:

— f satisfies condition (a) in the definition of an X-mapping. In fact, $f(H) = \{e\} \cup \{e' \mid e' \in H \backslash [C]\}$ and, hence, in order to show that $f(e') = e'$ for each edge of $f(H)$ it is sufficient to observe that $f(e) = e$ independently of the membership of e in $[C]$.

— f satisfies condition (b) in the definition of an X-mapping. Let u be in two or more edges of H. If u is a node of $[C]$, then either u is a node of C or u is in ∂C. In the former case, each edge of H containing u is an edge of $[C]$ and, hence, is mapped by f to e; in the latter case, each edge of H that contains u and is also an edge of $[C]$, is mapped by f to e (which contains u because, by Fact 1, $\partial C \subseteq Y \subseteq e$), and each edge of H that contains u and is not an edge of $[C]$, is mapped by f to itself (that is, to an edge containing u).

— f satisfies condition (c) in the definition of an X-mapping. Let e' be an edge of H containing a node v in X. If e' is an edge of $[C]$, then $e' \cap (V(C) \cap X) = \varnothing$ since $V(C) \cap X = \varnothing$; consequently, v is in ∂C and, hence, in Y (by Fact 1) and in e; now, since $f(e') = e$, $f(e')$ contains v.

Thus, f is an X-mapping of H. Now, let f' be an X-mapping of H such that $f'(H)$ is a minimal X-subscheme of H and $f'(H) \subseteq f(H)$. If v is not a node of $f(H)$ then v cannot be a node of $f'(H)$ and, hence, of $CC(H, X)$. If v is a node of $f(H)$ then, since each edge of H containing v is also an edge of $[C]$ and each edge of $[C]$ is mapped by f to e, e is the unique edge $f(H)$ containing v. Consequently, v is a node of $f(H)$ of degree one and, hence, cannot be a node of $CC(H, X)$. ▮

We now prove the correctness of our algorithm.

THEOREM 1. *Let H be a hypergraph and X a subset of $V(H)$. The algorithm Canonical Closure correctly computes $V(CC(H, X))$.*

Proof. Let Y be the output of the algorithm with input H and X. We show that $Y = V(CC(H, X))$.

We first show that $Y \subseteq V(CC(H, X))$. Let $v \notin V(CC(H, X))$. We shall prove that $v \notin Y$. By Lemma 2, there is an edge e' of $CC(H, X)$ that separates v from X in H. Let C be the e'-component of H containing v. Since e' is an edge of $CC(H, X)$, e' is a partial edge of H and, hence, of $\hat{H}$. Let C' be the e'-component of $\hat{H}$ containing v; by Lemma 1, $V(C) = V(C')$, and hence, e' separates v from X in $\hat{H}$. From Lemma 3 it follows that v is not a node of $CC(\hat{H}, X)$ and, since $Y \subseteq V(CC(\hat{H}, X))$ (see Step 2 of the algorithm), v cannot be a node of $\langle Y \rangle_H$.

We now show that $Y \supseteq V(CC(H, X))$. Let $v \notin Y$. We shall prove that $v \notin V(CC(H, X))$. The assumption that v does not belong to Y implies that:

(1) v is deleted during the execution of Step 1, and

(2) v is not added during the execution of Step 2.

From (1) and Lemma 2 it follows that there is an edge e' of $CC(\hat{H}, X)$ that separates v from X in $\hat{H}$. Let C' be the e'-component of $\hat{H}$ containing v.

Let us non distinguish two cases:

(i) There is no edge $\hat{e}$ of $\hat{H}$ such that both $v \in \hat{e}$ and $e' \subseteq \hat{e}$. Let $\hat{e}$ be an edge of $\hat{H}$ such that $e' \subseteq \hat{e}$. Let C'' be the $\hat{e}$-component of $\hat{H}$ containing v. It is obvious that $V(C'') \subseteq V(C')$ and, hence, since $V(C') \cap X = \varnothing$, also $V(C'') \cap X = \varnothing$. Therefore, $\partial C''$ separates v from X in $\hat{H}$. Now, since $\hat{H}$ is acyclic, by Fact 3, $\partial C''$ is a divider of $\hat{H}$, and, hence, by property (iv) of $\hat{H}$, $\partial C''$ is also a divider of H. Moreover, if

C is the $\hat{e}$-component of H containing v, by Lemma 1, $V(C) = V(C'')$. Hence, $\partial C''$ is a partial edge of H that separates v from X in H. By Lemma 3, v cannot be a node of $CC(H, X)$.

