Copyright © 1980, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

DYNAMIC RE-MATERIALIZATION: PROCESSING DISTRIBUTED QUERIES USING REDUNDANT DATA

by

Eugene Wong

Memorandum No. UCB/ERL M80/26

3 November 1980

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

DYNAMIC RE-MATERIALIZATION: PROCESSING DISTRIBUTED QUERIES USING REDUNDANT DATA

Eugene Wong

Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California, Berkeley, California 94720

ABSTRACT

In this paper an approach to processing distributed queries that makes explicit use of redundant data is proposed. The basic idea is to focus on the dynamics of materialization, defined as the collection of data and partial results available for processing at any given time, as query processing proceeds. In this framework the role of data redundancy in maximizing parallelism and minimizing data movement is clarified. What results is not only the discovery of new algorithms but an improved framework for their evaluation.

Research sponsored by the Honeywell Corporation under a grant from the Corporate Computer Science Center, by the Army Research Office under Grant DAAG29-78-G-0186, and by the Air Force Office of Scientific Research under Grant 78-3596.

1. Introduction

In this paper we propose a new formulation for the problem of processing queries in a distributed database system. By such a system we mean a collection of autonomous processors, communicating via a general communication medium, and accessing separate and possibly overlapping fragments of a database. The user's view of data is to be an integrated whole, both fragmentation and redundancy being invisible. Geographical dispersion, though sometimes present, is not an essential ingredient of such a system, and the range of systems so encompassed includes not only the classical geographically distributed databases but also configurations that are in effect database machines. The problem of distributed execution of queries is common to all these systems.

In the query processing algorithm designed for the SDD-1 distributed database management system [WONG 77], an irredundant subset of the database is used during the execution of any single query. No effort was made to exploit the possible existence of multiple copies either to maximize parallel operations or to minimize data moves. A related and somewhat hidden characteristic inherent in the SDD-1 algorithm is that parallel processing is opportunistic rather than deliberate.

These characteristics were recognized in [EPST 78] where the emphasis fell heavily on maximizing parallelism. The algorithm proposed there, and implemented for the distributed version of INGRES, achieves a high degree parallelism by partitioning one relation among the processing sites and replicating all other needed relations at every site. We shall call this the F-R (fragment and replicate) algorithm. For a query referencing many relations, the degree of data replication and the resulting communication cost to achieve this replication may be prohibitive. Thus, the F-R algorithm is best applied to pieces of a many-variable query, one at a time, each with only two or three variables. Experience of using the F-R algorithm in the distributed version of INGRES [EPST 80] indicates that the procedure of splitting a query before applying the F-R algorithm is not an easy one to optimize.

It is time then, to seek a new formulation of the problem of distributed query processing that puts the issue of redundancy and parallelism into better focus. One such formulation was suggested by some recent work on database partitioning in a distributed system [WONG 80].

II. Partioning a Database

Let $\mathcal{D}$ denote the database as viewed by a user. Let M_i denote the data residing at processing site i. We assume that $\bigcup_i M_i = \mathcal{D}$ and call $M = \{M_i\}$ a materialization of $\mathcal{D}$. Suppose that the database designer is free to choose M. How should he choose?

Among the major issues to be resolved is that of redundancy. Intuitively, the cost of redundancy is paid on updates and benefit accrued on retreval. What we need is a conceptual framework to make this precise. Let Q denote a collection of queries on a database $\mathcal{D}$. We shall say that a materialization M of $\mathcal{D}$ is self-sufficient (relative to Q) if for every q in Q and for every i there exists a local query q_i on M_i such that

$$\text{Result } (q,\mathcal{D}) = \bigcup_i \text{Result } (q_i,M_i)$$

Self-sufficiency means that no inter-communication is necessary to process q. The only data movement needed is a final one to collect the results.

For two materializations M and M' denote M > M' if $M_i \supseteq M_i'$ for every i. A locally sufficient M is said to be minimally redundant if there exists no M' < M (other than M itself) that is locally sufficient. Minimum redundancy means that data reduction at local sites cannot preserve local sufficiency.

Suppose we assume that it always takes longer to process a query when there are more data. Then, in terms of both retrieval and update, it is better to have minimal redundancy than not. Thus, minimally redundant materializations represent a desirable class of partitions for the database.

