TR-87-35

Using 2-way Semijoins in Distributed Query Processing

by

Hyunchul Kang
Nick Roussopoulos

Using 2-way Semijoins in Distributed Query Processing *

Hyunchul Kang **
and
Nick Roussopoulos

Department of Computer Science
University of Maryland
College Park, Maryland 20742

Abstract

In distributed query processing, the semijoin has been used as an effective operator in reducing relations referenced in the query to reduce the total amount of data transmission. In this paper, we propose a new relational algebra operator, *2-way semijoin*, which is an extended version of the semijoin, for more cost-effective distributed query processing. The 2-way semijoin is compared to the semijoin in terms of the reduction power and the propagation of reduction effects. We show that the 2-way semijoin has more reduction power than the semijoin and that the propagation of reduction effects by the 2-way semijoin is further than by the semijoin.

1. Introduction

In a distributed database system, processing a distributed query involves data transmission among the different sites of the computer network. In a system where the communication cost is a dominant factor among the costs of system resources, that is, the local processing cost is negligible compared to the data transmission cost, it is essential to reduce the total amount of data transmission to process a distributed query. In such a system, the optimal query processing strategy is defined as the strategy which can answer the query with minimum data transmission.

Given a distributed query in such a system, how to generate an optimal processing strategy has been a great interest to a number of researchers [21, 4, 18, 2, 5, 6, 22, 23]. In these papers, a given distributed query is processed by the following 3 phases:

(1) *Local processing phase*: at each site involved in the query processing, all local processing such as selections from relations and projections on the joining and target attributes is performed [1].

(2) *Reduction phase*: executes a *semijoin program* (a sequence of semijoins) generated by the query optimization algorithm to reduce relations in a cost-effective way and thus, to reduce the total amount of data transmission.

(3) *Final query processing phase*: selects a final processing site and sends all reduced relations to that site where the final query processing is performed to answer the query.

In this framework, the core of the optimization is in the reduction phase and the primary concern of a query optimization algorithm is to generate a semijoin program that will be used in this phase. The major difference from algorithm to algorithm lies in how to generate such semijoin programs.

Several algorithms to generate the optimal processing strategy have been proposed for some restricted class of queries such as simple queries [17], tree queries [24, 13], chain queries [12] and star queries [8, 9]. However for general queries with arbitrary complexity, even for queries with very primitive complexity, to generate an optimal processing strategy is proved to be NP-hard [17, 23, 16]. Therefore the optimization algorithms for general queries heavily depend on heuristics. Despite the differences in their heuristics, it is common for all algorithms that the *semijoin* [3] is used as an effective operator to reduce relations.

In this paper, we propose a new relational operator which we call a *2-way semijoin*, for more cost-effective distributed query processing. A 2-way semijoin is an extended version of the semijoin and like the semijoin, it can be used as an effective operator to reduce relations.

* This work was supported in part by NASA under contract No. NAS-5-29-265 and in part by the National Science Foundation under grant No. OIR-85-00108.

** Also with the Systems Research Center at University of Maryland, College Park, MD 20742.

[1] In the case of distributed database with replicated and/or fragmented relations, we assume that a set of non-redundant copies referenced in the query is pre-selected before processing begins.

The organization of this paper is as follows. In section 2 of this paper, we review the semijoin in distributed query processing. In section 3, we define the 2-way semijoin and compare it to the semijoin in terms of the reduction power and the propagation of reduction effects. Concluding remarks are in section 4.

2. Using Semijoins in Distributed Query Processing

2.1. Semijoin

Consider a semijoin from R_i to R_j on attribute A.[2] Let us denote this semijoin as $R_i - A \rightarrow R_j$. The result of the semijoin $R_i - A \rightarrow R_j$ is the projection on the attributes of R_j of the join of R_i and R_j.

Suppose, in a given distributed query, a join is to be computed between relation R_i at site i and relation R_j at site j on attribute A, $i \neq j$. To reduce the total amount of data transmission in computing the join, semijoins are used to reduce relation R_i and/or R_j before they are sent to the final processing site where the join of R_i and R_j is computed. The semijoin $R_i - A \rightarrow R_j$ is computed by the following steps:

(1) Send $R_i[A]$ from site i to j.

(2) Reduce R_j by eliminating tuples whose attribute A are not matching any value in $R_i[A]$.

Step (1) of the semijoin incurs the *cost* of the semijoin and step (2) returns the *benefit* of the semijoin.

2.2. Cost and Benefit of Semijoin

Consider a semijoin $R_i - A \rightarrow R_j$ when R_i and R_j are at different sites. Let the transmission cost be 1 per data unit transmitted. Then, the cost of the semijoin is

$$S(R_i[A])$$

where $S(R)$ is the size of relation R. Suppose this semijoin reduces R_j to R_j'. Then, the benefit is

$$S(R_j) - S(R_j')$$

A *cost-effective semijoin* is a semijoin whose benefit exceeds the cost.