(ii) There is an edge $\hat{e}$ of $\hat{H}$ such that both $v \in \hat{e}$ and $e' \subseteq \hat{e}$. From (2) it follows that e' is a partial edge of the compact component of H induced by $\hat{e}$ (in fact, otherwise, at Step 2 v, which is in $\hat{e}$, would be added to Y, contradicting the hypothesis that $v \notin Y$). Let C be the e'-component of H containing v; by Lemma 1, $V(C) = V(C')$, and hence, e' is a partial edge of H separating v from X in H. Hence, by Lemma 3, v cannot be a node of $CC(H, X)$. ▮

5. THE COMPLEXITY

In this section we show that the algorithm in Section 3 computes canonical closures in a hypergraph with m edges and n nodes in $O(m \times n)$ time.

The input of the algorithm is given by the compaction $\hat{H}$ of H, the set $\mathscr{H}$ of the compact components of H, and a subset X of $V(H)$. In order to make the test at Step 2 efficient we represent both $\hat{H}$ and $\mathscr{H}$ in one data structure, called the structural graph of H, relating the edges of $\hat{H}$ with the corresponding compact components of H.

Let K be the set of edges of the compact components of H, i.e $K = \bigcup_{H' \in \mathscr{H}} H'$. The *structural graph* G_H of H is the bipartite graph with vertex set $\hat{H} \cup K$, where an edge $\hat{e}$ of $\hat{H}$ is linked to each edge of the compact component of H induced by $\hat{e}$. Given a family of hypergraphs $\mathscr{K}$, let us denote by $\|\mathscr{K}\|$ the quantity $\sum_{H' \in \mathscr{K}} |H'|$. Then, G_H has $\|\mathscr{H}\|$ edges and at most $\|\mathscr{H}\| + |\hat{H}|$ vertices. Furthermore, $\|\mathscr{H}\| = |H|$ if H contains no partial-edge separators or H is acyclic.

EXAMPLE (Continued). Consider again the hypergraph shown in Fig. 1 whose compaction and compact components are shown in Figs. 3 and 4. The structural graph of H is shown in Fig. 6. Note that $\|\mathscr{H}\| = 9$. ▮

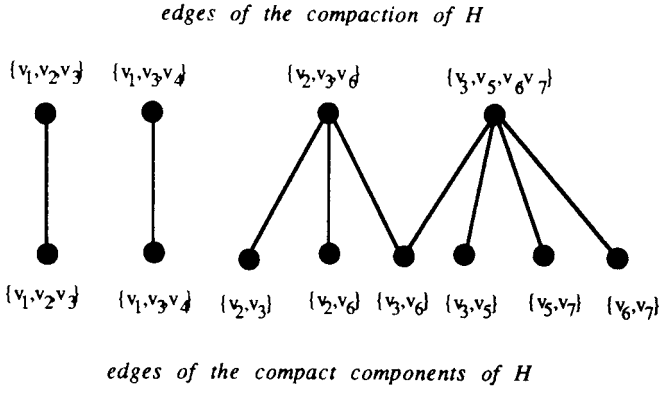

FIG. 6. The structural graph of H.

Let us, now, examine the complexity of the two steps of the algorithm.

Step 1 computes the Graham reduction of $\hat{H}$ and sources of its edges which can be done in $O(|\hat{H}| \times n)$ time [17].

