Optimizing Entity Join Queries by Extended Semijoins in a Wide Area Multidatabase Environment*

Pauray S.M. Tsai and Arbee L.P. Chen

Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan 300, R.O.C.
e-mail: alpchen@cs.nthu.edu.tw

Abstract

In this paper, we consider processing *entity join* queries in a wide area multidatabase environment where the query processing cost is dominated by the cost of data transmission. An entity join operation "integrates" tuples representing the same entities from different relations in which inconsistent data may exist. The semijoin technique has been successfully used in a distributed database system to reduce the cost of data transmission. However, it cannot be directly applied to process the entity join query. In this paper, an extension of the traditional semijoin, named *extended semijoin* is proposed to reduce the cost of data transmission for entity join query processing in a wide area multidatabase environment.

Keywords: Multidatabase, query optimization, entity join, extended semijoin, inconsistent data.

1 Introduction

Because of the increasing need for data sharing among multiple databases, the development of multidatabase systems has been considered as an important research issue. A *wide area multidatabase system* consists of autonomous component database systems in a wide area network, which may be heterogeneous.

Many research results about heterogeneous DBMSs are concerned with the problems of schema integration and interoperability, while very few papers discuss query optimization. The difficulty of query optimization for the heterogeneous DBMSs lies in the complexity of heterogeneities existing among the component databases. However, query optimization is an important research problem, which greatly affects the system performance.

The semijoin technique has been used to reduce the cost of data transmission [1, 2], which can minimize the query response time or total time. None of the above query optimization algorithms consider the data inconsistency problem. Du et al. [6] proposed a query optimization strategy to resolve the heterogeneities of component query optimizers in heterogeneous DBMSs. However, the data and schema conflicts were ignored as well.

In [9], we extended our results on probabilistic query processing [12] to consider joining two relations on their incompatible keys. This approach identifies the same entities from different relations considering various data and schema conflicts. We have formally defined the *entity join* operator, named EJ operator, which joins two relations on their compatible/incompatible keys in [10]. Besides, some transformation rules were proposed to transform a global query into local subqueries to be executed in multiple sites to lower the query processing cost.

In this paper, we consider processing entity join queries in a wide area multidatabase environment where the query processing cost is dominated by the cost of data transmission. In the distributed database system, the semijoin technique has been successfully used to reduce the cost of data transmission. However, the traditional semijoin cannot be directly applied to process the entity join query. We extend a traditional semijoin to handle data inconsistency such that it can be used to reduce the cost of data transmission in the wide area multidatabase environment.

This paper is organized as follows. Our previous research results are described in Section 2. In Section 3, we give an example to illustrate the motivation of the research. The determination of *feasible extended semijoins* for an EJ operation is discussed in Section 4. Finally, we conclude in Section 5.

*Appeared in Proc. IEEE International Conference on Parallel and Distributed Systems (1994). This work was partially supported by the Republic of China National Science Council under Contract No. NSC 84-2213-E-007-007.

2 Previous Research Results

2.1 Entity Join Operation

In [9], the probabilistic technique and the concept of *probabilistic partial value* [12] are used to identify and integrate the "same" entities in different databases. A probabilistic partial value is a set of possible values with a probability assigned to each possible value, in which exactly one possible value is the true value. The probabilistic partial value is denoted as $[u_1^{x_1}, u_2^{x_2}, ..., u_m^{x_m}]$, where $\{u_1, u_2, ..., u_m\}$ is the set of possible values, x_i is the associated probability of u_i and $\sum_{i=1}^{m} x_i = 1$. The entity join operation on relations R_1 and R_2 is denoted as $R_1 \overset{\bullet}{\bowtie} R_2$, which joins the "same" entities in R_1 and R_2. Two cases are considered in the following.