Given a distributed query, let IFS be the initial feasible strategy which is a processing strategy by which no relation referenced in the query is reduced before they are sent to the final query processing site. By adding a cost-effective semijoin to IFS, we can transform IFS to an improved strategy. In general, by adding a cost-effective semijoin to the current strategy, we can transform it to an improved one.

[2] We assume that the renaming rules are applied to the joining attributes such that the common joining attributes have the same attribute name [21, 25]. We also assume that when there are more than one joining attributes between two relations, we consider the semijoins on each attribute separately. These assumptions will also be used in defining the 2-way semijoin in section 3 for the notational convenience.

2.3. Cost and Benefit Estimation

To estimate the cost and benefit of a given semijoin, it is assumed that the *database profile* is maintained in the system catalog. The database profile consists of two parts: statistics on relations and statistics on domains of attributes. That is, the database profile includes: for each relation R, its cardinality, the average size of a tuple and, for each attribute of R, the number of occurrences of the unique values; for each domain of an attribute, its cardinality and the average size of a value.

Assuming the uniform distribution of attribute values and the independent distribution of values across attributes, the cost and benefit of semijoins can be estimated using the selectivities of joining attributes [4, 19]. Since it would be very expensive to dynamically keep the database profile up-to-date against every database update, it is brought up-to-date usually off-line on a periodic basis.

3. 2-way Semijoin

The original relational algebra operator is either a unary or a binary operator which produces only one result relation [14]. Some extensions to the relational algebra have been considered by allowing operators to produce more than one result relations [15, 1]. We define a new relational operator, *2-way semijoin*, which is an extended version of the semijoin. The 2-way semijoin is an *extended* operator in the sense that it produces two semijoins from its operands.

3.1. Definition of 2-way Semijoin

Formally, the result of a 2-way semijoin of R_i and R_j on attribute A is a set of two relations $\{R_i', R_j'\}$ where R_i' (R_j') is the projection on the attributes of R_i (R_j) of the join of R_i and R_j.[3] Let us denote the 2-way semijoin of R_i and R_j on attribute A as $R_i \leftarrow A \rightarrow R_j$. Then,

$$R_i \leftarrow A \rightarrow R_j = \{ R_i - A \rightarrow R_j, R_j - A \rightarrow R_i \}$$

Semantically, $R_i \leftarrow A \rightarrow R_j$ reduces R_i and R_j to R_i' and R_j', respectively such that

(1) $\text{join}(R_i', R_j') = \text{join}(R_i, R_j)$, and

(2) There is no pair of relation states $\{R_i'', R_j''\}$ such that $R_i'' \subseteq R_i'$, $R_j'' \subseteq R_j'$, and $\text{join}(R_i'', R_j'') = \text{join}(R_i, R_j)$.

[3] The complete definition of the 2-way semijoin which is consistent with the definition of the join is possible by allowing arbitrary join predicates. The natural 2-way semijoin can also be defined.

Suppose that a join is to be computed between relation R_i at site i and relation R_j at site j on attribute A, $i \neq j$. As is the case with semijoins, to reduce the total amount of data transmission in computing the join, 2-way semijoins can be used to reduce relation R_i and/or R_j before they are sent to the final processing site where the join of R_i and R_j is computed. A 2-way semijoin $R_i \leftarrow A \rightarrow R_j$ is computed by the following steps:

(1) Send $R_i[A]$ from site i to j.

(2) Reduce R_j by eliminating tuples whose attribute A are not matching any of $R_i[A]$. During reduction of R_j, partition $R_i[A]$ into $R_i[A]_m$ and $R_i[A]_{nm}$ where $R_i[A]_m$ is the set of values in $R_i[A]$ which match one of $R_j[A]$ and, $R_i[A]_{nm}$ is $R_i[A] - R_i[A]_m$ [4].

(3) Send either $R_i[A]_m$ or $R_i[A]_{nm}$ whichever is less in size from site j back to i.

(4) Reduce R_i using either $R_i[A]_m$ or $R_i[A]_{nm}$. If $R_i[A]_m$ is used, then tuples whose attribute A are not matching any of $R_i[A]_m$ are eliminated. If $R_i[A]_{nm}$ is used, then tuples whose attribute A are matching one of $R_i[A]_{nm}$ are eliminated.

