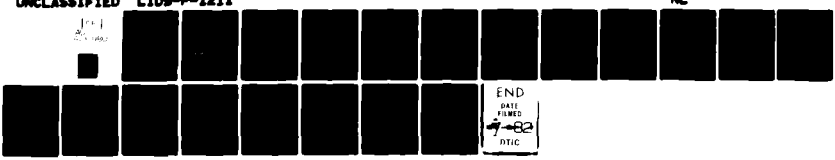
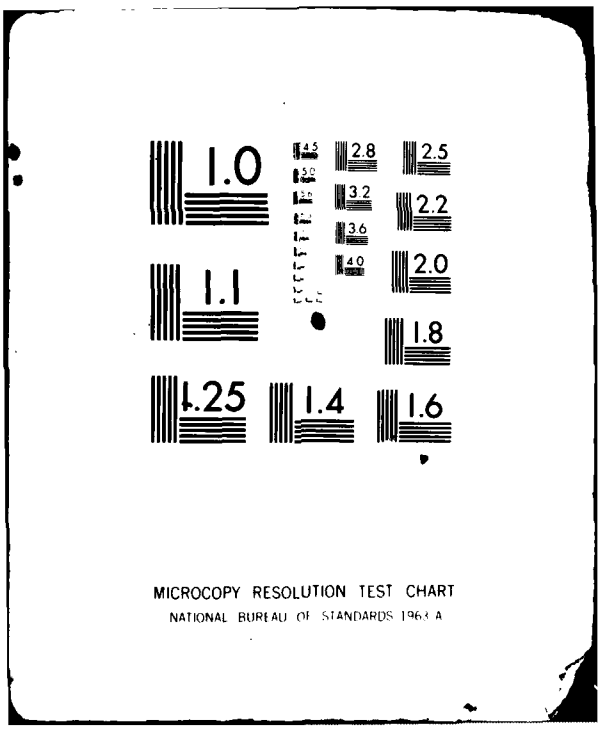
AD-A115 980 MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR INFORMA--ETC F/G 9/2
A MODEL FOR EQUI-JOIN QUERY PROCESSING IN DISTRIBUTED RELATIONA--ETC(U)
DEC 81 K HUANG, W B DAVENPORT N00014-77-C-0532
UNCLASSIFIED LIDS-P-1211 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

December 1981 LIDS-P-1211

A Model for Equi-join Query Processing in Distributed Relational Databases**

by

Kuan-Tsae Huang*

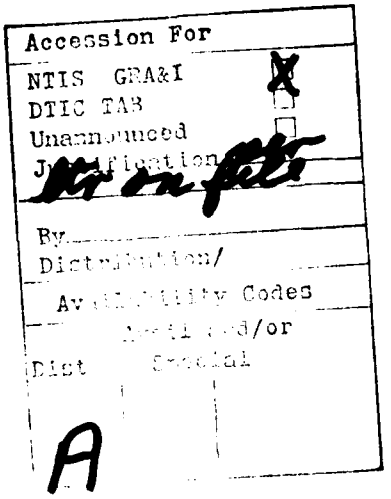
Wilbur B. Davenport Jr.

* The authors are with the Dept. of Electrical Engineering and Computer Science and the Laboratory for Information and Decision Systems at Mass. Institute of Technology, Cambridge, Mass. 02139.

** This research was conducted at the MIT Laboratory for Information and Decision Systems with support provided by the Office of Naval Research under contract ONR/N00014-77-C-0532.

Abstract

We develop a mathematical model to compute the minimum communication cost of a join-semijoin program for processing a given equi-join query. Some definitions and conditions uponwhich this paper is based are stated. We define a query processing graph for each equi-join query and characterize the set of join-semijoin programs which solve this query. A rule for estimating the size of the derived relation is derived. The parameters for estimating the size of derived relation form a consistent parameter system. With the assumption of communication cost dominance, the cost functions are linear in the size of data transmission. An optimization problem for distributed query processing is well formulated.



1. Introduction

Query processing in distributed relational databases corresponds to the translation of requests. formulated in a nonprocedural relational calculus like language. into a sequence of relational algebra operations which retrieve and update data stored in the distributed database management systems (DDBMSs).

