The Information Hypergraph Theory

Jianfang Wang

（信息超图理论）

SCIENCE PRESS
Beijing

The Information Hypergraph Theory

Jianfang Wang

（信息超图理论）

SCIENCE PRESS
Beijjng

Responsible Editor: Zhao Yanchao

Published by Science Press
16 Donghuangchenggen North Street
Beijing 100717, China

Printed in Beijing

ISBN 978-7-03-020651-0 (Beijing)

Preface

In the past years, all concepts in graphs were translated to hypergraphs, in special, the definition of cycles in hypergraphs is the same as in graphs. As extension of graphs, there are many results on trees, cycles, covering and coloring of hypergraphs.

In early 1980s, with the emergence of informational science, the information scientists introduced decomposition join approach into the design and study of databases with large sizes, while Tony T Lee pointed out "the core issue in the theory of database dependences is information preserving decomposition of a relation into several subrelations in such a way that the original relation can be regenerated by join operation". A decomposition of a relation induces a database scheme, that is a hypergraph on the attributer set. Beeri, Fagin, Maier and Yarmakakis pointed out "Berge-acyclicity is too restrictive an assumption to make about database schemes", and introduced a new definition of acyclic hypergraphs. We call the definition the acyclic-axiom of hypergraphs. Information scientists have proved that acyclic hypergraphs defined by the acyclic-axiom are very useful in database theory. Ullman said "there are many equivalent definitions of acyclic hypergraphs in the sense used in database theory; unfortunately, the notion, while intuitively appealing, differs from that used by graph theorists". I and my students investigated the acyclic hypergraphs and obtained some theoretical results. These results show that the acyclic-axiom of hypergraphs is reasonable and scientific.

The cycle-structure of hypergraphs plays an important role in the hypergraph theory and the database theory. A cycle of a hypergraph is not only a local structure, but also has relations with the whole structure of the hypergraph. In the cycle-axiom of hypergraphs, we found not only the local structure of a cycle were characterized, but also the inter-relations displayed between it and other parts of the hypergraph. This is a new idea and modular in discrete mathematics. In algebra, there are similar modulars, for example, normal subgroups of groups, ideals of rings, which are very important. Further more, the ideal theory of polynomial rings $F(x_1, x_2, \cdots, x_n)$ of the field F is very essential in algebraical geometry. Similarly, the cycle-axiom constitutes the foundation of hypergraph theory, and it will develope discrete mathematics to a new stage. It is similar to Galois's Theory in algebra, the modern algebra was developed from Galois's Theory.

The cycle structure of hypergraph is very complex and the cycle-axiom of hypergraphs displays the most essential role of hypergraphs. Cycles of a hypergraph need to divide into two classes: pseudo-cycles and essential cycles. The essential cyclomatic numbers of hypergraphs are given and the parameter is not monotone for edges in hypergraphs. It undergoes a long process to find the cycle-axiom of hypergraphs. In a few words, acyclic hypergraphs arise in the study of relational database schemes, cycles of hypergraphs arise in the study of acyclic hypergraphs.

A new hypergraph theory will be formed. A cycle of a graph will be a special degenerate case. I would like to collect the distributed papers to form a systemetic book, which is the first book in the field and contains some new results have not published before.

This book consists of ten chapters. The first three chapters are preparatory knowledge, the last seven chapters form the main parts of the book. This is a new theoretical system of hypergraphs. The cycle-axiom constitutes foundation of the theory.

It gives me a great pleasure to express gratitude to Professor Tony T Lee for his help with co-operation to do research on this topic. I am grateful to Doctors Haizhu Li, Guiying Yan, Baoguang Xu and Jiri Mutu for their help in a variety of ways. Specially thank to Dr. Baoguang Xu for his carefully reading manuscript improvement of original draft.

Specially, I express my great gratitude to Institute of Applied Mathematics, Chinese Academy of Sciences, and Center of Graphs, Combinatories and Networks in Academy of Mathematics and Systems Science National Nature Science Foundation (No. 10671199) for aiding finacially to this book.

Jianfang Wang
October 2007

Contents

Chapter 1

Basic Terminologies

Definition 1.1 *Let V be a finite set, $\mathcal{E}$ a set of subsets of V. We call $\mathcal{E}$ to be a hypergraph on V, if $e \neq \varnothing$ for any $e \in \mathcal{E}$ and $\bigcup_{e\in\mathcal{E}} e = V$. The elements of V and the elements of $\mathcal{E}$ are called vertices and edges respectively.*

If $|e| = k$ for any $e \in \mathcal{E}$, then $\mathcal{E}$ is called a k-uniform hypergraph. If $k = 2$, then $\mathcal{E}$ is a graph.