3.2. Cost and Benefit of 2-way Semijoin

The result of a 2-way semijoin $t: R_i \leftarrow A \rightarrow R_j$ is identical to the results of two separate semijoins $s: R_i \rightarrow A \rightarrow R_j$ and $s^r: R_j \rightarrow A \rightarrow R_i$. That is, the reduction effects of t are the same as those of s and s^r. Thus, the benefit of t is the sum of the benefits of s and s^r. Suppose t reduces R_i and R_j to R_i' and R_j', respectively. Then, the benefit of t which is due to reduction in step (2) and (4) of computing t is

$$[\,S(R_i) - S(R_i')\,] + [\,S(R_j) - S(R_j')\,]$$

However, in general, the cost incurred in computing t is different from the sum of costs incurred in computing s and s^r. The cost of t which is incurred in step (1) and (3) of computing t is

$$S(R_i[A]) + min\,[\,S(R_i[A]_m), S(R_i[A]_{nm})\,]$$

while the sum of costs of s and s^r is

$$S(R_i[A]) + S(R_j[A])$$

or if s^r is delayed until s is finished, the sum of costs would be

$$S(R_i[A]) + S(R_j'[A]) = S(R_i[A]) + S(R_i[A]_m)$$

[4] In $R_i[A]_m$ and $R_i[A]_{nm}$, m and nm stands for *match* and *not match*, respectively.

As is the case in the semijoin, a *cost-effective 2-way semijoin* is a 2-way semijoin whose benefit exceeds its cost. By adding a cost-effective 2-way semijoin to IFS, we can transform IFS to an improved strategy. In general, by adding a cost-effective 2-way semijoin to the current query processing strategy, we can transform it to an improved one.

3.3. Comparison of 2-way Semijoin to Semijoin

3.3.1. Reduction Power

The rationale behind the extension of the semijoin to the 2-way semijoin is that the 2-way semijoin can be more effective than the semijoin in reducing relations referenced in distributed queries.

For a semijoin $s: R_i \rightarrow A \rightarrow R_j$, let $C(s)$ and $B(s)$ be the cost and the benefit of s, respectively and let $D(s)$ be $B(s) - C(s)$. For a 2-way semijoin $t: R_i \leftarrow A \rightarrow R_j$, let $C(t)$ and $B(t)$ be the cost and the benefit in reducing R_j by t, respectively and let $D(t)$ be $B(t) - C(t)$. Let $C^+(t)$ and $B^+(t)$ be the cost and the benefit in reducing R_i by t, respectively and let $D^+(t)$ be $B^+(t) - C^+(t)$.

Lemma 1

Given a semijoin $s: R_i \rightarrow A \rightarrow R_j$, if the 2-way semijoin $t: R_i \leftarrow A \rightarrow R_j$ is performed instead of s, then

(1) If s reduces R_j to R_j', t reduces R_j to R_j'.

(2) The cost and benefit in reducing R_j by s is the same as those in reducing R_j by t.

(3) R_i is always reduced by t in a cost-effective way.

Proof

(1) and (2) are true by definition of the semijoin and of the 2-way semijoin.

For (3), suppose t reduces R_i to R_i'.
Let r and r' be $|R_i|$ and $|R_i'|$, respectively and let m, n be $|R_i[A]_m|$ and $|R_i[A]_{nm}|$, respectively.
Let w be the size of a tuple of R_i and let a be the size of a value of attribute A. Then we have,

$$C^+(t) = a \times min\,[\,m, n\,]$$

$$B^+(t) = w \times (r - r')$$

It is straightforward to observe that $w \geq a$ and that $r - r' \geq n$. And since $n \geq min\,[\,m, n\,]$,

$$D^+(t) = B^+(t) - C^+(t) = w \times (r - r') - a \times min\,[\,m, n\,] \geq w \times (r - r') - a \times n \geq 0$$

as desired. □

Corollary 1
For a semijoin $R_i - A \rightarrow R_j$, if it is cost-effective, then the 2-way semijoin $R_i \leftarrow A \rightarrow R_j$ is also cost-effective.

Corollary 2
For a 2-way semijoin $R_i \leftarrow A \rightarrow R_j$, if it is not cost-effective, then the semijoin $R_i - A \rightarrow R_j$ is not also cost-effective.

Lemma 1 states that the extension of the semijoin to the 2-way semijoin is done in a cost-effective way *no matter what the cost and the benefit in the first two steps of computing a 2-way semijoin, the last two steps are always performed cost-effectively*. We also note that the cost-effectiveness of the last two steps is independent of any particular cost and benefit estimation technique.

Lemma 2
Consider a pair of semijoins $s : R_i - A \rightarrow R_j$ and $s^r : R_j - A \rightarrow R_i$. Suppose neither s nor s^r is cost-effective. There exists a condition under which the 2-way semijoin $t : R_i \leftarrow A \rightarrow R_j$ is cost-effective and similarly, there exists a condition under which the 2-way semijoin $t^r : R_j \leftarrow A \rightarrow R_i$ is cost-effective.

Proof
Since s is not cost-effective, $D(s) < 0$. Thus, $D(t) < 0$. However, by Lemma 1, $D^+(t) \geq 0$.

When $D(t) < 0$ and $D^+(t) \geq 0$, it is straightforward to see that $B(t) + B^+(t) \geq C(t) + C'^+(t)$ iff $|D^+(t)| \geq |D(t)|$.