Given a database schema $D=\{R_1, R_2, \ldots, R_n\}$, a query can usually be written in a number of alternative algebraic expressions. In particular, each query can be put in the following form:

$$Q = \Pi_{TL}\, \sigma_q (R_1 x R_2 x \ldots x R_n)$$

where TL contains the attributes in the answer relation; q is a predicate and each R is a relation. Usually, TL is referred to as the target-list and q as the qualification of a query. We shall assume all queries are expressed in this canonical form. denoted by Q=(q, TL).

In distributed query processing, the execution of a query involves data transmissions which may take significant time in comparison with the subquery and elementary operation execution times. We assume that the data communication costs dominate the local processsing costs. so the local processing cost of a query (e.g. costs of selection and projection) are negligible. In this paper, our objective is to minimize the total data transmission cost for processing a query.

For a query Q=(q,TL), let $\{ R_1, R_2, \ldots R_n \}$ be the set of relation schemas referenced by q and let X be the set of attributes appearing in q. Before processing the query, we can project each relation R_i over attributes $(X \cup TL) \cap R_i$. We then execute those subqueries which reference to only one local relation.

A query Q=(q, TL) is a conjunctive equi-join query if the qualification q is a conjunction of equi-join clauses of the form $(R_i . X = R_j.Y)$, where X and Y are subsets of attributes of R_i and R_j respectively.

In this paper, we restrict our study to a class of equi-join queries. Although it is a subset of complete relational calculus language, it is a rich and large class of queries in practice.

Data transmission is required when two relations that must be joined reside at different sites. To perform the join, one way is to move entire relation from one site to the other. The other way is to replace a join by semijoins and then perform a join. Assume R_1 and R_2 at different sites and we want to join R_1 and R_2 at the site of R_2. Using semijoin strategy, one can send the projection of R on its joining column to R_1's site and perform a semijoin to reduce R_1 by R_2 before sending R_1 to R_2's site. This will be a profitable tactic only whenever the projection of R_2 on its joining columns smaller than the amount by which R_1 is reduced by the semijoin.

Prior works in distributed query processing [WONG77,GBWRR80,CHIU79,HY79] were either limited to strategies of performing semijoins first and then joins or without a consistent parameter system to estimate the size of derived relation.

In this paper, we first state some definitions and conditions uponwhich this paper based. We then define a query processing graph for each equi-join query and derive a theorem about the set of join-semijoin programs which solve this query. Next. we define a rule for estimating the size of derived relation and prove that the parameter system we defined is consistent. With the definition of cost functions, we develop a mathematical model to compute the minimum communication cost of a join-semijoin program for processing a given query.

2. Query Processing Model

A query Q specified by a qualification q over the relations R_1, R_2,...,R_n, and by a target list TL can be decomposed into a set of operations $\{ p_1, p_2 \ldots p_l \}$ which will produce the answer to the query, where $p_i \in \mathcal{A}$, the set of relational algebra operators. In general, a query can be decomposed into several different executing sequences which will produce the same answer. We call such an executing sequence a strategy. Let S(Q) denote the set of strategies which answer the query Q. The goal of the problem is to minimize the overall cost of executing this query Q. We can formulate this problem as

$$\underset{P \in S(Q)}{\text{MIN}} \; f(P, D[0]) = \sum_{i=1}^{\ell-1} f_i(p_i, D[i])$$

s.t. $P = p_1 p_2 \cdots\cdots p_\ell$

$D[i+1] = p_i(D[i])$

$D[0]$ is the initial database state

2.1 Definitions and Assumptions

We assume that the distributed database management system DDBMS consists of a collection of interconnected computers S_1, S_2 $\cdots\cdots S_n$ at different sites. Each computer, known as a node in the network, contains a DBMS. Data are logically viewed in the relational model. By the univeral relation interpretation [BFMMUY 81], we assume that each site only consists of one relation.