III. Query Processing by Dynamic Re-Materialization

In terms of the concepts that we have introduced for database partitioning, query processing can be viewed as a dynamic process of changing materializations. Let q be a single query. Let $M_i^{(t)}$ denote the data at i available and selected for processing q at any stage t of processing. $M^{(t)} = \{M_i^{(t)}\}$ will be called the materialization at t. Any algorithm for distributed query processing can be represented as a sequence of states: $(q^{(t)},M^{(t)})$, t = 0,1,2,...,N. The terminal state $(q^{(N)},M^{(N)})$ is required to be locally sufficient and to satisfy the condition

$$\bigcup_i \text{Result } (q_i^{(N)},M_i^{(N)}) = \text{Result } (Q,\bigcup_i M_i^{(0)})$$

In other words, from the terminal state only local processing and gathering up of results are needed to complete processing. Transition between two successive states $(q^{(t)},M^{(t)})$ and $(q^{(t+1)},M^{(t+1)})$ occurs as a result of data movement and/or local processing. A transition will be called a redistribution if only data movement is involved, and a local derivation if: (a) $(q^{(t+1)},M^{(t+1)})$ is derived from $(q^{(t)},M^{(t)})$ by local processing, (b) $M^{(t+1)} < M^{(t)}$, and (c) Result $(q^{(t+1)},M^{(t+1)})$ = Result $(q^{(t)},M^{(t)})$.

For any terminal state $(q^{(N)},M^{(N)})$ a measure of the parallelism that it affords is given by

$$\tau(q^{(N)},M^{(N)}) = \max_i \text{ (time to process } q_i^{(N)} \text{ on } M_i^{(N)})$$

The cost to reach $(q^{(N)},M^{(N)})$ can be expressed as

$$C(\{q_t,M_t\}) = C_0(N) + \sum_{t=1}^{N} T_t((q^{(t-1)},M^{(t-1)}),\ (q^{(t)},M^{(t)})$$

where $C_0(N)$ is the cost of resynchronization between transitions and T_t is the cost of making the $t^{\underline{th}}$ transition.

If a compatible scale for τ and C is known, the problem can then be stated as one of optimal control. Even though the optimization problem is unlikely to be solved in any general sense, it provides a framework that allows algorithms proposed on heuristic grounds to be evaluated.

IV. Strategies Based on an Initially Feasible Solution

Let $q^{(0)} = q$ be the query to be processed and $M^{(0)}$ the data initially available for processing q. We say (q,M) is an initially feasible solution if it is a "locally sufficient redistribution" of $(q^{(0)},M^{(0)})$, i.e., M is locally sufficient and derivable from $M^{(0)}$ by moving data. The cost of using such a strategy consists of several components, of which we assume the following to be dominant:

(a) $C(M^{(0)},M)$ = cost of moves

(b) $\tau(q,M)$ = cost of terminal parallel processing

We shall say a strategy is of the IFS type if it consists of the following steps:

(1) One seeks a $(q^{(1)},M^{(1)})$ that is a "local derivation" of $(q^{(0)},M^{(0)})$.

(2) If no such local derivation can be found, one seeks an initially feasible solution (q,M).

(3) One seeks to improve (q,M) by replacing the one-transition strategy $(q,M^{(0)}) \rightarrow (q,M)$ by a "short" sequence of transitions. Perform the first transition $(q,M^{(0)}) \rightarrow (q^{(1)},M^{(1)})$ in the sequence.

(4) Iterate, with $(q^{(1)},M^{(1)})$ replacing $(q^{(0)},M^{(0)})$.

Both the SDD-1 and F-R algorithms are variations of IFS algorithms. In the SDD-1 case, the initially feasible solution (q,M) is restricted to be not merely locally sufficient but single-site sufficient. That is, there exists a site j such that q can be processed entirely on M_j. The choice for M in the F-R algorithm is to replicate every relation but one, which is obviously locally sufficient, not only for a given q but for all q. It seems clear that to qualify for selection as the initial choice as a feasible solution, (q,M) should be at least "non-inferior" with respect to the pair of costs $(C(M^{(0)},M), \tau(q,M))$. That is, there exists no initially feasible solution that is equal or better in both (c,τ) and strictly better in at least one. Neither the SDD-1 nor the F-R algorithm guarantees this in general. Indeed, the choice is often poor in these cases.