At Step 2, for each edge e^* of $CC(\hat{H}, X)$ we have to test whether e^* is or is not a partial edge of the compact component of H induced by $source(e^*)$. To decide if this is or is not the case, we compare e^* with each neighbour of $source(e^*)$ in the structural graph of H. This requires a number of set comparisons not greater than the number of edges of the structural graph of H and, hence, a time $O(\|\mathscr{H}\| \times n)$. Since $\|\mathscr{H}\| \geqslant |\hat{H}|$, the algorithm Canonical Closure can be implemented in $O(\|\mathscr{H}\| \times n)$. Now, as is shown below (see Lemma 7), $\|\mathscr{H}\|$ is $O(m)$ and, therefore, our algorithm runs in time $O(m \times n)$. To prove that $\|\mathscr{H}\|$ is $O(m)$, we refer to the following algorithm, which, given a hypergraph H, correctly computes $\mathscr{H}$ and, hence $\hat{H}$ (see Theorem 10 in [15]).

ALGORITHM Compact.

```
Input: H;
Output ℋ;
  begin
    ℋ := {H}
    for every e ∈ H do
      begin
        let H_e be the hypergraph in ℋ containing e;
        if e is a separator of H_e or contains a node of H_e
        having degree one
          then
            begin
              ℋ := ℋ\{H_e} ∪ {{e}};
              for every e-component C of H_e do
                begin
                  ℋ := ℋ ∪ {[C]_{H_e}};
                  if e is a subset of V([C]_{H_e}) then
                    ℋ := ℋ\{{e}}
                end
            end
      end
  end
```

The way the algorithm Compact succeeds in determining the set $\mathscr{H}$ of the compact components of H can be pictured with a tree (*decomposition tree*). Let $e_1, ..., e_m$ be any ordering of edges of H; the decomposition tree T is defined as follows:

— the root of T is H;

— the nonleaf vertices of T are the hypergraphs H_{e_i}, $i = 1, ..., m$; a child of H_{e_i} is either $\{e_i\}$ or the closure of an e_i-component of H_{e_i};

— the leaves of T are the compact components of H.

EXAMPLE (Continued). Let us consider again the hypergraph H of Fig. 1 with the following ordering of its edges:

$e_1 = \{v_3, v_6\}$, $e_2 = \{v_2, v_6\}$, $e_3 = \{v_1, v_2, v_3\}$, $e_4 = \{v_1, v_3, v_4\}$, $e_5 = \{v_3, v_5\}$, $e_6 = \{v_6, v_7\}$, and $e_7 = \{v_5, v_7\}$. The decomposition tree is shown in Fig. 7. Here H_{e_1} is H (see Fig. 1). $H_{e_2} = H_{e_3}$ is shown in Fig. 8, $H_{e_5} = H_{e_6} = H_{e_7} = H''$ is shown in Fig. 9. H' is shown in Fig. 10. Notice that the leaves of the decomposition tree (i.e., $\{e_3\}$, $\{e_4\}$, H', and H'') are the compact components of H (see Fig. 4). ▮

The set of hypergraphs resulting from the decomposition of H after examining the first i edges of H (according to the algorithm Compact) will be denoted by $\mathscr{H}_i$. Accordingly, $\mathscr{H}_m = \mathscr{H}$. Moreover, we denote by $\mathscr{C}_i$ the set of hypergraphs that in the decomposition tree of H are children of H_{e_i}.

Remark. Each edge of H_{e_i} distinct from e_i is an edge of exactly one child of H_{e_i}; therefore the contribution to $\|\mathscr{C}_i\|$ of the edges of H_{e_i} distinct from e_i is equal to $|H_{e_i}| - 1$. Moreover, the contribution of e_i to $\|\mathscr{C}_i\|$ is at most equal to $|\mathscr{C}_i|$, since for every e_i-component C of H_{e_i} ∂C is a subset of e_i and may be an edge of the closure of C (i.e., of a child of H_{e_i}). To sum up, one has

$$\|\mathscr{C}_i\| \leqslant |\mathscr{C}_i| + |H_{e_i}| - 1$$

LEMMA 4. *Let $\mathscr{H}_i$, $1 \leqslant i \leqslant |H|$, be the set of hypergraphs resulting from the decomposition of H after examining the first i edges of H. Then, for all i*

$$\|\mathscr{H}_i\| \leqslant |\mathscr{H}_i| + |H| - 1.$$

Proof. We prove the statement by induction on *i*.

Basis ($i = 1$). Since $H_{e_1} = H$ and $\mathscr{H}_1 = \mathscr{C}_1$ the statement follows from the above remark.