Data transmission in the network is via communication links. We assume that the transmission cost to send one byte of data between any two sites i & j is known and equal to c_{ij}. So the cost function of transmitting data of volume V between two sites i & j is a linear function $C_{ij}(V) = c_{ij} * V$. We assume that all possible subqueries involving data at single site are preprocessed; this we call local processing. The effort of local processing is to reduce the amount of data that needs further processing. After local processing, the following parameters of the qeury can be defined.

n = number of sites (i.e. relations) in the remaining query

$d_i = |R_i|$, number of attributes in site R_i

$X_{ij} = R_i \cap R_j$, the set of attributes of joining domains between R_i & R_j

r_i = number of tuples in relation R_i

$w(A)$ = the width of data item of attribute A in relation R_i

$s_i = r_i * \sum_{A \in R_i} w(A)$, the size of the relation R_i

$w(X_{ij}) = \sum_{A \in X_{ij}} w(A)$

Next. we define several terminologies used in this paper.

Let (R_i, r_i) and (R_j, r_j) be two relation states and $X \subseteq R_i \cap R_j$.

Definition:

The equi-join of R_i and R_j on X, denoted by $R_i \overset{X}{\bowtie} R_j$, is $\{t \mid t$ is a tuple over $R_i \cup R_j$ such that $t[R_i] \in r_i \wedge t[R_j] \in r_j\}$.

The semijoin of R_i and R_j on X, denoted by $R_i \overset{X}{|\!\ltimes} R_j$, equals $R_i |\!\ltimes| R_j[X]$. Equivalently it equals $\{t_i \mid t_i \in r_i \wedge (\exists t_j \in r_j, \ni t_i[X] = t_j[X])\}$.

The natural join of R_i & R_j, denoted by $R_i |\!\bowtie\!| R_j$, is the join of R_i & R_j on $R_i \cap R_j$.

The natural semijoin of R_i & R_j, denoted by $R_i |\!\ltimes R_j$, is the semijoin of R_i & R_j on $R_i \cap R_j$.

Note that the join (resp. semijoin) operator is weaker than the natural-join (resp. natural semi-join) operator in that:

$R_i \,|X|\, R_j \subseteq R_i \,|X|_X\, R_j \quad \forall\, X \subseteq R_i \cap R_j$.

$R_i \,|X\, R_j \subseteq R_i \,|X_X\, R_j \quad \forall\, X \subseteq R_i \cap R_j$.

Definition:

A qualification q is called sub-natural iff for each clause $R_i.A_{iK}=R_j.A_{jl}$, $A_{iK}=A_{jl}$.

q is called natural iff the converse holds as well. i.e. for all relation schemas R_i and R_j, and for all $A_K \in R_i \cap R_j$, $R_i.A_K=R_j.A_K$ is a clause of q.

Definition:

Given a database schema $D=\{R_1, R_2, \ldots, R_n\}$, a query is called an natural join query (NJQ) (resp. sub-natural join query, SNJQ) iff there exists a natural qualification (resp. sub-natural qualification) for it and $TL \subseteq U(D)$.

As shown in [BG 81], any query Q=(q,TL) with an equijoin qualification q and a target list TL can be efficiently transformed into an equivalent natural join query. Instead of the class of equijoin queries, EQJ, we shall construct the query processing model in terms of the class of natural join query, NJQ.

In DDBMS, we define two types of directed operators.

Definition:

1. $<|X|_{ij}$ (or $R_i<|X|\, R_j$) is the distributed natural join operator which send R_j to R_i and perform natural join of R_i and R_j at R_i's site.

2. $<|X_{ij}$ (or $R_i<|X\ R_j$) is the directed natural semijoin operator which project $X=R_i \cap R_j$ over R_j, send the results to R_i and perform the join of R_i and the result at R_i's site. (i.e. $R_i\ |X|\ \pi_X R_j$ at R_i's site).

Note that $|X|>_{ij} = R_i\ |X|>R_j$ and $X|>_{ij} = R_i\ X|>R_j$ are similarily defined. One can use them interchangeablely. The semijoin operation only reduces the relation state without changing relation schema.

Definition:

A join-semijoin program $P=p_1 p_2 \ldots p_l$ is a sequence of distributed natural join and distributed natural semijoin operators.

A natural join qualification q with final node at R can be done by sending all relations R_i, $i\neq 1$, to R_1 and performing $R_1\ |X|R_2|X|\ldots|X|R_n$ at node R_1. So $R_2|X|>R_1$, $R_3|X|>R_1$, ... $R_n|X|>R_1$ or its permutation are join-semijoin programs of this qualification q.