Example 4.1 Consider a conceptual schema given by:

person (socsec, name, state-of-res)

corp (cid, cname, state-of-inc)

emp (socsec, cid, position, salary)

Suppose that there are two sites with their local schemas given by the following view definition statements:

```
range of p is person
range of c is corp
range of e is emp
define person 1 (p.all) where (p. state-of-res = "NY")
define corp 1 (c.all) where (p.state-of-inc = "NY")
define emp 1(e.all) where (e.salary ≤ 25000)
define person 2 (p.all) where (p.state-of-res ≠ "NY")
define corp 2 (c.all) where (c.state-of-inc ≠ "NY")
define emp 2 (e.all) where (e.salary > 25000)
```

Now consider a query

```
retrieve (p.name) where (p.socsec = e.socsec)
                    and (c.cid = e.cid)
                    and (c.name = "IBM")
```

We begin with the materialization.

$M_1^{(0)}$ = (person 1, corp 1, emp 1)

$M_2^{(0)}$ = (person 2, corp 2, emp 2)

Processing the clauses that involve only local operations, we get

$M_1^{(1)}$ = (P1, C1, E1)

$M_2^{(1)}$ = (P2, C2, E2)

where

Pk = person k projected on (socsec, name)

Ck = corp k restricted to (name = "IBM") and projected on (cid)

Ek = emp k projected on (socsec, cid)

Now assume the following statistics for these relations

relation	# tuples	tuple width in bytes
P1	1000	29 (9,20)
P2	1000	29 (9,20)
C1	1	5
C2	0	5
E1	1000	14
E2	1000	14
C1 (cid) E2	100	14
C1 (cid) E1	1000	14

The initial feasible solution in the SDD-1 algorithm would consist of site 1 as the final processing site and

M = {move P2 and E2 to site 1}

which entails moving 43 K bytes of data. On the other hand the F-R algorithm would yield a materialization

M_1 = (P1, C1, E1, E2)
M_2 = (P2, C1, E1, E2)

with

M_1 = {move E2 to site 1}
M_2 = {move C1 and E1 to site 2}

which in this example corresponds to the $M^{(2)}$ that minimizes communication cost C(M) and entails moving 28 K bytes of data. M_2 can be reduced by joining C1 to E1 and moving the join instead of E1 (1405 bytes). M_1 can be reduced by moving C1 to site 2, joining C1 with E2, and moving the join. The resulting sequence of materialization would appear as follows, where $\bowtie$ denotes join:

$M^{(1)}$ = {(P1, C1, E1), (P2, E2)}
$M^{(2)}$ = {(P1, C1, E1), (P2, C1, E2)}
$M^{(3)}$ = {(P1, E1 $\bowtie$ C1), (P2, E2 $\bowtie$ C1)}
$M^{(4)}$ = {(P1, E1 $\bowtie$ C1, E2 $\bowtie$ C1), (P2, E1 $\bowtie$ C1, E2 $\bowtie$ C1)}

and $M^{(4)}$ is now locally sufficient. The total amount of data moved is 2805 bytes, and no more processing is involved than either the F-R or the SDD-1 algorithm. For our example, the strategy that we have found is just about the best possible over a wide range of relative costs for communication and local processing.

V. Repeated-Join Strategies

The database-partition problem suggests the following class of query processing strategies: Consider a relational database $\mathcal{D} = \{R_1, R_2, \ldots, R_m\}$ where R_k are relations. We shall say a query q is admissible if it is a finite repetition of "restriction", "projection" and "join" on the relations in $\mathcal{D}$. We shall say an admissible q is elementary if it involves at most one join. Now, suppose that for any $\mathcal{D}$ we know how to find a "good" materialization $M(\mathcal{D})$ that is locally sufficient for all elementary queries. Then, we can construct a query processing algorithm as follows: Construct a sequence

$$\mathcal{D} = \mathcal{D}^{(0)}, \mathcal{D}^{(1)}, \ldots\ldots, \mathcal{D}^{(N)}$$

$$q^{(0)}, q^{(1)}, \ldots\ldots, q^{(N)}$$

such that $q^{(N)} = q$, and for each t $q^{(t)}$ is an elementary query on $\mathcal{D}^{(t)}$ and $\mathcal{D}^{(t+1)} \subset \{\mathcal{D}^{(t)}, \text{Result}(q^{(t)}, \mathcal{D}^{(t)})\}$. Since for each $\mathcal{D}^{(t)}$ we know how to find a materialization $M(\mathcal{D}^{(t)})$ that is locally sufficient for all elementary queries, $M(\mathcal{D}^{(t)})$ is a fortiori locally sufficient for $q^{(t)}$. The repeated-join algorithm consists of repeating for each t the following steps.