Definition 1.2 *A hypergraph $\mathcal{E}$ is called to be irreducible, or simple, if for any pair of different edges e and e', $e \setminus e' \neq \varnothing$.*

In the following all hypergraphs are irreducible except exposition.

Let $\mathcal{E}$ and $\mathcal{E}'$ be two hypergraphs on V. If $\mathcal{E}' \subset \mathcal{E}$, then $\mathcal{E}'$ is called a partial hypergraph of $\mathcal{E}$. Let $\mathcal{E}$ be a hypergraph on V, $S \subset V$, $\mathcal{E}[S] = \{e \in \mathcal{E} : e \subseteq S\}$ is called the induced partial hypergraph of $\mathcal{E}$ induced by S.

Set $V^{(k)} = \{S \subseteq V : |S| = k\}$.

Let $\mathcal{E}$ be a hypergraph on V, $\mathcal{E}_{(k)} = \{S \in V^{(k)} : S \subseteq e \text{ for some } e \in \mathcal{E}\}$.

For example: $V = \{a, b, c, d, e\}, \mathcal{E} = \{abcd, abce, bcde\}$, $V^{(3)} = \{abc, abd, abe, acd, ace, ade, bcd, bce, bde, cde\}$, $\mathcal{E}_{(3)} = \{abc, abe, acd, ace, bcd, bde, abd, acd, cde\}$.

Definition 1.3 *A hypergraph $\mathcal{E}$ is called to be conformal if for any clique φ of $\mathcal{E}_{(2)}$, there exists $e \in \mathcal{E}$ such that $Q \subseteq e$.*

Definition 1.4 *Let $\mathcal{E}$ be a hypergraph on V. $\bar{\mathcal{E}} = \{\bar{e} = V \setminus e : e \in \mathcal{E}\}$ is called the adjoint hypergraph of $\mathcal{E}$.*

For example: $V = \{a, b, c, d, e, f, g\}$, $\mathcal{E} = \{abc, cde, def, efg\}$, $\bar{\mathcal{E}} = \{defg, abfg, abcg, abcd\}$.

Let $\mathcal{E}$ be a hypergraph on V. A vertex $x \in V$ is called to be isolated if it only belongs to one edge of $\mathcal{E}$. An edge e is called an ear of $\mathcal{E}$ if there exists another edge e' such that every vertex of $e \setminus e'$ is isolated. If for any edge $e' \in \mathcal{E} \setminus \{e\}$, $e \cap e' = \varnothing$, then e is also an ear.

For example: $\mathcal{E} = \{abc, cde, efa, ace\}$, b, d, f are isolated vertices of $\mathcal{E}$, abc, cde, efa are ears of $\mathcal{E}$.

The information scientists introduced the concept of acyclic hypergraphs[3,28,29,41].

Acyclic-axiom A hypergraph is acyclic if it can be reduced to an empty set by repeatedly using the following two operations:

GR_1: delete x if x is an isolated vertex;

GR_2: delete e_i if there is an edge e_j, such that $e_i \subseteq e_j$, for any $i \neq j$.

An equivalent form: $\mathcal{E}$ is an acyclic hypergraph if it can be reduced to an empty set by removing repeatedly ears.

Graham reduction Let $\mathcal{E}$ be a hypergraph. The Graham reduction of $\mathcal{E}$ is obtained from $\mathcal{E}$ by using repeatedly the operation GR_1 and GR_2 for $\mathcal{E}$, or by using ear removing unlit impossible.

Equivalently, let $\mathcal{E}$ be a hypergraph, the hypergraph $\mathcal{E}'$ obtained from $\mathcal{E}$ by using ear removing until impossible is also called the Graham reduction of $\mathcal{E}$.

The acyclic-axiom of hypergraphs A hypergraph $\mathcal{E}$ is acyclic if its Graham reduction is empty.

A hypergraph is cyclic if it is not acyclic.

A partial hypergraph of an acyclic hypergraph may be cyclic. Graphs have not this property.

For example: $\mathcal{E} = \{abc, cde, efa, ace\}$ is an acyclic hypergraph, $\mathcal{E}' = \{abc, cde, efa\}$ is cyclic. See Fig.1.1.