Thus, t is cost-effective under the condition that $|D^+(t)| \geq |D(t)|$. Similarly, t^r is cost-effective under the condition that $|D^+(t^r)| \geq |D(t^r)|$. □

By Lemma 1 and Lemma 2, for two relations to be joined, even when only one or none can be reduced cost-effectively using semijoins, both may be reduced cost-effectively using a 2-way semijoin.

3.3.2. Propagation of Reduction Effects

Given a query Q, let J_Q be the set of join clauses of Q, and let $J_Q{}^+$ be the transitive closure of J_Q, that is, the set of all the join clauses inferred by J_Q. A join clause $R_i.A = R_j.A$ in $J_Q{}^+$ is associated with two semijoins $R_i - A \rightarrow R_j$ and $R_j - A \rightarrow R_i$, and two 2-way semijoins: $R_i \leftarrow A \rightarrow R_j$ and $R_j \leftarrow A \rightarrow R_i$. Although the latter two are the same operations by definition, we distinguish the two because they incur different amount of data transmission costs when R_i and R_j are at different sites. For a given query Q, let S be the set of all the possible semijoins associated with $J_Q{}^+$. S is partitioned into two subsets S_e and $\overline{S_e}$ where S_e is the set of all the cost-effective semijoins and $\overline{S_e}$ is $S - S_e$. Similarly, let T be the set of all the possible 2-way semijoins associated with $J_Q{}^+$. T is partitioned into two subsets T_e and $\overline{T_e}$ where T_e is the set of all the cost-effective 2-way semijoins and $\overline{T_e}$ is $T - T_e$.

Given a query, S, S_e, $\overline{S_e}$ and T, T_e, $\overline{T_e}$ are initialized as defined above. Initially, it is obvious that $|S| = |T|$ and, $|S_e| \leq |T_e|$. In the latter, the equality holds by Corollary 1 and the inequality holds by Lemma 2.

In this section, we compare the 2-way semijoin to the semijoin in terms of the propagation of reduction effects. When a semijoin in S is selected by the query optimization algorithm as the next semijoin in the query processing strategy, its reduction effects are propagated to other semijoins in S: the cost and the benefit of some semijoins in S are updated and thus, some semijoins in S_e may move to $\overline{S_e}$ and vice versa. Similarly, when a 2-way semijoin in T is selected as the next 2-way semijoin in the query processing strategy, its reduction effects are propagated to other 2-way semijoins in T: the cost and the benefit of some 2-way semijoins in T are updated and thus, some 2-way semijoins in T_e may move to $\overline{T_e}$ and vice versa.

Suppose that a semijoin $s : R_i - A \rightarrow R_j$ in S is selected. Then, there are four types of semijoins in S which are directly interacting with s (see Fig. 3.1).

type-1: $R_j - B \rightarrow R_k$ type-2: $R_k - B \rightarrow R_j$

type-3: $R_i - B \rightarrow R_k$ type-4: $R_k - B \rightarrow R_i$

Suppose that a 2-way semijoin $t : R_i \leftarrow A \rightarrow R_j$ in T is selected. Then, there are also four types of 2-way semijoins in T which are directly interacting with t (see Fig. 3.2).

type-1: $R_j \leftarrow B \rightarrow R_k$ type-2: $R_k \leftarrow B \rightarrow R_j$

type-3: $R_i \leftarrow B \rightarrow R_k$ type-4: $R_k \leftarrow B \rightarrow R_i$

When a semijoin s in S is selected, the semijoin s_1 in S which directly interacts with s is denoted as s_1' after the cost and the benefit of s_1 are updated due to the reduction effects of s. Then, with respect to s, let $\Delta D(s_1)$ be $D(s_1') - D(s_1)$. Similarly, when a 2-way semijoin t in T is selected, the 2-way semijoin t_1 in T which directly interacts with t is denoted as t_1' after the cost and the benefit of t_1 are updated due to the reduction effects of t. Then, with respect to t, let $\Delta D(t_1)$ be $D(t_1') - D(t_1)$ and let $\Delta D^+(t_1)$ be $D^+(t_1') - D^+(t_1)$. We note the following properties about interactions among semijoins.

Property 1
Let s and s_1 be semijoins such that s_1 is of type-1 with respect to s. Then, after s is performed, $\Delta D(s_1) \geq 0$.

Property 2
Let s and s_2 be semijoins such that s_2 is of type-2 with respect to s. Then, after s is performed, $\Delta D(s_2) \leq 0$.

Property 3
Let s and s_{34} be semijoins such that s_{34} is either of type-3 or of type-4 with respect to s. Then, after s is performed, $\Delta D(s_{34}) = 0$.

As for interactions among 2-way semijoins, we have the following lemmas.

Lemma 3
Let t and u be 2-way semijoins such that u is either of type-1 or of type-3 with respect to t. Then, after t is performed, $\Delta D^+(u) \leq 0$.