Inductive Step. Suppose that the statement is true for $i < |H|$. We shall prove that it also holds for $i + 1$. It is clear that

$$\mathscr{H}_{i+1} = (\mathscr{H}_i - \{H_{e_{i+1}}\}) \cup \mathscr{C}_{i+1}. \tag{4}$$

Therefore, one has

$$\|\mathscr{H}_{i+1}\| = \|\mathscr{H}_i\| - |H_{e_{i+1}}| + \|\mathscr{C}_{i+1}\|$$

which, by the remark, becomes

$$\|\mathscr{H}_{i+1}\| \leqslant \|\mathscr{H}_i\| + |\mathscr{C}_{i+1}| - 1.$$

From the inductive hypothesis it then follows that

$$\|\mathscr{H}_{i+1}\| \leqslant (|\mathscr{H}_i| + |H| - 1) + |\mathscr{C}_{i+1}| - 1$$

and, finally, since by (4) $|\mathscr{H}_{i+1}| = |\mathscr{H}_i| - 1 + |\mathscr{C}_{i+1}|$, we obtain

$$\|\mathscr{H}_{i+1}\| \leqslant |\mathscr{H}_{i+1}| + |H| - 1. \quad ▮$$

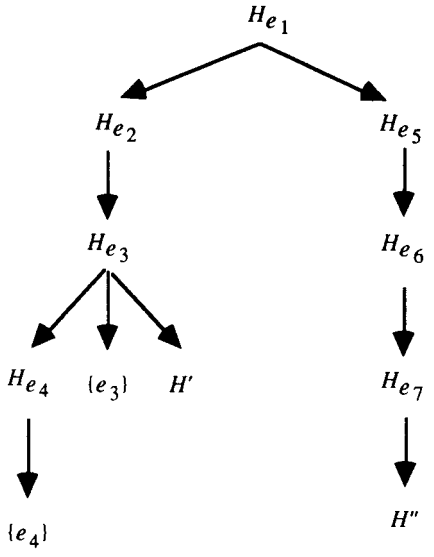

FIGURE 7

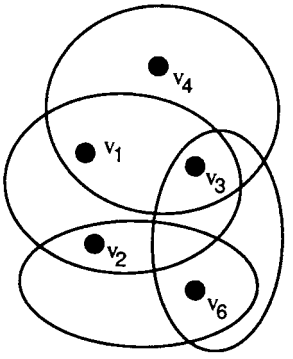

FIGURE 8

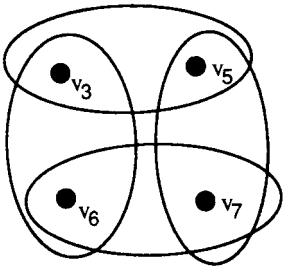

FIGURE 9

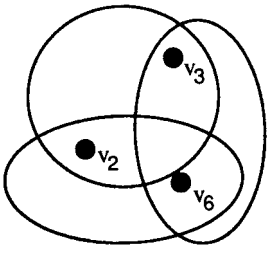

FIGURE 10

Since $\mathscr{H}_m = \mathscr{H}$ it follows that

COROLLARY 1. *Let $\mathscr{H}$ be the set of compact components of hypergraph H. Then, $\|\mathscr{H}\| \leqslant |\mathscr{H}| + |H| - 1$.*

To prove that $\|\mathscr{H}\|$ is $O(m)$, by Corollary 1 it is sufficient to show that $|\mathscr{H}|$ is $O(m)$. To achieve this (see Lemma 7 below), we need a few preliminary technical results.

LEMMA 5. *Let H be a hypergraph, e and edge of H that includes a separator of H, and e' an edge of H that contains no separators of H. Let $\mathscr{C}$ be the hypergraph family composed of the closures of all e-components of H. Then e' is an edge of exactly one hypergraph H' in $\mathscr{C}$ and $|H'| \geqslant 3$.*

Proof. Let v be a node belonging to $e' \backslash e$ and C an e-component of H containing v; obviously, e' is an edge of C and of no other e-component of H. Moreover, since e' contains no separators of H, $[C] \supset \{e'\}$ and, by Fact 2, e' contains no separators of $[C]$. Therefore, e' necessarily has a nonempty intersection with at least two edges of $[C]$. ∎

LEMMA 6. *Every compound compact component of a hypergraph has three or more edges.*

Proof. Let $(e_1, ..., e_m)$ be any ordering of edges of H. By Fact 2, one has:

(a) for all i and j such that $1 \leqslant j < i \leqslant m$, if e' is an edge of H_{e_i} and e' is a subset of e_j, then e' contains no separators of H_{e_i}.