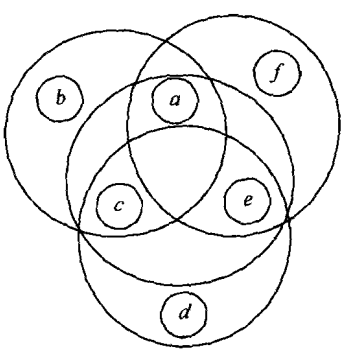

Fig.1.1

Theorem 1.1 *An acyclic hypergraph with at least two edges has at least two ears.*

Proof We prove the theorem by induction on the number of edges. It is obvious for an acyclic hypergraph with two edges.

Suppose the theorem is true for all acyclic hypergraphs with $k \geqslant 2$ edges, $\mathcal{E}$ is

an acyclic hypergraph with $k+1$ edges. Since $\mathcal{E}$ is acyclic, $\mathcal{E}$ has an ear e. Let $\mathcal{E}_1 = \mathcal{E} - e$ and $e_1 \in \mathcal{E}_1$ such that each vertex of $e \setminus e_1$ is isolated, so $\mathcal{E}_1$ is acyclic. By the induction hypothesis, $\mathcal{E}_1$ has two ears e_2, e_3 since removing e from $\mathcal{E}$ can not generate new ears except e_1, at least one of e_2 and e_3 is an ear of $\mathcal{E}$. □

Definition 1.5 (Running Intersection Property) *A hypergraph $\mathcal{E}$ has running intersection property if there exists an order $e_1, e_2, \cdots, e_m$ of $\mathcal{E}$ such that $e_i \cap (e_1 \cup e_2 \cup \cdots \cup e_{i-1}) \subseteq e_{j_i}$ for any $i, 2 \leqslant i \leqslant m$, there is $j_i < i$.*

Theorem 1.2[3,28,41] *A hypergraph $\mathcal{E}$ is acyclic iff $\mathcal{E}$ has the running intersection property.*

Proof Let $|\mathcal{E}| = m$. Suppose $\mathcal{E}$ is acyclic, then there exists an order $e_1, e_2, \cdots, e_m$ of $\mathcal{E}$ such that e_i is an ear of $\mathcal{E}_i$, $i = 1, 2, \cdots, m$, where $\mathcal{E}_m = \mathcal{E}$, $\mathcal{E}_i = \mathcal{E}_{i+1} \setminus \{e_{i+1}\}$, $i = 1, 2, \cdots, m-1$. So $\mathcal{E}_i$ is acyclic for any i, $2 \leqslant i \leqslant m$. Thus, there is $j_i < i$ such that $e_i \cap (e_1 \cup e_2 \cup \cdots \cup e_{i-1}) \subset e_{j_i}$ for any $i, 2 \leqslant i \leqslant m$. $\mathcal{E}$ has the running intersection property.

Now suppose $\mathcal{E}$ has the intersection property, then there exists an order e_1, $e_2, \cdots, e_m$ of $\mathcal{E}$ such that $e_i \cap (e_1 \cup e_2 \cup \cdots \cup e_{i-1}) \subset e_{j_i}$ for any $i, 2 \leqslant i \leqslant m$, there is $j_i < i$. Hence e_i is an ear of $\mathcal{E}_i$ for $2 \leqslant i \leqslant m$. Thus the Graham reduction of $\mathcal{E}$ is empty. □

Definition 1.6 *Let $\mathcal{E}$ be a hypergraph, the join-tree of $\mathcal{E}$ is a tree T, in which the vertices of T are the edges of $\mathcal{E}$ satisfying the following properties:*

(1) *the edge (e_i, e_j) has the labeling $e_i \cap e_j$.*

(2) *For any pair e_i, e_j, $u \in e_i \cap e_j$, each edge in the unique path from e_i to e_j in T includes u.*

For example: $\mathcal{E} = \{abcd, bcde, cdf, bceg, uvw, uvx, uwy\}$, the tree shown in Fig.1.2 is a join-tree of the hypergraph $\mathcal{E}$.