Proof
Let t be $R_i \leftarrow A \rightarrow R_j$ and u be $R_j \leftarrow B \rightarrow R_k$ (of type-1, see Fig. 3.2). Suppose t reduces R_j to R_j' with selectivity p, that is, $|R_j'| = p \times |R_j|$.

With respect to u, let m, n be $|R_j[B]_m|$, $|R_j[B]_{nm}|$, respectively and let m', n' be $|R_j'[B]_m|$, $|R_j'[B]_{nm}|$, respectively. Let w be the size of a tuple of R_j and let b be the size of a value of attribute B.

Suppose u would reduce R_j to R_j'' if u were performed before t. Let r, r' be the number of tuples of R_j eliminated by u and the number of tuples of R_j' eliminated by u. Then we have,

$$r = |R_j| - |R_j''|$$

$$r' = p \times |R_j| - p \times |R_j''| = p \times r$$

case : $m \leq n$:
Since $m' = p \times m$ and $n' = p \times n$, $m' \leq n'$. Thus,
$\Delta C^+(u) = C^+(u') - C^+(u)$
$= b \times m' - b \times m = b \times m \times (p-1)$
and
$\Delta B^+(u) = B^+(u') - B^+(u)$
$= w \times r' - w \times r = w \times r \times (p-1)$
Since $b \leq w$ and $m \leq n \leq r$, $\Delta B^+(u) \leq \Delta C^+(u)$.

case : $m \geq n$:
We have $m' \geq n'$. Thus,
$\Delta C^+(u) = b \times n \times (p-1)$
$\Delta B^+(u) = w \times r \times (p-1)$
Since $b \leq w$ and $n \leq r$, $\Delta B^+(u) \leq \Delta C^+(u)$.

$$\Delta D^+(u) = D^+(u') - D^+(u)$$
$$= (B^+(u') - C^+(u')) - (B^+(u) - C^+(u))$$
$$= (B^+(u') - B^+(u)) - (C^+(u') - C^+(u))$$
$$= \Delta B^+(u) - \Delta C^+(u)$$

Thus, $\Delta D^+(u) \leq 0$.

The proof for type-3 2-way semijoin is the same by switching R_i and R_j. □

Lemma 4
Let t and u be 2-way semijoins such that u is either of type-2 or of type-4 with respect to t. Then, after t is performed, $\Delta D^+(u) \geq 0$.

Proof
Let t be $R_i \leftarrow A \rightarrow R_j$ and u be $R_k \leftarrow B \rightarrow R_j$ (of type-2, see Fig. 3.2). Using the notations in the proof of Lemma 3, $\Delta D^+(u) = \Delta B^+(u) - \Delta C^+(u)$.

After t is performed, R_j is reduced but R_k is not affected. Thus, when u is performed after t, it is always that $\Delta B^+(u) \geq 0$. Therefore, if $\Delta C^+(u) \leq 0$, then $\Delta D^+(u) \geq 0$. Otherwise, it should be that $\Delta B^+(u) \geq \Delta C^+(u)$.

In case u is performed before t, let m, n be $|R_k[B]_m|$, $|R_k[B]_{nm}|$, respectively.
In case u is performed after t, let m', n' be $|R_k[B]_m|$, $|R_k[B]_{nm}|$, respectively.
Let w be the size of a tuple of R_k and let b be the size of a value of attribute B.

case : $m \leq n$:
Since $m' \leq m$ and $n' \geq n$, $m' \leq n'$. Thus, $\Delta C^+(u) = b \times m' - b \times m \leq 0$.

case : $m \geq n \wedge m' \geq n'$:
$\Delta C^+(u) = b \times n' - b \times n = b \times (n' - n)$
Since $n' \geq n$, $\Delta C^+ \geq 0$.
However, $\Delta B^+(u) \geq w \times (n' - n)$
Since $w \geq b$, $\Delta B^+(u) \geq \Delta C^+(u)$.

case : $m \geq n \wedge m' \leq n'$:
$\Delta C^+(u) = b \times m' - b \times n = b \times (m' - n)$
Suppose $\Delta C^+(u) \leq 0$.
Since $n' \geq m'$, $n' - n \geq m' - n$. Thus,
$\Delta B^+(u) \geq w \times (n' - n) \geq w \times (m' - n)$
$\geq b \times (m' - n) = \Delta C^+(u)$

The proof for type-4 2-way semijoin is the same by switching R_i and R_j. □

Based on these interactions, we compare the propagation of the reduction effects of t to type-i 2-way semijoins in T to that of s to type-i semijoins in S, $i = 1,2,3,4$, respectively. All the conditions stated below can be formally quantified.