2.1.1 Compatible Keys

Consider the relations in Figure 1, where relation **TC** in one database records the data for students who belong to the tea club, and relation **TW** in another database records the data for students who take Technical Writing. Assume the key attributes *id* in both relations are compatible and two tuples in **TC** and **TW**, which have the same *id* value, represent the same entity. For example, the query "Find all twenty-year-old sophomores who belong to the tea club and take Technical Writing" can be expressed as $\sigma_{(age=20)\wedge(class=sophomore)}(\mathbf{TC} \overset{\bullet}{\bowtie} \mathbf{TW})$. Figure 2 shows the result of entity join on **TC** and **TW**. The final query result is depicted in Figure 3. The column *poss* in Figure 3 denotes the possibilities of the answer tuples.

2.1.2 Incompatible Keys

Consider the relations in Figure 4, where relation **Teacher** in one database records the personal data of teachers at X University, and relation **Consultant** in another database records the data of consultants in the computer science division of Y Company. The key attribute *id* in **Teacher** represents the identification number for teachers at X University while that in **Consultant** represents the identification number for consultants at Y Company. It is meaningless to compare the *id* value in **Teacher** with the *id* value in **Consultant**. In other words, the keys in **Teacher** and **Consultant** are incompatible.

An approach [9] is proposed to compare the values in the common attributes (e.g., attributes *name*, *degree* and *age* in **Teacher** and **Consultant**) and related attributes (e.g., the attributes *department* and *speciality* in **Teacher** and **Consultant**, respectively) to determine whether two tuples represent the same entity. There are two rules derived in [9] useful for the consideration of query optimization.

1. In some cases, the values of a common attribute in t_1 and t_2 are not allowed inconsistent if t_1 and t_2 are considered the same entity. Such a common attribute is called a *dominant attribute*. Conversely, a common attribute is called a *nondominant attribute* if the values of the common attribute in t_1 and t_2 are allowed to be different when these two tuples are considered the same entity.
   Rule 1: If one of the dominant attributes has different values in t_1 and t_2, t_1 and t_2 are considered representing different entities.

2. Consider a nondominant attribute a_i. Let the values of a_i in t_1 and t_2 be v_1 and v_2, respectively, the *difference distance* between v_1 and v_2, denoted $DD(v_1, v_2)$, be a value representing the difference between v_1 and v_2, and the *maximum difference distance* for attribute a_i, denoted $MDD(a_i)$, be a value which $DD(v_1, v_2)$ cannot exceed if t_1 and t_2 are considered to represent the same entity.
   Rule 2: If the difference distance between values of $t_1.a_i$ and $t_2.a_i$ is greater than $MDD(a_i)$, then the two entities represented by t_1 and t_2 are different.

2.2 Local Processing for Entity Join Queries

Since inconsistent data may exist in different databases, cares need to be taken for the local processing. For example, consider the query $\sigma_{(age=20)\wedge(class=sophomore)}(\mathbf{TC} \overset{\bullet}{\bowtie} \mathbf{TW})$ on relations in Figure 1. The query cannot be transformed into $(\sigma_{(age=20)\wedge(class=sophomore)}\mathbf{TC}) \overset{\bullet}{\bowtie} (\sigma_{(age=20)\wedge(class=sophomore)}\mathbf{TW})$ for local processing, even though *age* and *class* are the common attributes of relations **TC** and **TW**. Observe the tuple (30, Mary, 20, sophomore, 4863) in **TC** and the tuple (30, Mary, 20, CS, junior) in **TW**. These two tuples represent the same entity, but their *class* values are inconsistent. The entity represented by these two tuples can be considered to be the integrated tuple (30, Mary, 20, CS, [sophomore$^{1/2}$,junior$^{1/2}$], 4863) as shown in Figure 2, and it is a possible answer tuple for the original query as shown in Figure 3. However, the query result obtained by executing the transformed query $(\sigma_{(age=20)\wedge(class=sophomore)}\mathbf{TC}) \overset{\bullet}{\bowtie} (\sigma_{(age=20)\wedge(class=sophomore)}\mathbf{TW})$ is empty. Therefore, the transformed query is not equivalent to the original one because some tuples representing real world entities which may satisfy the query condition are eliminated in the process of local processing.