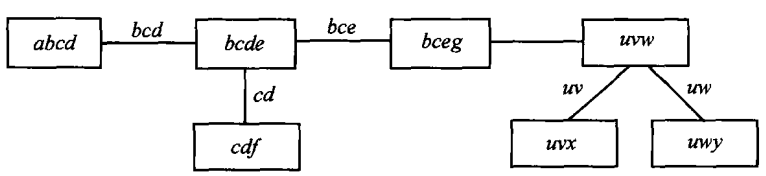

Fig.1.2 A join-tree of the hypergraph $\mathcal{E}$

Theorem 1.3[3] *A hypergraph $\mathcal{E}$ is acyclic iff $\mathcal{E}$ has a join-tree.*

Proof Suppose $\mathcal{E}$ has a join-tree T, then T has at least two leaves. Let e_1 be a leaf of T, e_2 the adjacent vertex of e_1 in T. For any $e \neq e_1$, the path from e to e_1 in

T includes the edge (e_1, e_2), so $e \cap e_1 \subseteq e_1 \cap e_2$ and e_1 is an ear of $\mathcal{E}$. $T_1 = T - e_1$ is a join-tree of $\mathcal{E} - e_1$. To remove a leaf of T_1 (which is an ear of $\mathcal{E} - e_1$) continuesly will reduce an empty set. That is, $\mathcal{E}$ will be reduced to empty set. Hence $\mathcal{E}$ is acyclic.

Now suppose $\mathcal{E}$ is acyclic hypergraph. We shall prove $\mathcal{E}$ has a join-tree. By induction on the number of edges. When $|\mathcal{E}| = 2$, it is obvious. Assume it is true for $|\mathcal{E}'| < m$ and $|\mathcal{E}| = m$. Since $\mathcal{E}$ is acyclic, $\mathcal{E}$ has an ear e_1. Let $\mathcal{E}' = \mathcal{E} - e_1$ and $e_2 \in \mathcal{E}'$ such that every vertex of $e_1 \setminus e_2$ is isolated. By the induction hypothesis, $\mathcal{E}'$ has a join-tree T'. Adding the vertex e_1 and the edge (e_1, e_2) to T' forms a tree T. Label the edge (e_1, e_2) by $e_1 \cap e_2$. We shall prove T is a join-tree of $\mathcal{E}$. For any pair of e_i and e_j, if both of $e_i \neq e_1$ and $e_j \neq e_1$ are hold, the path from e_i to e_j in T is also the (e_i, e_j)-path in T'. Thus each vertex of $e_i \cap e_j$ is on every edge in the (e_i, e_j)-path in T. Next, we prove that for any $e \in \mathcal{E}'$, every edge of the path from e to e_1 in T include $e \cap e_1$. Since every vertex of $e_1 \setminus e_2$ is isolated. $e_1 \cap e = e_2 \cap e$. Let $u \in e_1 \cap e_2$, then by the induction hypothesis, in T' the labeling of every edge of the path P' from e to e_2 includes u. The edge (e_1, e_2) includes u, so $P' + (e_1, e_2)$ is the unique path from e to e_1 in T. T is a join-tree of $\mathcal{E}$. □

Example 1.1 $\mathcal{E} = \{abc, cde, efa, ace\}$, obviously, $\mathcal{E}$ is an acyclic hypergraph. A join-tree of $\mathcal{E}$ is shown in Fig.1.3.

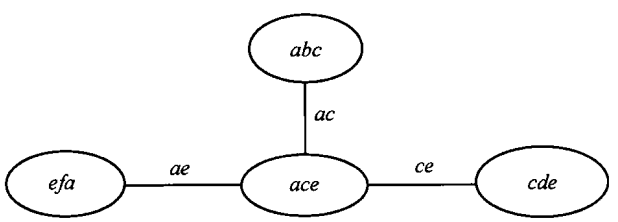

Fig.1.3 Join-tree

Example 1.2 $\mathcal{E} = \{abcd, bcde, cdef, cdh, defg\}$, $abcd \cap cdh = cd$. The labeling of each edge in the path from $abcd$ to cdh in a join-tree of $\mathcal{E}$ contains cd. It is shown in Fig.1.4.

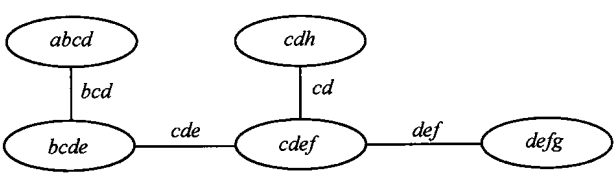

Fig.1.4 Join-tree

Chapter 2

Relational Databases

The database theory plays the centre role in information theory. The information scientists introduced the concept of acyclic hypergraphs when they studied relational databases. In this chapter, we simply introduce some concepts and results of relational databases. For the references see [3,28,29,41].