type-1
Let s_1 and t_1 be a type-1 semijoin and a type-1 2-way semijoin, respectively (see Fig. 3.1, 3.2). Then, the propagation of the reduction effects of s to s_1 and of t to t_1 are

s to s_1
By Property 1, if $s_1 \in S_e$, $s_1' \in S_e$. If $s_1 \in \overline{S_e}$, there is a condition E_1 under which $s_1' \in S_e$.

t to t_1
By Property 1, $\Delta D(t_1) \geq 0$. By Lemma 3, $\Delta D^+(t_1) \leq 0$ but by Lemma 1, $D^+(t_1') \geq 0$. Thus, if $t_1 \in T_e \wedge D(t_1) \geq 0$, $t_1' \in T_e$. If $t_1 \in T_e \wedge D(t_1) \leq 0$, there is a condition E_2 under which $t_1' \in \overline{T_e}$; if $t_1 \in \overline{T_e}$, there is a condition E_3 under which $t_1' \in T_e$.

comparison
Suppose s_1 and t_1 are on the same relations. If $t_1 \in T_e \wedge D(t_1) < 0$, since $D(s_1) = D(t_1)$, $s_1 \in \overline{S_e}$. Thus, if E_2 holds, by Corollary 2, $s_1' \in S_e$. If $t_1 \in \overline{T_e}$, by Corollary 2, $s_1 \in \overline{S_e}$. Since

$\Delta D(s_1) = \Delta D(t_1)$, by Corollary 1, if E_1 holds, E_3 holds

type-2

Let s_2 and t_2 be a type-2 semijoin and a type-2 2-way semijoin, respectively (see Fig. 3.1, 3.2). Then, the propagation of the reduction effects of s to s_2 and of t to t_2 are

s to s_2.

By Property 2, if $s_2 \in \overline{S_e}$, $s_2' \in \overline{S_e}$; if $s_2 \in S_e$, there is a condition F_1 under which $s_2' \in \overline{S_e}$.

t to t_2.

By Property 2, $\Delta D(t_2) \leq 0$ but by Lemma 4, $\Delta D^+(t_2) \geq 0$. Thus, if $t_2 \in \overline{T_e}$, there is a condition F_2 under which $t_2' \in T_e$; if $t_2 \in T_e$, there is a condition F_3 under which $t_2' \in \overline{T_e}$.

comparison.

Suppose s_2 and t_2 are on the same relations. If $t_2 \in T_e$ and F_3 holds, by Corollary 2, $s_2' \in \overline{S_e}$. Thus, if F_3 holds, F_1 holds. But by Lemma 2, when F_1 holds, F_3 does not always hold; If $t_2 \in \overline{T_e}$, by Corollary 2, $s_2 \in \overline{S_e}$. Thus, $s_2' \in \overline{S_e}$ but by Lemma 2, F_2 may hold.

type-3

Let s_3 and t_3 be a type-3 semijoin and a type-3 2-way semijoin, respectively (see Fig. 3.1, 3.2). Then, the propagation of the reduction effects of s to s_3 and of t to t_3 are

s to s_3.

By Property 3, if $s_3 \in S_e$, $s_3' \in S_e$; if $s_3 \in \overline{S_e}$, $s_3' \in \overline{S_e}$.

t to t_3.

The same as t to t_1 of type-1 above.

comparison

Suppose s_3 and t_3 are on the same relations. If $t_3 \in T_e \wedge D(t_3) < 0$, since $D(s_3) = D(t_3)$, $s_3 \in S_e$. Thus, $s_3' \in S_e$ regardless of whether E_2 holds or not; If $t_3 \in \overline{T_e}$, by Corollary 2, $s_3 \in \overline{S_e}$. Thus, $s_3' \in \overline{S_e}$ regardless of whether E_3 holds or not.

type-4

Let s_4 and t_4 be a type-4 semijoin and a type-4 2-way semijoin, respectively (see Fig. 3.1, 3.2). Then, the propagation of the reduction effects of s to s_4 and of t to t_4 are

s to s_4.

By Property 3, if $s_4 \in S_e$, $s_4' \in S_e$, if $s_4 \in \overline{S_e}$, $s_4' \in \overline{S_e}$.

t to t_4.

The same as t to t_2 of type-2 above.

comparison

Suppose s_4 and t_4 are on the same relations. If $t_4 \in \overline{T_e}$, by Corollary 2, $s_4 \in \overline{S_e}$. Thus, $s_4' \in \overline{S_e}$ regardless of whether F_2 holds or not. If $t_4 \in T_e \wedge D(t_4) < 0$, since $D(s_4) = D(t_4)$, $s_4 \in \overline{S_e}$. Thus, $s_4' \in \overline{S_e}$ regardless of whether F_3 holds or not. If $t_4 \in T_e \wedge D(t_4) \geq 0$, since $D(s_4) = D(t_4)$, $s_4 \in S_e$. Thus, $s_4' \in S_e$ regardless of whether F_3 holds or not.