Let H' be a compound compact component of H and let H_{e_i} be the parent of H' in a decomposition tree T of H. Since H' is a compound compact component of H, H' is the closure of an e_i-component of H_{e_i} and contains at least one edge e' distinct from e_i. Since H' is a leaf of T, for no j, $i \leqslant j \leqslant m$, $H' = H_{e_j}$; hence, if e_j is an edge of H containing e', then $j < i$ and, by (a), e' contains no separators of H_{e_i}. By Lemma 5, H' contains three or more edges. ∎

COROLLARY 2. *Let $\mathscr{K}$ be the set of compound compact components of a hypergraph. Then, $\|\mathscr{K}\| \geqslant 3\,|\mathscr{K}|$.*

Proof. By Lemma 6. ∎

LEMMA 7. *Let H be a hypergraph with m edges and $\mathscr{H}$ the set of compact components of H. Then $|\mathscr{H}| \leqslant (3m-1)/2$ and $\|\mathscr{H}\| \leqslant (5m-3)/2$.*

Proof. Let $\mathscr{S}$ and $\mathscr{K}$ be respectively the sets of simple and compound compact components of H. First of all, observe that, by the very definition of $\mathscr{S}$, one has $\|\mathscr{S}\| = |\mathscr{S}|$ and

$$\|\mathscr{H}\| = \|\mathscr{S}\| + \|\mathscr{K}\| = |\mathscr{S}| + \|\mathscr{K}\|. \tag{5}$$

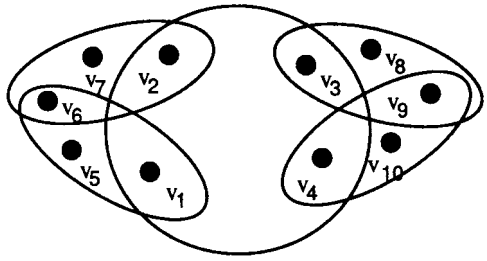

FIGURE 11.

On the other hand, by Corollary 1, one has $\|\mathscr{H}\| \leqslant |\mathscr{H}| + m - 1$ and, since $|\mathscr{H}| = |\mathscr{S}| + |\mathscr{K}|$, one has

$$\|\mathscr{H}\| \leqslant |\mathscr{S}| + |\mathscr{K}| + m - 1. \tag{6}$$

By combining (5) and (6) we obtain

$$\|\mathscr{K}\| \leqslant |\mathscr{K}| + m - 1. \tag{7}$$

On the other hand, by Corollary 2, $\|\mathscr{K}\| \geqslant 3\,|\mathscr{K}|$ which, together with (7), gives

$$|\mathscr{K}| \leqslant (m-1)/2. \tag{8}$$

Finally, by the very definition of $\mathscr{S}$ one has $|\mathscr{S}| \leqslant m$; so, by (8), one has

$$|\mathscr{H}| = |\mathscr{S}| + |\mathscr{K}| \leqslant m + (m-1)/2 = (3m-1)/2$$

and, by Corollary 1,

$$\|\mathscr{H}\| \leqslant [(3m-1)/2] + m - 1 = (5m-3)/2. \quad ∎$$

The hypergraph H shown in Fig. 11 proves that the above upper bounds on $|\mathscr{H}|$ and $\|\mathscr{H}\|$ are tight. In fact, $m = 5$, $|\mathscr{H}| = 7 = (3m-1)/2$, and $\|\mathscr{H}\| = 11 = (5m-3)/2$.