Let $\Omega = \{A_1, A_2, \cdots, A_n\}$ be a set of attributers. For each attributer A_i, there is a set of possible values called its domain, denoted by $\mathrm{dom}(A_i)$. That is, the set of values which A_i can takes.

A relation on Ω, denoted by $R[\Omega]$, is a subset of $\mathrm{dom}(A_1) \times \mathrm{dom}(A_2) \times \cdots \times \mathrm{dom}(A_n)$. Let $D_i = \mathrm{dom}(A_i)$, then $R[\Omega] \subseteq D_1 \times D_2 \times \cdots \times D_n$, n is called the arity or the degree of $R[\Omega]$. A relation is a table. Each row is called a tuple, a column corresponds an attributer. The set of attributes in a relation is called a relation scheme. $D_1 \times D_2 \times \cdots \times D_n$ is the set of all n-tuples $(a_1, a_2, \cdots, a_n)$, where $a_i \in D_i$. For example, if $n = 2$, $D_1 = \{0, 1\}$, $D_2 = \{a, b, c\}$, then $D_1 \times D_2 = \{(0, a), (0, b), (0, c), (1, a), (1, b), (1, c)\}$. $\{(0, a), (0, c), (1, b)\}$ is a relation. See the following Table 1.

The tuple $(a_1, a_2, \cdots, a_n)$ has n components, the i-th component is a_i. In general, we denote $(a_1, a_2, \cdots, a_n)$ by $a_1 a_2 \cdots a_n$.

Let Ω be the set of attributers, $\Omega_i \subseteq \Omega, i = 1, 2, \cdots, m$. If for any $i \in \{1, 2, \cdots, m\}$, $\Omega_i \neq \varnothing$ and $\bigcup_{i=1}^{m} \Omega_i = \Omega$, then we call $D = \{\Omega_1, \Omega_2, \cdots, \Omega_m\}$ as a database scheme on Ω. If R_i is a relation on Ω_i, we call $R = \{R_1, R_2, \cdots, R_m\}$ as a database on D.

Table 1

A_1	A_2
0	a
0	c
1	b

2.1 Operators and operands in relational algebra

Recall that a relation is a set of n-tuples for some fixed n, the arity of the relation. We sometimes find it is convenient to give the components of the tuples names, which are the attributers of its relation, of course, while sometimes it is convenient

to let the components be anonymous and to refer to them by numbers. When defining relational algebra, we assume columns need not be named, and order in tuples is significant. When dealing with relations as database, it is assumed that all relations are finite, and we shall adopt this assumption without explicit mansion in the future.

The operands of relational algebra are either constant relations or variables denoting relations of a fixed arity.

(1) Union. The union of relations R and S, denoted by $R \cup S$, is the set of tuples in R or S or both. We require that R and S have the same arity.

(2) Set difference. The difference between relations R and S, denoted by $R-S$, is the set of tuples in R but not in S. We again require that R and S have the same arity.

(3) Cartesian product. Let R and S be relations of arity k_1 and k_2, respectively, then $R \times S$, the Cartesian product of R and S is the set of $(k_1 + k_2)$ -tuples whose first k_1 components form a tuple in R and whose last k_2 components form a tuple in S.

(4) Projection. The idea of this operation is that for a relation R, to remove some of the components and/or rearrange some of remaining components for a relation R. If R is a relation of arity k, we let $\Pi_{i_1,i_2,\cdots,i_m}(R)$, where the i_j's are distinct integers in the range 1 to k, denote the projection of R onto components $i_1, i_2, \cdots, i_m$, that is, the set of m-tuples $a_1a_2\cdots a_m$ such that there is some k-tuple $b_1b_2\cdots b_k$ in R with $a_j = b_{i_j}$, $j = 1, 2, \cdots, m$. For example, $\Pi_{3,1}(R)$ is the set by taking each tuple t in R such that it is a 2-tuple formed from the third and first components of t in the order. If R has attributes labeling its columns, then we substitute attribute names for component numbers, and use the same attribute names in the projected relation. For example, if relation R is $R[A, B, C, D]$, then $\Pi_{C,A}(R)$ is the same as $\Pi_{3,1}(R)$, and the resulting relation has attribute C naming its first column and attribute A naming its second column.

(5) Selection. Let F be a formula involving:

(i) Operands which are constants or component number.