For all types and all cases, the propagation of reduction effects by 2-way semijoins is further than by semijoins except one case in type-4. However, even in that case, the propagation by 2-way semijoins could be as far as by semijoins. To see this point, we restate that case.

When $s_4 \in S_e \wedge t_4 \in T_e$, it is always that $s_4' \in S_e$ but it may be that $t_4' \in \overline{T_e}$ under the condition F_3.

Let t^r be the reverse 2-way semijoin of t, that is, if t is $R_i \leftarrow A \rightarrow R_j$, then t^r is $R_j \leftarrow A \rightarrow R_i$. We note that when t_4 is in T, t_4^r is also in T. t_4^r is of type-3 with respect to t (with respect to which t_4 is of type-4). Thus, if $t_4^r \in T_e$, $(t_4^r)' \in T_e$ if E_2 does not hold, and the same reduction effects by t_4 could be achieved by $(t_4^r)'$ cost-effectively. If $t_4^r \in \overline{T_e}$, $(t_4^r)' \in T_e$ if E_3 holds, and $(t_4^r)'$ could be performed cost-effectively.

4. Concluding Remarks

In distributed query processing, the semijoin has been used as an effective operator in reducing relations. In this paper, we proposed a new relational algebra operator, 2-way semijoin, for more cost-effective distributed query processing. By Lemma 1, its Corollaries and Lemma 2, we showed that the 2-way semijoin has more reduction power than the semijoin. Studying the interactions among 2-way semijoins and among semijoins, we also showed that the propagation of reduction effects by the 2-way semijoin is further than by the semijoin.

We are now considering a set of new heuristics to generate a query processing strategy using 2-way semijoins. For example, some existing heuristic algorithms based on semijoins [4, 5] can be easily either modified by replacing semijoins with 2-way semijoins or extended by using 2-way semijoins together with semijoins. The early simulation results showed a little improvement in the former case and a significant improvement in the latter.

The query processing strategy generated by a heuristic algorithm is, in general, suboptimal. The properties that an optimal semijoin program should possess have been studied [7] and the algorithms for improving semijoin programs have been proposed [10, 11, 20, 4, 22, 25]. Most of these improving techniques for semijoin programs are applicable to query processing strategies using 2-way semijoins as well but we need to investigate for the characteristics of them which are not present in semijoin programs for further improvements.

References

[1] Adiba, M., *Derived Relations: A Unified Mechanism for Views, Snapshots and Distributed Data*, Proc. 7th Int. Conf. on Very Large Data Bases, 1981.

[2] Apers, P., Hevner, A.R. and Yao, S.B., *Optimization Algorithm for Distributed Queries*, IEEE Trans. Softw. Eng. SE-9,1, Jan. 1983.

[3] Bernstein, P.A. and Chiu, D.W., *Using Semi-Joins to Solve Relational Queries*, J. ACM 28,1, Jan. 1981

[4] Bernstein, P.A., Goodman, N., Wong, E., Reeve, C. and Rothnie, J.B., *Query Processing in a System for Distributed Databases (SDD-1)*, ACM Trans. Database Syst. 6,4, Dec. 1981.

[5] Black, P. and Luk, W., *A New Heuristic for Generating Semi-Join Programs for Distributed Query Processing*, Proc. IEEE COMPSAC 1982.

[6] Chang, J., *A Heuristic Approach to Distributed Query Processing*, Proc. 8th Int. Conf. on Very Large Data Bases, 1982

[7] Chen, A.L.P and Li, V.O.K., *Properties of Optimal Semijoin Programs for Distributed Query Processing*, Proc. IEEE COMPSAC 1983.

[8] Chen, A.L.P and Li, V.O.K., *Deriving Optimal Semi-Join Programs for Distributed Query Processing*, Proc. IEEE INFOCOM 1984.

[9] Chen, A.L.P and Li, V.O.K., *Optimizing Star Queries in a Distributed Database System*, Proc. 10th Int. Conf. on Very Large Data Bases, 1984.

[10] Chen, A.L.P and Li, V.O.K., *Improvement Algorithms for Semijoin Query Processing Programs in Distributed Database Systems*, IEEE Trans. Computers C-33,11, Nov 1984

[11] Chen, A.L.P. and Li, V.O.K., *Improving Semi-Join Programs for Distributed Query Processing*, Proc. IEEE COMPSAC 1984.

[12] Chiu, D.W., Bernstein, P.A. and Ho, Y., *Optimizing Chain Queries in a Distributed Database System*, SIAM J. COMPT 13,1, Feb 1984.

[13] Chiu, D.W. and Ho, Y., *A Methodology for Interpreting Tree Queries into Optimal Semi-Join Expressions*, Proc. ACM SIGMOD Int. Conf. on Management of Data, 1980

[14] Codd, E.F., *A Relational Model for Large Shared Data Banks*, Comm. ACM, 13,6, Jun. 1970

[15] Codd, E.F., *Extending the Database Relational Model to Capture More Meaning*, ACM Trans. Database Syst., 4,4, Dec 1979

[16] Goodman, N. and Shmueli, O., *The Tree Property is Fundamental for Query Processing*, Proc. ACM Symp. on Principles of Database Systems, 1982.