THEOREM 2. *Given a hypergraph H, a subset X of $V(H)$, and the compact components of H, the canonical closure of X in H can be computed in time $O(m \times n)$.*

Proof. By Theorem 1 and Lemma 7. ∎

6. A JOIN PLAN

In this section we show how to exploit the acyclicity of the compaction of a database scheme in order to produce an efficient project-join expression to answer a query.

Let H be a database scheme and X a set of attributes. By applying the algorithm Canonical Closure we determine $Y = V(CC(H, X))$.

Consider the subhypergraph $\langle Y \rangle_{\hat{H}}$. Since $\hat{H}$ is acyclic, $\langle Y \rangle_H$ is acyclic too. Let $((\cdots(\hat{e}_1 \bowtie \hat{e}_2) \bowtie \cdots) \bowtie \hat{e}_p)$ be a monotone, sequential join expression for $\langle Y \rangle_{\hat{H}}$. With every

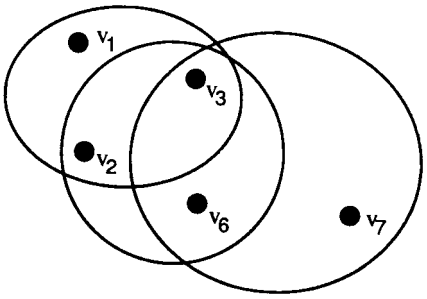

FIGURE 12

edge e of $CC(H, X)$ let us associate a label $l(e) = \min(i \mid e \subseteq \hat{e}_i)$. The expression to be used to answer the query on X is obtained from the expression $\pi_X((\cdots(\hat{e}_1 \bowtie \hat{e}_2) \bowtie \cdots) \bowtie \hat{e}_p)$ by substituting each $\hat{e}_i$ with $\pi_{e'_i}(E_i)$, where

$$e'_i = \hat{e}_i \cap V(CC(\hat{H}, X))$$

and

$$E_i = \bowtie e,$$

the join being extended over all edges e of $CC(H, X)$ such that $l(e) = i$.

The merit of the resulting join expression rests on the following fact. Nonmonotonicity may occur in the evaluation of E_i whenever the set of edges having label i is a cyclic hypergraph. However, if by r_i we denote the relation resulting from evaluating the expression $(\cdots((\pi_{e'_1}(E_1) \bowtie \pi_{e'_2}(E_2)) \bowtie \cdots) \bowtie \pi_{e'_i}(E_i))$, $1 \leqslant i \leqslant p$, then we have that the cardinality of r_i is not greater than the cardinality of r_{i+1} and, hence, of r_p, from which the answer to the query is obtained by projection onto X. This is because $\langle Y \rangle_{\hat{H}}$ is acyclic and $((\cdots(\hat{e}_1 \bowtie \hat{e}_2) \bowtie \cdots) \bowtie \hat{e}_p)$ is a monotone, sequential join expression for $\langle Y \rangle_{\hat{H}}$.

EXAMPLE (Continued). To answer the query on $X = \{v_1, v_7\}$, we start computing $CC(\hat{H}, X)$ (see Fig. 12) and $CC(H, X)$ (see Fig. 13). The subhypergraph of $\hat{H}$

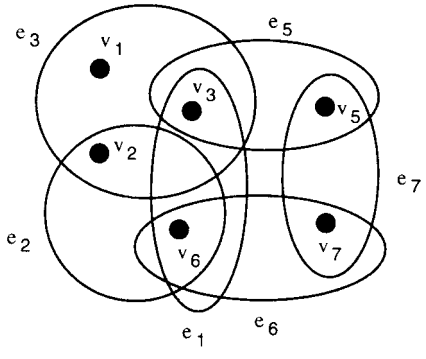

FIGURE 13

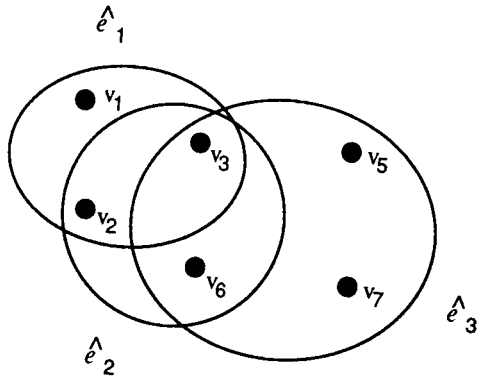

FIGURE 14

induced by the node set of $CC(H, X)$ is shown in Fig. 14; $(\hat{e}_1 \bowtie \hat{e}_2) \bowtie \hat{e}_3$ is a monotone, sequential join expression for $\langle V(CC(H, X)) \rangle_{\hat{H}}$ so that the labels associated to the edges of $CC(H, X)$ are