TC

id	name	age	class	phone
2	John	21	junior	4335
30	Mary	20	sophomore	4863
41	Jane	22	senior	3776
46	Paul	18	freshman	2375
53	John	22	senior	6447
55	Tony	19	freshman	6450

TW

id	name	age	dept	class
30	Mary	20	CS	junior
43	Joe	23	CE	senior
46	Paul	18	CS	freshman
55	Tony	20	CS	sophomore
57	Joe	19	CS	freshman

Figure 1: Relations **TC** and **TW** in two different databases.

id	name	age	dept	class	phone
30	Mary	20	CS	[sophomore$^{1/2}$, junior$^{1/2}$]	4863
46	Paul	18	CS	freshman	2375
55	Tony	[19$^{1/2}$, 20$^{1/2}$]	CS	[freshman$^{1/2}$, sophomore$^{1/2}$]	6450

Figure 2: The result of $\mathbf{TC} \overset{\bullet}{\bowtie} \mathbf{TW}$.

In [10], an approach which correctly transforms a global query with EJs into local subqueries to be executed in multiple sites is proposed. The queries considered are of the form $\sigma_P(R_1 \overset{\bullet}{\bowtie} R_2)$, where R_1 and R_2 represent relations in different databases and P represents the conjunction of a set of simple selection predicates $P_1, P_2, ..., P_n$. A *simple selection predicate* is defined as of the form *attr* **op** C, where *attr* represents a relation attribute, **op** denotes an operator, and C represents a constant. The *associated attribute* for predicate P_i is the *attr* component of P_i. Let S_p be the set $\{P_1, P_2, ..., P_n\}$ and S_c the set $\{P_i | P_i \in S_p$ and a_i is a common attribute for R_1 and $R_2\}$. Predicates in S_c are called *common selection predicates* for R_1 and R_2. Besides, a predicate is called a *locally-executable predicate* (*LEP*) for relation R_k, $k = 1, 2$, if it can be locally executed on R_k without affecting the correctness of the query result. Two transformation rules obtained by the rules in Section 2.1.2 are described as follows.

- **Transformation 1**: By Rule 1, if the associated attribute of the common selection predicate P_i is a dominant attribute, query $\sigma_{P_1 \wedge ... \wedge P_i \wedge ... \wedge P_n}(R_1 \overset{\bullet}{\bowtie} R_2)$ can be equivalently transformed into
  $\sigma_{P_1 \wedge ... \wedge P_{i-1} \wedge P_{i+1} \wedge ... \wedge P_n}((\sigma_{P_i} \ R_1) \overset{\bullet}{\bowtie} (\sigma_{P_i} \ R_2))$.
- Assume P_i is a common selection predicate, and the associated attribute a_i is a nondominant attribute. Obviously, predicate P_i cannot be a *LEP* for R_1 and R_2. In the following, we consider a relaxed form of P_i, which can be shown to be a *LEP* for R_1 and R_2 [10], assuming a_i a numerical attribute.
  - **case 1**: If P_i represents $a_i = v$, the relaxed constraint for P_i is defined as
    $RC(P_i)$: $v - MDD(a_i) \leq a_i \leq v + MDD(a_i)$.
  - **case 2**: If P_i represents $a_i \geq v$, the relaxed constraint for P_i is defined as
    $RC(P_i)$: $a_i \geq v - MDD(a_i)$.
  - **case 3**: If P_i represents $a_i \leq v$, the relaxed constraint for P_i is defined as
    $RC(P_i)$: $a_i \leq v + MDD(a_i)$.