2.2 Query Processing Graph

We define a processing graph of a qualification over a database schema $D=\{R_i\}_{i=1}^{n}$ to be a graph with two type of edges, $<V_q, A_q, B_q>$. V_q is the set of node which is equal to D. A_q is a set of semijoin edges which is $\{a_{ij}=(R_i,R_j)\in A_q |$ if $R_i\cap R_j\neq\emptyset$ and $R_i \not\subseteq R_j\}$. We denote it by i ---><--- j with one arrow on the edge. $B_q = V_q\ xV_q = \{b_{ij} |\forall\ i\neq j\}$ is the set of join edges. We denote it by i

--->>--- j with two arrows on the edge.

Note that if $R_i \wedge R_j = \phi$, then we can not perform a semijoin between R_i and R_j . So a is not a semijoin edge. If $R_i \leq R_j$, then $R_i = R_i \wedge R_j$. The semijoin of $R_i \ltimes R_j$, $R_i X|> R_j$, is the same as join of R_i to R_j, $R\ |X|\ R$. This operation is covered by join edge b_{ij}.

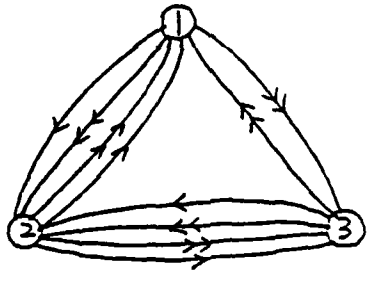
Example:

$R_1 = \{ A_1, A_2, A_3, A_4 \}$

$R_2 = \{ A_2, A_3, A_5, A_6 \}$

$R_3 = \{ A_5, A_7, A_8 \}$

The processing graph of the natural join qualification q is:



Without lost of generality, from now on we assume that the final node of a query is node 1.

Definition:

A join-semijoin program P is correct for a natural join qualification q if after executing the program P, the final node will have a new relation $R'_1 = R_1\ |X|\ R_2\ |X|\ldots|X|\ R_n$.

Lemma 1:

Any directed path of edges in B_q from R_{K_0} to R_{K_ℓ}, b_{K_0,K_1}, b_{K_1,K_2} ..., $b_{K_{\ell-1},K_\ell}$ will form a relation $R_{K_0}|X|R_{K_1}|X|...|X|R_{K_\ell}$ in node R_{K_ℓ}.

Proof: WE prove this by induction on the length of the path. If $l=1$, then the path is b_{K_0,K_1}. After this operation we will have $R'_{K_1} \leftarrow R_{K_0}|X|R_{K_1}$. By induction assumption on $l-1$, $R'_{K_{\ell-1}} \leftarrow R_{K_0}|X|R_{K_1}|X|...|X|R_{K_{\ell-1}}$. In the case of l, for the first $l-1$ edges of this path, $R'_{K_{\ell-1}} = R_{K_0}|X|R_{K_1}|X|...|X|R_{K_{\ell-1}}$ by induction assumption. After performing $b_{K_{\ell-1},K_\ell}$, we will have $R'_{K_\ell} = R'_{K_{\ell-1}}|X|R_{K_\ell} = R_{K_0}|X|R_{K_1}|X|...|X|R_{K_\ell}$.

Lemma 2:

Any directed spanning tree toward node 1 of edges in B_q will form $R_1|X|R_2|X|...|X|R_n$ at node 1.

Proof: By lemma 1, any path from R_K to R_1 will form a new relation at node 1 by joing all relations in this path. A directed spanning tree will contain each node i exactly once. If we execute the pathes toward node 1 one by one, we will result the relation $R_1|X|R_2|X|...|X|R_n$ at node 1.

Theorem 1:

Let $Q=(q,TL)$ be a natural join query and $TL=R_1 U ... U R_n$. Let P be a join-semijoin program for q, then P is correct iff there exist a subset of the set $\{b_{ij}\}$ in P which forms a inversely directed spanning tree toward node R_1.

Proof: IF: Natural semijoin only reduce a relation state without losing any correct data and do not change the relation schema. So after performing the sequence of joins in the directed spanning tree toward R_1, we get the $R_1 |X| \ldots |X| R_n$ at node R_1. Any other join operation does not change the state. This implies P is correct.