(ii) The arithmetic comparison operators $<$, $=$, $>$, $\leqslant$, $\neq$ and $\geqslant$, and

(iii) The logical operators $\bigwedge$(and), $\bigvee$(or), and $-$(not).

The $\sigma_F(R)$ is the set of tuples t in R satisfying the following properties: we substitute the i-th component of t for any occurrences of the number i in formula F for all i, the formula F becomes true. For example, $\sigma_{2>3}(R)$ denotes the set of tuples in R whose second component exceeds its third component, while $\sigma_{1=\text{“Smith”} \bigvee 2=\text{“Jones”}}(R)$ is the set of tuples in R whose first component has the

value "Smith" or the second component has the value "Jones". If the columns a relation are named, then the selected formular can be instead by name.

(6) Intersection. $R \cap S$ is shorthand for $R - (R - S)$.

Example 2.1 Let R, S show as in Fig.2.1.

$R =$

A	B	C
a	b	c
d	a	f
c	b	d

$S =$

D	E	F
b	g	a
d	a	f

Fig.2.1

Then the operations of (a) $R \cup S$, (b) $R - S$, (c) $R \times S$, (d) $\pi_{AC}(R)$, (e) $\sigma_{B=b}(R)$ are as follows (Fig.2.2).

a	b	c
a	d	f
c	b	d
b	g	a

(a) $R \cup S$

a	b	c
c	b	d

(b) $R - S$

A	B	C	D	E	F
a	b	c	b	g	a
a	b	c	d	a	f
d	a	f	b	g	a
d	a	f	d	a	f
c	b	d	b	g	a
c	b	d	d	a	f

(c) $R \times S$

A	C
a	c
d	f
c	d

(d) $\pi_{AC}(R)$

A	B	C
a	b	c
c	b	d

(e) $\sigma_{B=b}(R)$

Fig.2.2 Some operations

(7) Quotient. Let R and S be relations of arity r and s, respectively, where $r > s$ and $S \neq \varnothing$, then $R \div S$ is the set of $(r-s)$-tuples t such that for all s-tuples u in S, the tuple tu is in R. Let T stand for $\Pi_{1,2,\cdots,r-s}(R)$, then $(T \times S) - R$ is the set of r-tuples that are not in R, but are formed by taking the first $r - s$ components of a tuple in R and following it by a tuple in S. Let

$$V = \Pi_{1,2,\cdots,r-s}((T \times S) - R),$$

V is the set of $(r-s)$ -tuples t such that tu in the first $r - s$ components of a tuple in R is not in R for some s-tuple u in S. Hence $T - V$ is $R \div S$, that is,

$$R \div S = \Pi_{1,2,\cdots,r-s}(R) - \Pi_{1,2,\cdots,r-s}((\Pi_{1,2,\cdots,r-s}(R) \times S) - R).$$

Example 2.2 $R \div S$ is shown in Fig.2.3.

(8) Join. The θ-join of R and S on columns i and j denoted by $R \bowtie_{i\theta j} S$, where θ is an arithmetic comparison operator ($=$, $<$, and so on), is shorthand for

$\sigma_{i\theta(r+j)}(R \times S)$, if R is of arity r. That is, the θ-join of R and S is those tuples in the Cartesian product of R and S such that the i-th component of R stands in relation θ to j-th component of S. If θ is "=", the operation is often called an equijoin.

a	b	c	d
a	b	e	f
b	c	e	f
e	d	c	d
e	d	e	f
a	b	d	e

(a) R

c	d
e	f

(b) S

a	b
e	d

(c) $R \div S$

Fig.2.3

Example 2.3 Relation R, S and $R_{\bowtie_{B<D}}S$ are shown in Fig.2.4.

A	B	C
1	2	3
4	5	6
4	5	6

(a) Relation R

D	E
3	1
6	2

(b) S

A	B	C	D	E
1	2	3	3	1
1	2	3	6	2
4	5	6	6	2

(c) $R \bowtie_{B<D} S$

Fig.2.4

(9) Natural Join. The natural join, denoted by $R \bowtie S$, is applicable only when both R and S have columns that named by attributes. To compute $R \bowtie S$, we

(i) Compute $R \times S$.

(ii) For each attribute A which is in both a column in R and a column in S select from $R \times S$ those tuples whose values agree in the columns for $R \cdot A$ and $S \cdot A$. Recall that $R \cdot A$ is the name of the column of $R \times S$ corresponding to the column A of R, and $S \cdot A$ is defined analogously.