  The notion of the relaxed constraint can be easily extended to that for nonnumerical attributes.
- **Transformation 2**: By Rule 2, if the associated attribute of a common selection predicate P_i is a nondominant attribute, query $\sigma_{P_1 \wedge ... \wedge P_i \wedge ... \wedge P_n}(R_1 \overset{\bullet}{\bowtie} R_2)$ can be equivalently transformed into
  $\sigma_{P_1 \wedge ... \wedge P_i \wedge ... \wedge P_n}((\sigma_{RC(P_i)} \ R_1) \overset{\bullet}{\bowtie} (\sigma_{RC(P_i)} \ R_2))$.

3 Motivation

In a distributed database system, a join clause "R_i joins R_j on attribute a," denoted by $R_i \bowtie_a R_j$, can be considered having two associated semijoins: $R_i \ltimes_a R_j$ and $R_j \ltimes_a R_i$. The semijoin $R_i \ltimes_a R_j$ is implemented as follows: First, we project R_j on the join attribute a, then ship this projection to the site of R_i and finally perform the join with R_i. The selectivity of $R_i \ltimes_a R_j$ is defined as the number of distinct values in $R_j.a$ divided by the number of distinct values in the domain of a, assuming the domains of $R_i.a$ and $R_j.a$ are the same. The traditional semijoin cannot be directly applied to process the entity join query since inconsistent data may cause tuples representing the same entities to be eliminated.

Consider the query **Teacher** $\overset{\bullet}{\bowtie}$ **Consultant** on relations in Figure 4. The attribute *age* is a common attribute used to identify the same entities in **Teacher** and **Consultant**. Let $MDD(age)$ be 2. For **Consultant**, only tuples shown in Figure 5 remain after the traditional semijoin **Consultant** $\ltimes_{age}$ **Teacher**. Observe the tuple (7004,John,CS,MS,26) in **Teacher** and the tuple (101,John,CN,MS,25,B) in **Consultant** of Figure 4. These two tuples may represent the same entity. However, the tuple (101,John,CN,MS,25,B) is eliminated by **Consultant** $\ltimes_{age}$ **Teacher**. Therefore, these two tuples cannot involve in the final EJ processing.

In the following section, the traditional semijoin

id	name	age	dept	class	phone	poss
30	Mary	20	CS	[sophomore$^{1/2}$, junior$^{1/2}$]	4863	0.5
55	Tony	[19$^{1/2}$, 20$^{1/2}$]	CS	[freshman$^{1/2}$, sophomore$^{1/2}$]	6450	0.25

Figure 3: The result of $\sigma_{(age=20)and(class=sophomore)}(\mathbf{TC} \overset{\bullet}{\bowtie} \mathbf{TW})$.

Teacher

id	name	department	degree	age
7001	Mary	CS	~	30
7002	Paul	EE	PhD	29
7003	John	CS	PhD	29
7004	John	CS	MS	26
7005	Lina	EE	PhD	34
7006	Paul	CS	PhD	30

Consultant

id	name	specialty	degree	age	city
101	John	CN	MS	25	B
102	Jose	DB	MS	36	~
103	James	DB	BS	24	A
104	Mary	CN	MS	29	B
105	John	AI	MS	26	B
106	Dick	DB	BS	24	A
107	Tony	IP	MS	28	~
108	Paul	~	MS	25	B
109	Mary	AI	PhD	30	B
110	Paul	DB	PhD	30	~

Figure 4: Relations **Teacher** and **Consultant** in two different databases.

technique is extended to reduce the cost of data transmission for processing entity join queries without affecting the correctness of the query result.

4 Preprocessing EJs by Extended Semijoins

In this section, we consider extending the semijoin technique to preprocess EJs. Some notations are expressed as follows.

- $dom[R_i.a]$: represents the domain of attribute a in relation R_i. We also use $dom[a]$ to represent the domain of attribute a in a relation if it does not cause ambiguity.
- $card(S_j)$: represents the number of distinct values (or tuples) in the set S_j, where S_j can be R_i, $dom[R_i.a]$ or $\Pi_a(R_i)$.