$$l(e_1) = l(e_2) = 2, \qquad l(e_3) = 1, \qquad l(e_5) = l(e_6) = l(e_7) = 3.$$

Furthermore, we have

$$e'_1 = \hat{e}_1, \qquad e'_2 = \hat{e}_2, \qquad e'_3 = \{v_3, v_6, v_7\}.$$

Therefore, the answer to the query on X is computed by the expression:

$$\pi_X((e_3 \bowtie (e_1 \bowtie e_2)) \bowtie \pi_{e'_3}(e_5 \bowtie e_6 \bowtie e_7)). \quad \blacksquare$$

REFERENCES

1. A. V. Aho, Y. Sagiv, and J. D. Ullman, Efficient optimization of a class of relational expressions, *ACM-TODS* **4** (1979), 435–454.
2. A. V. Aho, Y. Sagiv, and J. D. Ullman, Equivalences among relational expressions, *SIAM. J. Comput.* **8** (1979), 218–246.
3. G. Ausiello, A. D'Atri, and M. Moscarini, Chordality properties on graphs and minimal conceptual connections in semantic data models, *J. Comput. System Sci.* **33** (1986), 179–202.
4. C. Beeri, R. Fagin, D. Maier, and M. Yannakakis, On the desirability of acyclic database schemes, *J. ACM* **3** (1983), 479–513.
5. C. Berge, "Hypergraphs," North-Holland, Amsterdam, 1989.
6. A. D'Atri and M. Moscarini, Recognition algorithms and design methodology for acyclic database schemes, *in* "Advances in Computing Research" (P. Kanellakis and F. Preparata, Eds.), Vol. 3, JAI Press, Greenwich, CT, 1986.
7. A. D'Atri and M. Moscarini, On hypergraph acyclicity and graph chordality, *Inf. Process. Lett.* **29** (1988), 271–274.
8. R. Fagin, Degrees of acyclicity for hypergraphs and relational database schemes, *J. ACM* **3** (1983), 514–550.

9. N. Goodman, O. Shmueli, and Y. C. Tay, GYO reductions, canonical connections, tree and cyclic schemas, and tree projections, *J. Comput. System Sci.* **29** (1984), 338–358.
10. M. Gyssens, P. G. Jeavons, and D. A. Cohen, Decomposing constraint satisfaction problems using database techniques, *Artif. Intell.* **66** (1994), 57–89.
11. D. Maier, D. Rozenshtein, and D. S. Warren, Window functions, *in* "Advances in Computing Research" (P. Kanellakis and F. Preparata, Eds.), Vol. 3, JAI Press, Greenwich, CT, 1986.
12. D. Maier and J. D. Ullman, Connections in acyclic hypergraphs, *Theor. Comput. Sci.* **32** (1984), 185–199.
13. F. M. Malvestuto, Answering queries in categorical databases, *in* "Proc. of the VI ACM Symp. on Principles of Database Systems, 1987," pp. 82–89.
14. F. M. Malvestuto, A universal table model for categorical databases, *Inform. Sci.* **49** (1989), 203–223.
15. F. M. Malvestuto and M. Moscarini, "Separability of Hypergraphs by Partial Edges," Research Report SI/05 Dipartimento di Scienze dell'Informazione, I University of Rome, 1994.
16. Y. Sagiv and O. Shmueli, Solving queries by tree projections, *ACM-TODS* **18** (1993), 487–511.
17. R. E. Tarjan and M. Yannakakis, Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs, *SIAM J. Comput.* **13** (1984), 566–591.
18. J. D. Ullman, "Principles of Database and Knowledge-Base Systems," Vol. II, The New Technologies, Computer Science Press, Rockville, MD, 1989.