(a) Execute $q^{(t)}$ on $M(\mathcal{D}^{(t)})$

(b) To obtain $\mathcal{D}^{(t+1)}$, add $\text{Result}(q^{(t)}, \mathcal{D}^{(t)})$ to $\mathcal{D}^{(t)}$ and eliminate the relations no longer needed in processing q.

(c) Construct $M(\mathcal{D}^{(t+1)})$

How good this algorithm is depends on :

(1) Whether we can construct $M(\mathcal{D}^{(t)})$ as claimed, and

(2) the cost in resynchronization and data movement in making the transition $M(\mathcal{D}^{(t)}) \rightarrow M(\mathcal{D}^{(t+1)})$.

Our preliminary study suggests that the efficacy of this class of algorithms is enhanced if we augment the semantics of the relational model and use the semantics to restrict the class of admissible queries.

Roughly speaking, the semantic augmentation that we undertake corresponds to distinguishing between entities and relationships [CHEN 76, WONG 79], but we shall define the semantics strictly in terms of the constructs of the relational model.

First, we classify the sets that serve as domains of the relations in the database into identifier and value. D is an identifier domain if and only if there is a unique relation E_D such that the elements of D are in one-to-one correspondence with the tuples of E_D. We shall say D is the key of E_D.

Every relation must have at least one identifier domain. A relation will be called an e-relation (entity) if it has a key, and an r-relation (relationship) otherwise.

Example 5.1

person (* socsec, name, state-of-res)

corp (* cid, cname, state-of-inc)

emp (socsec, cid, position, salary)

where underscore indicates an identifier domain and * indicates a key. Clearly, "person and "corp" are e-relations and "emp" is an r-relation.

Now suppose that we limit the admissible data manipulation operations to the following:

(a) Restriction - boolean condition on values

(b) Projection

(c) Join - on an identifier domain D

(d) Closure

Note that admissible joins are limited. For example, the join

person (state-of-res = state-of-inc) corp

would be an inadmissible operation, but the following operation is admissible:

$$(\text{person } (\overset{\text{socsec}}{\bowtie}) \text{ emp } (\overset{\text{cid}}{\bowtie}) \text{ corp}) \text{ [state-of-res = state-of-inc]}$$

Let $\overset{D}{\ltimes}$ denote the semijoin operator defined in [BERN 79]. That is, $A \overset{D}{\ltimes} B$ is the projection on A of the join $A \overset{D}{\bowtie} B$. The following proposition gives a condition for local sufficiency in terms of the semijoin.

Proposition 5.1 Let $\mathcal{D}$ be a collection of relations. Let M be a materialization of $\mathcal{D}$ such that to each e-relation E corresponds a unique E(k) in M_k such that

$$E = \bigcup_k E(k)$$

Then, M is locally sufficient for all elementary queries if for every $R \in \mathcal{D}$ and every identifier domain D in R

$$R \overset{D}{\ltimes} E_D(k) \in \text{closure } (M_k) \text{ for every } k. \tag{5.1}$$

Proof: For R and S in $\mathcal{D}$, we can write

$$R \overset{D}{\bowtie} S = \bigcup_k (R \overset{D}{\ltimes} E_D(k) \overset{D}{\bowtie} (S \overset{D}{\ltimes} E_D(k))$$

Since projection and restriction commute with union, the proposition is proved. ¤

Condition (5.1) provides a simple means for testing the local sufficiency of a materialization M for elementary queries. Further, if M fails the test, (5.1) provides a means for augmenting M to make it locally sufficient. As such, (5.1) makes the repeated-join algorithm work. At each step t in the algorithm, to construct $M(\mathcal{D}^{(t+1)})$, we only need to distribute enough of the result $(q^{(t)},\mathcal{D}^{(t)})$ so that

$$(5.2) \qquad \text{result } (q^{(t)},\mathcal{D}^{(t)}) \overset{D}{\ltimes} E_D(k) \in \text{closure } (M_k(\mathcal{D}^{(t+1)}))$$

for every k and every D.