ONLY IF: For each semijoin operation a_{ij} in P, $R_i \cap R_j \neq \emptyset$ and $R_i \subseteq R_j$. So performing the semijoin operations do not move the full relation state from node R_i to node R_j. We still need to perform a join to move the full table of R_i to R_j. If there do not exist a subset of $\{b_{ij}\}$ in P which form a directed spanning tree toward node R_1, then some information of those nodes which do not have a path toward R_1 will lose some information.

From theorem 1, we know the set of correct programs of the NJQ qualification is the set of join-semijoin programs such that there exists a directed spanning tree toward R out of the set of join edges in P. We denote the set of correct programs by $\mathcal{P}$. The distributed query processing problem becomes to find a program $p \in \mathcal{P}$ with minimum communication cost. For a program p, if we change the order of the sequence of operations, the total communication cost will be different. The set of correct programs is very large. In fact, after executing one operation in P, it will change the number of rows and columns of some relations. This change then affects the communication cost of next operation. So the communication cost of one operation will depend

on the previous subsequence of operations.

2.3 Estimate the Size of the Derived Relations

In order to compare the communication cost of query processing strategies. it is very important to have a method to estimate the size of a relation after one operation. Also the system for estimation of the size of the derived relation must be consistent in the sense that if two sequences of operations will produce the same results. the estimated sizes of the result according the two sequences of operations must be the same.

We introduce the notion of semijoin reducibility and join reducibility of R_i to R_j, denoted by α_{ij} and β_{ij} respectively, for each pair of relations R_i and R_j. Where $0 \leq \alpha_{ij} \leq 1$ and $0 \leq \beta_{ij} \leq 1$. The interpretation of the semijoin reducibility α_{ij} of R_i to R_j is that the percentage of rows of R_j will be reduced after performing the semijoin R_i X|> R_j. At stage t, if the number of rows of R_j is $r_j[t-1]$ and the semijoin reducibility of R_i to R_j is $\alpha_{ij}[t-1]$, the the number of rows of R_j after performing semijoin R_i X|> R_j will be reduced to $r_j[t] = r_j[t-1] * (1-\alpha_{ij}[t-1])$. Note that the semijoin reducibility of R_i to R_j is not equal to the semijoin reducibility of R_j to R_i and $\alpha_{ii}[t]=0$ for all t. The interpretation of the join reducibility of R_i to R_j is that after performing join R_i |X|> R_j, the number of rows of new relation R_i|X|R_j at site j will be $r_j[t] = r_i[t-1] * r_j[t-1] * (1- \alpha_{ij}[t-1]) * (1- \alpha_{ji}[t-1]) * (1- \beta_{ij}[t-1])$. This is because the affect of join R_i |X|> R_j is equivalent to perform the semijoins

R_i X|> R_j and R_jX|> R_i and then to perform the join of R_i to R_j. Both semijoin reducibilities and join reducibility affect the number of rows of the new relation. The join reducibility of R_j to R_i is the same as the join reducibility of R_i to R_j. i.e. $\beta_{ij}[t] = \beta_{ji}[t]$. Also $\beta_{ii}[t]=0$ for all t. For this paper, we assume the set of reducibilities $\{ \alpha_{ij}, \beta_{ij} \}$ can be known in advance by some statistical measurement.

Since the number of rows and columns of a relation will be changed after one operation, the reducibilities of this relation with other relations will be changed too. We define how the reducibilities will be changed after one operation. Assume the database state before the operation p_t be $D=\{R_1[t-1],...R_n[t-1]\}$, the number of rows of each relation $R_i[t-1]$ be $r_i[t-1]$, and the semijoin and join reducibilities of $R_i[t-1]$ to $R_j[t-1]$ are $\alpha_{ij}[t-1]$ and $\beta_{ij}[t-1]$.

If the operation p_t at stage t is a_{ij}, i.e. R_iX|>R_j, then the database schema will remain the same and only the number of rows of relation $R_j[t]$ will be changed to equal to $r_j[t-1] * (1-\alpha_{ij}[t-1])$ and the number of rows of all other relations will remain the same.