(iii) For each attribute A above, project out the column $S \cdot A$. Formally, if $A_1, A_2, \cdots, A_k$ are all the attribute names used for both R and S, $R \bowtie S$ is $\Pi_{i_1,i_2,\cdots,i_m}\, \delta_{R\cdot A_i = S\cdot A_1 \wedge \cdots \wedge R\cdot A_k} = S \cdot A_k\ (R \times S)$, where $i_1, i_2, \cdots, i_m$ is the list of all components of $R \times S$ in order the components $S \cdot A_1 \wedge \cdots \wedge S \cdot A_k$.

Example 2.4 R, S and $R_{\bowtie}S$ are shown in Fig.2.5.

(10) Semijoin. The semijoin of relations R and S, denoted by $R \ltimes S$, is $\Pi_R(R \bowtie S)$. Note that, as usual, taking the name of a relation is idenlical with its relation scheme. Observe the semijoin is not a symmetric operation. $R \rtimes S$ is $\Pi_S(R \bowtie S)$.

A	B	C
a	b	c
d	b	c
b	b	f
c	a	d

(a) R

B	C	D
b	c	d
b	c	e
a	d	b

(b) S

A	B	C	D
a	b	c	d
a	b	c	e
d	b	c	d
d	b	c	e
c	a	d	b

(c) $R \bowtie S$

Fig.2.5

Example 2.5 R, S and $R \bowtie S$ are shown in Fig.2.5. $R \ltimes S$ and $R \rtimes S$ are shown in Fig.2.6.

A	B	C
a	b	c
d	b	c
c	a	d

(a) $R \ltimes S$

B	C	D
b	c	d
b	c	e
a	d	b

(b) $R \rtimes S$

Fig.2.6

Perhaps a better way to view the semijoin $R \ltimes S$ is that we project S onto the set of attributes $R \cap S$, then delete from R all tuples t such that $t[R \cap S]$ is not in this projection. That is, we remove from R all tuples that are dangling in the sense that they produce no tuples in the join $R \bowtie S$.

The way we take semijoin in a distributed environment reflects the second meaning of this operation. We project S onto $R \cap S$ and ship the projection to the node of R. At the latter node, we perform what is technically a natural join $R \bowtie \Pi_{R \cap S}(S)$, to delete the tuples that dangle in R. Suppose that the projections of R and S onto $S \cap R$ have size r' and s' and $R \ltimes S$ and $S \ltimes R$ have size r'' and s'', respectively. Next, we compute $R \bowtie S$ by the following steps:

(1) Ship $\Pi_{R \cap S}(S)$ to the node of R.

(2) Compute $R \ltimes S$ at the node of R.

(3) Ship $R \ltimes S$ to the node of S.

(4) Compute $(R \ltimes S) \bowtie S$ at the node of S.

Theorem 2.1 $(R \ltimes S) \bowtie S = R \bowtie S$.

Proof Since $R \ltimes S = \Pi_R(R \bowtie S)$, it follows that $R \ltimes S \subseteq R$, thus $(R \ltimes S) \bowtie S \subseteq R \bowtie S$. Conversely, suppose t is a tuple in $R \bowtie S$, then $u = t[R]$ must be in R, and $V = t[S]$ must be in S. It follows that u is in $R \ltimes S$, so t is in $(R \ltimes S) \bowtie S$. Thus $R \bowtie S \subseteq (R \ltimes S) \bowtie S$. □

Example 2.6 $R[A,B]$ and $S[B,C]$ are shown in Fig.2.7.

$R \ltimes S = \{ab | ab \text{ is in } R \text{ and } b \text{ is in } S\}$. Since $\Pi_B(S) = \{1,7\}$ ab is only $0,1$. Then

$$R \ltimes S = \{0,1\}, \quad (R \ltimes S) \bowtie S = \{0,1,6\} = R \bowtie S. \tag{2.1}$$

A	B
0	1
2	3
4	5

(a) R

B	C
1	6
7	8

(b) S

Fig.2.7

(11) Semijoin programs. When we must take the join of more two relations in a distributed database, the number of ways we can take semijoins grows rapidly. In general, there may be some benefit to taking a large number of semijoins, the end result of which is that one or more of the relations have been made as small as possible. It is not hard to see that if we have relations $R_1, R_2, \cdots, R_n$, whose natural join we wish to take, then the smallest relation that R_i can become by the use of semijions is $\Pi_{R_i}(R_1 \bowtie R_2 \bowtie \cdots \bowtie R_n)$. We call this relation the reduction of R_i(with respect to $R_1, R_2, \cdots, R_n$).