Reconsider the query $\sigma_P(R_1 \overset{\bullet}{\bowtie} R_2)$ as described in Section 2.2. Assume $\sigma_P(R_1 \overset{\bullet}{\bowtie} R_2)$ can be equivalently transformed into $\sigma_{P_{01} \wedge P_{02} \wedge ... \wedge P_{0k}}((\sigma_{P_{11} \wedge P_{12} \wedge ... \wedge P_{1n}} R_1) \overset{\bullet}{\bowtie} (\sigma_{P_{21} \wedge P_{22} \wedge ... \wedge P_{2m}} R_2))$ in the local processing. Let the selectivities for predicates $P_{11}, P_{12}, ..., P_{1n}$, $P_{21}, P_{22}, ...$, and P_{2m} be $q_{11}, q_{12}, ..., q_{1n}, q_{21}, q_{22}, ...$, and q_{2m}, respectively, and R_1' and R_2' be the results of executing $(\sigma_{P_{11} \wedge P_{12} \wedge ... \wedge P_{1n}} R_1)$ and $(\sigma_{P_{21} \wedge P_{22} \wedge ... \wedge P_{2m}} R_2)$, respectively. Assume attributes in a relation are independent for the discussion. Then, $card(R_1')$ and $card(R_2')$ can be computed as $card(R_1) \times \Pi_{j=1}^{n} q_{1j}$ and $card(R_2) \times \Pi_{j=1}^{m} q_{2j}$, respectively. By the results of local processing on R_1 and R_2, we consider how to preprocess $R_1' \overset{\bullet}{\bowtie} R_2'$ by extended semijoins.

The determination of *feasible extended semijoins* for an EJ operation is discussed in Section 4.1. In Section 4.2, we consider the minimum number of data items needed to be transmitted for performing an feasible extended semijoin. The computation of the selectivity for a feasible extended semijoin is presented in [11]. Once the selectivities of extended semijoins are obtained, the traditional semijoin algorithm can be applied to find an optimal execution plan to process a query with EJs.

4.1 Feasible Extended Semijoin

An extended semijoin is *feasible* if the correctness of the query result is not affected after the extended semijoin is performed. We consider the case where the keys of R_1' and R_2' are compatible/incompatible in the following.

1. the keys of R_1' and R_2' are compatible
   Let a_k be the keys of R_1' and R_2'. $R_1' \overset{\bullet}{\bowtie} R_2'$ can be considered having two associated extended semijoins denoted as $R_1' \overset{\bullet}{\ltimes}_{a_k} R_2'$ and $R_2' \overset{\bullet}{\ltimes}_{a_k} R_1'$. Assume the domains of the key attributes are the same. The way for implementing $R_1' \overset{\bullet}{\ltimes}_{a_k} R_2'$ is the same as that for implementing the traditional semijoin $R_1' \ltimes_{a_k} R_2'$. Obviously, both $R_1' \overset{\bullet}{\ltimes}_{a_k} R_2'$ and $R_2' \overset{\bullet}{\ltimes}_{a_k} R_1'$ are feasible.

2. the keys of R_1' and R_2' are incompatible
   Since the keys of R_1' and R_2' are incompatible, the values in the common attributes and related attributes are compared to determine whether two tuples in R_1' and R_2' represent the same entity. We consider the extended semijoins of $R_1' \overset{\bullet}{\bowtie} R_2'$ based on the common attributes. Assume the domains of a common attribute in R_1'

id	name	specialty	degree	age	city
104	Mary	CN	MS	29	B
105	John	AI	MS	26	B
109	Mary	AI	PhD	30	B
110	Paul	DB	PhD	30	~

Figure 5: The result of **Consultant** $\times_{age}$ **Teacher**.

and R_2' are the same. Let a_i be a common attribute of R_1' and R_2', t_1 be a tuple of R_1' and t_2 a tuple of R_2'. Two cases are considered as follows.