Since this semijoin operation a_{ij} will reduce the number of rows of R_j, the semijoin reducibility of R_j with all other relations R_K will be reduced too. The semijoin reducibility $\alpha_{hK}[t]$ of $R_h[t]$ with all other $R_K[t]$ will be changed as the following rules:

$$\alpha_{hk} = \begin{cases} 0 & h=i,\ h=j \\ \alpha_{hk}[t-1] & h=j,\ k=i \\ \alpha_{hk}[t-1]+\alpha_{ik}[t-1]-\alpha_{hk}[t-1]\,\alpha_{ik}[t-1] & h=j,\ k\neq i,j \\ \alpha_{hk}[t-1]+\alpha_{hi}[t-1]-\alpha_{hk}[t-1]\,\alpha_{hi}[t-1] & h\neq i,j,\ k=j \\ \alpha_{hk}[t-1] & \text{otherwise} \end{cases}$$

The semijoin operation does not affect the join reducibilities. So all join reducibilities at this stage stay the same as last stage. i.e.

$\beta_{ij}[t] = \beta_{ij}[t-1]$ for all h and k.

If the operation p_t at stage t is b_{ij}, i.e. $R_i |X| > R_j$, then the database schema at node j, $R_j[t]$ will change to $R_i[t-1] \cup R_j[t-1]$, and the relation state at site j will be $R_i[t-1] |X| R_j[t-1]$. The number of rows of $R_j[t]$, $r_j[t]$, equal to $r_j[t-1] * r_i[t-1] * (1-\alpha_{ij}[t-1]) * (1-\alpha_{ji}[t-1]) * (1-\beta_{ij}[t-1])$. All other relation states will remain the same. Because this is a join operation, the semijoin reducibilities and join reducibilities will be affected.

The semijoin reducibility of $R_h[t]$ to $R_k[t]$ will be changed as follows:

$$\alpha_{hk}[t] = \begin{cases} 0 & h=i,\ k=j \\ \alpha_{hk}[t-1] & h=j,\ k=i \\ \alpha_{hk}[t-1]+\alpha_{ih}[t-1]-\alpha_{hk}[t-1]\alpha_{ik}[t-1] & h=j,\ k\neq i,j \\ \alpha_{hk}[t-1]+\alpha_{hi}[t-1]-\alpha_{hk}[t-1]\alpha_{ki}[t-1] & h\neq i,j,\ k=j \\ \alpha_{hk}[t-1] & \text{otherwise} \end{cases}$$

The join reducibility of $R_h[t]$ to $R_k[t]$ will be changed as follows:

$$\beta_{jk}[t] = \beta_{jk}[t-1] + \beta_{ik}[t-1] - \beta_{jk}[t-1]\beta_{ik}[t-1] \qquad \forall k \neq i, j$$

$$\beta_{ij}[t] = \begin{cases} 1 - \dfrac{1}{Y_i[t-1]} & \\ 1 & \text{if } Y_i[t-1] = 0 \end{cases}$$

2.4 Consistent Parameter System

We say that a parameter system is consistent if this parameter system will produce the same estimates of the size of the results when the two programs really produce tha same results. We now define a parameter system which we shall use to estimate the size of the derived relations has the parameters $\{ r_i , \alpha_{ij} , \beta_{ij} \}$.

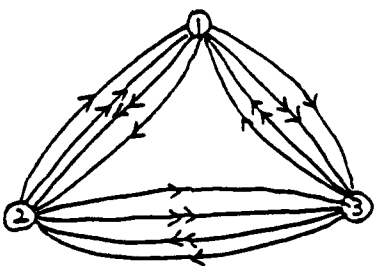
Theorem:

The parameter system $\{r_i [t], \alpha_{ij}[t], \beta_{ij}[t]\}$ we defined above is consistent.

Proof: This is because a semijoin operation and join operation can only affect the size of derived relations once and the rules of updating reducibilities have reflecting this fact. So the order of the operations does not affect the estimate of the size of the derived relation.