Usually, all we want in our query is the reduction of one R_i. This is true for the familiar sort of query where we fix a value for one attribute and ask for the associated values of another attribute.

Example 2.7 Let A, B, C be attributes, AB, BC, AC represent $R[A, B]$, $R[B, C]$, $R[A, C]$ respectively and for some n, suppose that the current values of these relations are $AB = \{a_1b_1, a_2b_2, \cdots, a_nb_n\}$, $BC = \{b_1c_1, b_2c_2, \cdots, b_nc_n\}$, $AC = \{a_2c_1, a_3c_2, \cdots, a_{n+1}c_n\}$. It is easy to see that the join of these three relations is empty. However, it is easy to show by induction on i, that after i steps of any semijoin program, no tuple will be deleted from any of the three relations unless it has a value with subscript i or less, or a subscript $n - i + 1$ or more. For example, a_3c_2 could not be deleted until the second step. In fact, it takes six steps to delete that tuple from AC; the shortest semijoin program is $AB := AB \ltimes AC$; $BC := BC \ltimes AB$; $AC := AC \ltimes AC \ltimes BC$; $AB := AB \ltimes AC$; $BC := BC \ltimes AB$; $AC = AC \ltimes BC$. It turns out that acyclicity has something to do with the question of whether a semijoin to reduce a relation exists. In fact, any natural join can be viewed as a hypergraph, where the vertices are attributes and the edges are relation schemes in the join.

Theorem 2.2[41] *If a join expression has an acyclic hypergraph, then there is a semijoin program to reduce any relation in the join. Conversely, if the hypergraph is cyclic, then there is at least one relation for which no semijoin program is quaranted to compute its reduction.*

Proof The converse portion, showing that cyclic hypergraph do not have semijoin program reducing each of the relations, is left as a rather difficult exercise generalizing Example 2.7. The first part of the theorem can be proved by giving the following algorithm for constructing a semijoin program that computes the reduction of a particular relation R. The algorithm is defined inductively, starting with join of a single relation R, for which the empty program suffices, and proceeding to progressively large sets of relations that have an acyclic hypergraph. Suppose we have an acyclic hypergraph of more than one edge. Then by definition, it has an ear, said edge S. Assume at first $S \neq R$, as S is an ear, there must be some edge T such that each vertex in S is either unique to S or in T. Let us start our semijoin program with the step $T := T \ltimes S$.

By induction, the remaining hypergraph, which must also be acyclic, yields a semijoin program to compute $\Pi_R(R_1 \bowtie R_2 \bowtie \cdots \bowtie R_k)$, where $R_1 \bowtie R_2 \bowtie \cdots \bowtie R_k$ are the edges of the remaining hypergraph. Suppose without loss of generality that $T = R_1$. If we proceed the constructed program with $R_1 := R_1 \ltimes S$, the resulting program computes $\Pi_R((R_1 \ltimes S) \bowtie R_2 \bowtie \cdots \bowtie r_k)$. As S has no attribute in common with any of the R_i's that it does not have in common with $T = R_1$, the above expression is equal to $\Pi_R(S \bowtie R_1 \bowtie \cdots \bowtie R_k)$, proving that the constructed semijoin program reduced R.

If $S = R$, by induction to construct a semijoin program to reduce T. In remaining hypergraph, we follow it by the step $R := R \ltimes T$. The resulting program $R \ltimes (\Pi_T(R_1 \bowtie \cdots \bowtie R_k)$ is the reduction of R, since R has no attribute in common with the R_i's that it does not share with T. □

Example 2.8 Fig.2.8 shows an acyclic hypergraph.

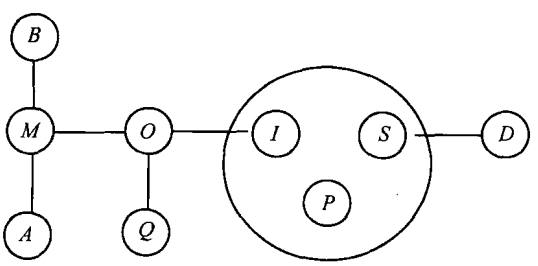

Fig.2.8