- **case 1**: a_i is a dominant attribute
  $R_1' \overset{\bullet}{\bowtie} R_2'$ can be considered having two associated extended semijoins, denoted $R_1' \overset{\bullet}{\ltimes}_{a_i} R_2'$ and $R_2' \overset{\bullet}{\ltimes}_{a_i} R_1'$, based on attribute a_i. The way for implementing $R_1' \overset{\bullet}{\ltimes}_{a_i} R_2'$ and $R_2' \overset{\bullet}{\ltimes}_{a_i} R_1'$ is the same as that for implementing $R_1' \ltimes_{a_i} R_2'$ and $R_2' \ltimes_{a_i} R_1'$, respectively. $R_1' \overset{\bullet}{\times}_{a_i} R_2'$ and $R_2' \overset{\bullet}{\times}_{a_i} R_1'$ are feasible because if the values of a_i in t_1 and t_2 are different, then t_1 and t_2 are considered representing different entities.

- **case 2**: a_i is a nondominant attribute
  Since a_i is nondominant, tuples t_1 and t_2 may represent the same entity if $DD(t_1.a_i, t_2.a_i)$ is not greater than $MDD(a_i)$. Let $\Pi_{a_i}(R_1')$ be $\{b_1, b_2, ..., b_k\}$ and $MDD(a_i)$ be m. Two cases are considered below.

  (a) a_i is not an associated attribute of any selection predicate
  We define the *extension* $E(b_r)$ of b_r, $1 \le r \le k$, as

  $$E(b_r) = \{x | x \in dom[a_i] \text{ and } DD(x, b_r) \le m\}.$$

  Extension $E(b_r)$ implies that if the value of $t_1.a_i$ is b_r, then the tuples in R_2' which have the values of a_i belonging to $E(b_r)$ may join with t_1 for $R_1' \overset{\bullet}{\bowtie} R_2'$. That is, if $t_2.a_i \notin E(b_r)$, t_1 and t_2 represent different entities by Rule 2 in Section 2.1.2. The *extension* of projection $\Pi_{a_i}(R_1')$ is defined as $\bigcup_{1 \le r \le k} E(b_r)$. The extended semijoin $R_j \overset{\bullet}{\ltimes}_{E(a_i)} R_l$ is implemented as follows: First, relation R_l is projected over the attribute a_i, then we ship this projection to the site of R_j and compute the extension of the projection at the site of R_j, finally the join of R_j and the extension of this projection is performed. Therefore, $R_1' \overset{\bullet}{\bowtie} R_2'$ can be considered having two feasible extended semijoins, denoted $R_1' \overset{\bullet}{\ltimes}_{E(a_i)} R_2'$ and $R_2' \overset{\bullet}{\ltimes}_{E(a_i)} R_1'$, based on attribute a_i.

  (b) a_i is the associated attribute of selection predicate P_i
  Since R_1' and R_2' are the results of local processing on R_1 and R_2, respectively, each tuple in R_1' or R_2' satisfies the relaxed constraint $RC(P_i)$ but may not satisfy P_i. Therefore, the extension $E(b_r)$ of b_r is defined as

  $$E(b_r) = \begin{cases} \{x | x \in dom[a_i] \text{ and } DD(x, b_r) \le m \} \\ \text{if } b_r \text{ satisfies } P_i \\ \{x | x \in dom[a_i] \text{ and } DD(x, b_r) \le m \\ \text{and } x \text{ satisfies } P_i\} \text{ otherwise} \end{cases}$$

  where $1 \le r \le k$. Assume tuples t_1 and t_2 are considered representing the same entity. Extension $E(b_r)$ implies that if $t_1.a_i$ does not satisfy P_i and the join tuple of t_1 and t_2 may satisfy the query, then $t_2.a_i$ must satisfy P_i. $R_1' \overset{\bullet}{\ltimes}_{E(a_i)} R_2'$ and $R_2' \overset{\bullet}{\ltimes}_{E(a_i)} R_1'$ are feasible extended semijoins for $R_1' \overset{\bullet}{\bowtie} R_2'$ as in case (a) except that the definition of $E(b_r)$ is different.