Example:

Suppose we have three relations R_1, R_2 and R_3, where $R_i \cap R_j \neq \phi$ and $R_i \not\subseteq R_j \; \forall$ i,j; and suppose $\alpha_{ij}[0]$, β_{ij} [0] are given. then the processing graph will be:



Let $P_1 = b_{31}\, b_{21}$ and $P_2 = a_{12}\, a_{13}\, b_{23}\, b_{31}$. The two programs will produce the same results $R_1[0] \mid X \mid R_2[0] \mid X \mid R_3[0]$ at site 1. By the rules of estimating the size of the derived relation, the estimate sizes of $R_1[0] \mid X \mid R_2[0] \mid X \mid R_3[0]$ derived by these two programs will be the same. Which is $\prod_{i=1}^{3} r_i[0] * \prod_{\substack{i,j=1 \\ i \neq j}}^{3} (1 - \alpha_{ij}[0]) * \prod_{\substack{i,j=1 \\ i<j}}^{3} (1 - \beta_{ij}[0])$.

2.5 Problem Formulation

In order to write down the mathematical formulation of distributed query optimization problem, we need to know the cost function of each operation. From our assumption of linear cost function before, we can write down the cost function of this type of operations at stage t. The projection of R_i over $R_i \cap R_j$ may result the number of rows of R_i smaller after compressing. Here we ignore this fact. We assume the number of rows after projection on R_i over $R_i \cap R_j$ equal to:

$$W_{I_i} = \text{Min} \{ r_i[t], \prod_{A \in X_{ij}} |\text{dom}(A)| \}$$

The cost of operation a_{ij} will be

$$\text{Cost}(a_{ij}) = c_{ij} * (w_{I_i} * \sum_{A \in X_{ij}} w(A))$$

and the cost of operation b_{ij} will be

$$\text{Cost}(b_{ij}) = c_{ij} * (r_i[t] * \sum_{A \in R_i} w(A)).$$

Based on the distributed query processing model we developed, the formulation of the disributed query optimization

problem is as follow:

INPUT:

1. a distributed database schema $D=\{ R_1[0],\ldots,R_n[0]\}$
2. the width $w(A)$ of each attribute A in $U(D)$
3. the number of rows $r_i[0]$, of each relation R_i
4. the semijoin reducibility $\alpha_{ij}[0]$ of each pair of relations R_i & R_j with $\alpha_{ii}[0]=0$
5. the join reducibility $\beta_{ij}[0]$ of each pair of relations R_i & R_j with $\beta_{ii}[0]=0$ and $\beta_{ij}[0]=\beta_{ji}[0]$.

OBJECTIVE:

Find an optimal join-semijoin program to solve the natural join program.

Let $P=p_1\ p_2\ \ldots p_\ell$, then the problem is to minimize $\sum_{x=1}^{l} cost(p_x)$ according the rules of updating the parameters and cost functions defined above.

3. Conclusion

We have developed a mathematical model for distributed query processing problem for a class of equi-join queries. We also define rules for estimating the size of derived relation. The parameter system based on those rules is consistent. The future research will be to develop algorithms for solving this problem. The reason for difficulty in solving this problem is that computing the cost of one operation depend on the size of derived relation which is the result of previous operations. A special

case of this problem has been shown [HUA81] to have NP-complete complexity. An efficient optimal algorithm for this problem seems unlikely.

REFERENCES

[BFMMUY 81]
Beeri, C. etc. Properties of acyclic database schemas. 13th. ACM STOC 335-362 May 1981.

[BG 81]
Bernstein, P. A. and Goodman, N. The power of natural semijoins. SIAM J. of Comput. 10(4). Nov 1981.

[CHIU 79]
Chiu, D. M. Optimal query interpretation for distributed databases. Ph. D. thesis, Harvard Univ. 1979.

[GBWRR 80]
Goodman, N., Bernstein. P. A., Wong, E., Reeve, C. L., Rothnie, J. B. Query Processing in SDD-1. Computer Cooperation of America report, 1980.

[HUA 81]
Huang, K. T. Computation Complexity of Distributed Query Processing Problem. (in preparation)

[HY 79]
Hevner, A. R. and Yao, S. B. Query processing in distributed database systems. IEEE Trans. on Software Engineering, SE-5(3):177-187, May 1979.

[WONG 77]
Wong, E. Retrieving Dispersed data in SDD-1: A systems for distributed databases. Proc. 1977 Berkeley Workshop on Distributed Data Management and Computer Network. May 1977.