[17] Hevner, A.R., *The Optimization of Query Processing on Distributed Database Systems*, Ph.D. Dissertation, Dept. of Computer Science, Purdue Univ., 1979

[18] Hevner, A.R. and Yao, S.B., *Query Processing in Distributed Database System*, IEEE Trans. on Softw. Eng. SE-5, 3, May 1979

[19] Luk, W.S. and Black, P.A., *On Cost Estimation in Processing a Query in a Distributed Database System*, Proc. IEEE COMPSAC 1981

[20] Luk, W.S. and Luk, L., *Optimizing Semi-Join Programs for Distributed Query Processing*, Proc. 2nd Int. Conf. on Data Bases, 1983

[21] Wong, E., *Retrieving Dispersed Data from SDD-1: A System for Distributed Databases*, Proc. 2nd Berkeley Workshop on Distributed Data Management and Computer Networks, 1977

[22] Yu, C.T. and Chang, C., *On the Design of a Query Processing Strategy in a Distributed Database Environment*, Proc. ACM SIGMOD Int. Conf. on Management of Data, 1983.

[23] Yu, C.T., Lam, K., Chang, C. and Chang, S., *Promising Approach to Distributed Query Processing*, Proc. 7th Berkeley Workshop on Distributed Data Management and Computer Networks, 1982

[24] Yu, C.T., Lam, K. and Ozsoyoglu, M.Z., *Distributed Query Optimization for Tree Queries*, Dept. of Information Eng., Univ. of Illinois at Chicago Circle, Jul. 1980

[25] Yu, C.T., Lilien, L., Guh, K., Templeton, M., Brill, D. and Chen, A., *Adaptive Techniques for Distributed Query Optimization*, Proc. 2nd Int. Conf. on Data Engineering, 1986

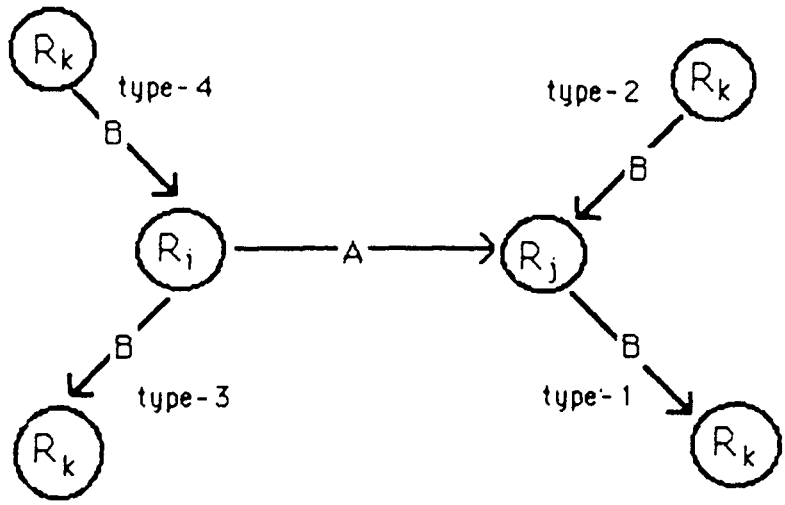

Fig. 3.1. interacting semijoins

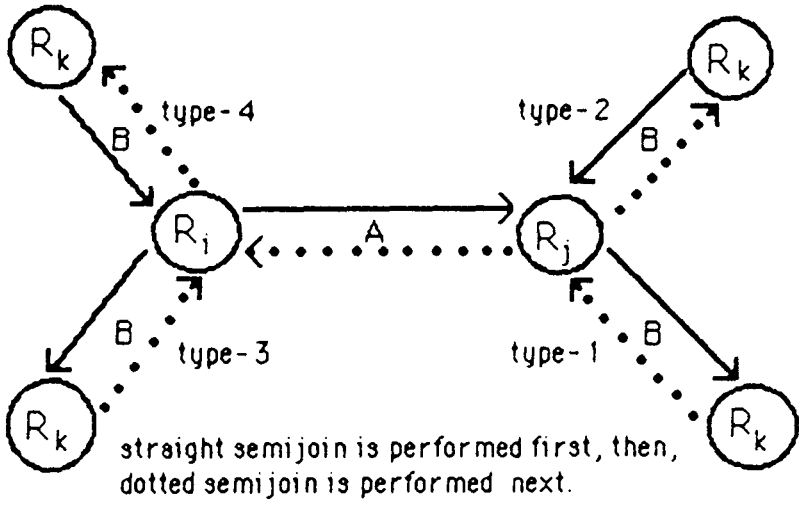

Fig. 3.2. interacting 2-way semijoins