Example 5.2 Take the schema in example 5.1, and consider the same query as in example 5.1. We have

$$\mathcal{D} = \{\text{person, corp, emp}\}$$

$$q = \text{person} \overset{\text{socsec}}{\ltimes} (\text{emp} \overset{\text{cid}}{\ltimes} (\text{corp(name="IBM")})) \text{ [name]}$$

Define person k and corp k as in example 4.1, but define

$$\text{emp k} = (\text{emp} \overset{\text{socsec}}{\ltimes} \text{person k}) \cup (\text{emp} \overset{\text{cid}}{\ltimes} \text{corp k})$$

Take the initial materialization to be

$$M^{(0)} = \{(\text{person k, corp k, emp k}),\ k = 1,2\}$$

Then $M^{(0)}$ is locally sufficient for all elementary queries on $\mathcal{D}$.

Now, take

$$\mathcal{D}^{(0)} = \mathcal{D} = \{\text{person, corp, emp}\}$$
$$q^{(0)} = (\text{emp} \overset{\text{cid}}{\ltimes} (\text{corp(name="IBM")})) \text{ [socsec]}$$

Here, we have no need to distinguish $q^{(0)}$ and its result, Hence, we can write

$$\mathcal{D}^{(1)} = \{\text{person}, q^{(0)}\}$$
$$q = q^{(1)} = (\text{emp} \overset{\text{cid}}{\ltimes} q^{(0)}) \text{ [name]}$$

As in example 4.1, assume that

$$\text{corp 2 (name="IBM")} = \phi$$

Hence, $q_2^{(0)} = \phi$ and

$$q^{(0)} = q_1^{(0)}$$

To satisfy (5.1), we can take

$$M_1^{(1)} = (\text{person 1}, q_1^{(0)})$$

$$M_2^{(1)} = (\text{person 2}, q_1^{(0)})$$

which requires moving $q_1^{(0)}$ to site 2. Alternatively, we can take

$$M_2^{(1)} = (\text{person 2}, q_1^{(0)} \overset{\text{socsec}}{\propto} \text{person 2})$$

which would entail first moving (person 2) [socsec] to site 1 and then moving $q_1^{(0)} \overset{\text{socsec}}{\propto}$ person 2 to site 2. However, the double move would be ovbiated by storing at each site an index for the distribution of identifiers.

For a given q, the sequence $q^{(n)}$ is by no means unique, and the optimization problem is to choose $q^{(n)}$ so as to minimize cost, however cost is defined.

VI. Conclusion

In this paper we propose a new approach to distributed query processing. This approach focuses on how the data available at each site change as processing proceeds. We believe that issues of parallelism and redundancy are rendered clearer by this approach. Our immediate goal is not so much to find better algorithms, but to provide a conceptual framework in which new classes of algorithms can be formulated in a natural way.

REFERENCES

[BERN79] Bernstein, P.A. & Chiu, D.W., "Using Semi-joins to Solve Relational Queries," Computer Corp. Am. Tech. Rep. CCA-01-79. To appear in JACM.

[EPST76] Epstein, R.; Stonebraker, M.; Wong, E; "Distributed Query Processing in a Relational Data Base System", Proc. 1978 ACM-SIGMOD Conference on Management of Data, Austin, Texas, June 1978.

[EPST80] Epstein, R. & Stonebraker, M; "Analysis of Distributed Data Base Processing Strategies," 1980. International Conference on Very Large Data Bases, Montreal, October 1980.

[HELD75] Held, G.D., M.R. Stonebraker, & E. Wong; "INGRES-A Relational Data Base System," Proc. NCC vol. 44, 1975.

[HEVN78] Hevner, A. & Yao, S.B., "Query Processing on a Distributed Data Base," Proceedings of the Third Berkeley Workshop on Distributed Data Management and Computer Networks, LBL-7953 UC-32, Lawrence Berkeley Laboratory, Berkeley, California, August 1978.

[WONG77] Wong, E.; "Retrieving Dispersed Data from SDD-1; A System for Distributed Databases," 1977 Berkeley Workshop on Distributed Data Management and Computer Networks, Lawrence Berkeley Laboratory, May 1977.

[CHEN76] Chen, P.P., "The entity-relational model - towards a unified view of data." Transactions on Database Systems 1,1 (Mar. 1976), pp. 9-36.

[WONG79] Wong, E. & Katz, R.H.; "Logical Design and Schema Conversion for Relational and DBTG Databases," International Conference on the Entity-Relationship Approach to Systems Analysis and Design, Los Angeles, 1980.

[WONG80] Wong, E.: "The Design of Representation Schemas," Technical Report, Honeywell Corporate Computer Science Center, Bloomington, MN, July 1980.