4.2 Transmitting Needed Data Items for Performing Extended Semijoins

We consider the minimum number of data items needed to be transmitted for performing an extended semijoin in the case where a_i is the associated attribute of selection predicate P_i. Let $dom[a_i]$ be $\{x_1, x_2, ..., x_n\}$ and $MDD(a_i)$ be m.

4.2.1 The case where predicate P_i is "$a_i = v$"

Consider the minimum number of data items needed to be transmitted for the extended semijoin $R_2' \overset{\bullet}{\times}_{E(a_i)} R_1'$ in the case where P_i represents "$a_i = v$." Assume $\Pi_{a_i}(R_1')$ is $\{b_1, b_2, ..., b_j\}$. If $card(\Pi_{a_i}(R_1')) = 1$, the unique value in $\Pi_{a_i}(R_1')$ is transmitted. If $card(\Pi_{a_i}(R_1')) = j$, $j > 1$, then

1. if $v \in \{b_1, b_2, ..., b_j\}$, only v needs to be transmitted to the site of R_2'. This is because $E(v)$ dominates the extension of $\Pi_{a_i}(R_1')$. In other words, $E(v)$ is $\{max(v - m, x_1), ..., v, ..., min(v + m, x_n)\}$, which is equal to the extension of $\Pi_{a_i}(R_1')$;

2. if $v \notin \{b_1, b_2, ..., b_j\}$, only one of the elements in $\{b_1, b_2, ..., b_j\}$ needs to be transmitted to the site of R_2'. This is because all the extensions of elements in $\{b_1, b_2, ..., b_j\}$ are the same, namely, $E(b_r) = \{v\}, 1 \le r \le j$.

4.2.2 The case where predicate P_i is "$a_i \ge v$"

Consider the minimum number of data items needed to be transmitted for the extended semijoin

$R_2' \overset{\bullet}{\times}_{E(a_i)} R_1'$ in the case where P_i represents "$a_i \geq v$." $RC(P_i)$ is $a_i \geq v - m$. All the values in $\Pi_{a_i}(R_1')$ fall in the range of $[v-m, x_n]$, where x_n is the maximum number in $dom[a_i]$. For the discussion, assume $x_n \geq v + m$. Let G_1, G_2 and G_3 be $\{v-m, v-m+1, ..., v-1\}$, $\{v, v+1, ..., x_n - m\}$ and $\{x_n - m + 1, x_n - m + 2, ..., x_n\}$, respectively, and S_1, S_2 and S_3 be the sets of elements in $\Pi_{a_i}(R_1')$ belonging to G_1, G_2 and G_3, respectively.

$$E(e) = \begin{cases} \{v, ..., e+m\} & \text{if } e \in G_1 \\ \{e-m, ..., e+m\} & \text{if } e \in G_2 \\ \{e-m, ..., x_n\} & \text{if } e \in G_3 \end{cases}$$

Let $E(S_i)$ be $\bigcup_{e \in S_i} E(e)$, $i = 1, 2$ or 3. If $card(\Pi_{a_i}(R_1')) = 1$, the unique value in $\Pi_{a_i}(R_1')$ is transmitted. If $card(\Pi_{a_i}(R_1')) = j$, $j > 1$, consider the data items in S_1, S_2 and S_3, respectively, needing to be transmitted.

1. Assume S_1 is $\{v_1, v_2, ..., v_i\}$, where v_i is the maximum number in S_1. For the elements in S_1, only v_i needs to be transmitted to the site of R_2' since $E(S_1) = E(v_i)$.

2. Assume S_3 is $\{u_1, u_2, ..., u_j\}$, where u_1 is the minimum number in S_3. For the elements in S_3, only u_1 needs to be transmitted to the site of R_2' since $E(S_3) = E(u_1)$.

3. Assume S_2 is $\{w_1, w_2, ..., w_l\}$. The elements in S_2 are enumerated in an increasing order. For any two successive elements r and s in S_2, $E(r)$ and $E(s)$ may overlap or be exclusive. If we can find a set S_2' such that $S_2' \subseteq S_2$ and $E(S_2') = E(S_2)$, then only the subset S_2' needs to be transmitted to the site of R_2'. The algorithm shown below generates a minimum subset S_2' for S_2 such that $E(S_2')$ is equal to $E(S_2)$.

For the case where predicate P_i is "$a_i \leq v$," the minimum number of data items needing to be transmitted can be considered in a similar way.

ALGORITHM
Input: set S_2 (assume S_2 is $\{w_1, w_2, ..., w_l\}$)
Output: set S_2'
begin
  $S_2' \leftarrow \emptyset$
  $w_{l+1} \leftarrow 0$
  if $l \leq 2$ **then** $S_2' \leftarrow S_2$
  else
    $x \leftarrow w_1$
    $y \leftarrow w_2$
    $z \leftarrow w_3$
    $i \leftarrow 3$
    repeat
      $i \leftarrow i + 1$
      if $(z - x - 1) \leq 2m$ **then**
        if $E(y) \subset (E(x) \cup E(z))$
          $y \leftarrow w_{i-1}$
          $z \leftarrow w_i$
        else if $E(y) \not\subset (E(x) \cup E(z))$
          $S_2' \leftarrow S_2' \cup \{x\}$
          $x \leftarrow w_{i-2}$
          $y \leftarrow w_{i-1}$
          $z \leftarrow w_i$
        end if
      until $i > l$
      $S_2' \leftarrow S_2' \cup \{x, y\}$
  end if
end

5 Conclusion

In this paper, we consider processing entity join queries in a wide area multidatabase environment where the query processing cost is dominated by the cost of data transmission. In the distributed database system, the semijoin technique has been successfully used to reduce the cost of data transmission. However, the traditional semijoin cannot be directly applied to process the entity join query due to the existence of inconsistent data. In this paper, the semijoin technique is extended to preprocess entity joins to reduce the cost of data transmission and final join processing. Feasible extended semijoins for an EJ are determined and the minimum number of data items needed to be transmitted for performing an feasible extended semijoin is discussed.

References

[1] P.M.G. Apers, A.R. Hevner, and S.B. Yao, Optimization algorithms for distributed queries, *IEEE Transactions on Software Engineering,* 9 (1), (1983) pp. 57-68.

[2] P. Bernstein, N. Goodman, E. Wong, C. Reeve and J. Rothnie, Query Processing in a System for Distributed Databases (SDD–1), *ACM Transactions on Database Systems,* 6 (4), (1981) pp. 602-625.

[3] W. Du, R. Krishnamurthy and M.C. Shan, Query Optimization in Heterogeneous DBMS, *Proc. VLDB,* (1992) pp. 277-291.

[4] P.S.M. Tsai and A.L.P. Chen, Querying uncertain data in heterogeneous databases, *Proc. IEEE Third International Workshop on Research Issues on Data Engienring: Interoperability in Multidatabase Systems* (1993) pp. 161-168.

[5] P.S.M. Tsai and A.L.P. Chen, A Localized Approach to Query Optimization in Heterogeneous Database Systems, to appear in *Journal of Information Science and Engineering.*

[6] P.S.M. Tsai and A.L.P. Chen, Optimizing Entity Join Queries in a Wide Area Multidatabase Environment, *Technical Report, National Tsing Hua University* (1994).

[7] F.S.C. Tseng, A.L.P. Chen, and W.P. Yang, Answering heterogeneous database queries with degrees of uncertainty, *Distributed and Parallel Databases: an International Journal,* Kluwer Academic Publishers, (1993) pp. 281-302.