A COMPARISON OF THE USE OF LINKS AND SECONDARY INDICES IN A RELATIONAL DATA BASE SYSTEM

by

Michael Stonebraker

Memorandum No. ERL-M591

4 September 1975

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

A COMPARISON OF THE USE OF LINKS AND SECONDARY INDICES IN A RELATIONAL DATA BASE SYSTEM

by

MICHAEL STONEBRAKER
Electronics Research Laboratory
and
Lawrence Berkeley Laboratory
University of California
Berkeley, Ca.

ABSTRACT

The possibility of supporting relational data sublanguages on top of a data base system with an underlying network data structure has been widely suggested. In this paper we present a formal model of a mix of interactions in one non procedural relational language. Under a collection of assumptions concerning the data base and the performance criteria, we compare the following performance oriented data sructures:

1) secondary indices

2) a network structure similar to a pointer array implementation of DBTG sets

3) structures 1) and 2) together

It is shown that option 2) is never preferred to both 1) and 3) over the range of model parameters. Hence, the sole use of sets or "links" as a performance oriented access path is questionable.

I INTRODUCTION

It has been suggested [1,2] that relational data sublanguages can be supported on top of lower level systems with an underlying network data structure. In [1] the sole performance oriented access path which can associate tuples in two different relations is the DBTG set (assuming that tuples are stored as records). In [2] this notion is generalized to one termed a "link" which is a (perhaps many to many) relationship between tuples in two relations. Two tuples are in the link if and only if a predicate associated with the link has a value of true for the pair.

In this paper we explore the performance implications of the use of links and secondary indices. To this end, we construct in the next three sections a formal model of the data base and the interaction mix associated with it and specify a performance measure.

Then in Sections 5-8 we indicate auxiliary performance oriented access paths. Section 5 indicates one form of secondary indices and specifies processing assumptions for the interaction mix in this situation. Section 6 similarly treats links while Section 7 deals with processing when both links and secondary indices are present. The last access path considered is the use of links under an alternate set of updating assumptions. Section 9 explores the performance of the various structures over the range of model parameters and draws conclusions.

II ASSUMPTIONS CONCERNING RELATIONS

We will compare three storage structures containing auxiliary information about two relations, R1 and R2. For all cases we make the following assumptions about the relations:

A1) R1 has M tuples and occupies MP pages of secondary storage
R2 has N tuples and occupies NP pages of secondary storage

A2) R1 and R2 can be accessed by a key to address transformation; i.e. the access method used to store R1 and R2 is "keyed" [3]. For simplicity we assume that D1 and D2 are the respective keys and are simple domains.

A3) Record placement for tuples on pages is identical in all cases to be analyzed.

III ASSUMPTIONS CONCERNING THE MIX OF TRANSACTIONS

A4) There are three forms of interactions

a) RETRIEVE F1(X), F2(Y), F3(X,Y)
WHERE Q1(X) AND Q2(Y) AND Q(X,Y)

Here, X and Y are tuple variables [3,4] for relations R1 and R2 respectively Moreover, F1, F2 and F3 are legal functions and Q1, Q2 and Q legal qualifications in the query language considered.

Lastly, we assume that Q(X,Y) involves a collection of domains DR1 and DR2 in R1 and R2 respectively.

b) REPLACE u in DR1 by F1(X) WHERE Q1(X)

c) REPLACE u in DR1 by F1(X)

WHERE Q1(X) AND Q2(Y) AND Q(X,Y)

We note that QUEL [3], Data Language/ALPHA[4], and SEQUEL [5] all allow expressions of these forms.

We assume that Q1(X), Q2(Y), F1(X), F2(Y) and F3(X,Y) are terms involving only the appropriate tuple variables and may vary from interaction to interaction. The linking term, Q(X,Y), however, is not a variable.

We assume that any other interactions that may be present in the mix of transactions require the same amount of time in all four cases and will not be considered further.

We assume that interaction a) occurs with probability P1, interaction b) with probability P2 and interaction c) with probability 1-P1-P2.

We furthermore assume that the expected number of tuples in R1 which satisfy the qualification Q1(X) is L1 and that this number is much smaller than MP.

We now indicate an example of each form of transaction in the data sublanguage, QUEL.

Suppose EMP(EMP#, NAME, MARITAL-STATUS, AGE, SALARY, DEPT#) and DEPT(DEPT#, SALES, FLOOR#) are the two relations of interest.

a) Find all employees and their floor numbers who are under 30 and are in a department with a sales volume over 100k.

```
RANGE OF E IS EMP
RANGE OF D IS DEPT
RETRIEVE (E.NAME, D.FLOOR#) WHERE
   E.AGE < 30  AND  D.SALES > 100k  AND  E.DEPT# = D.DEPT#
```

b) Put Smith in department 20.

```
RANGE OF E IS EMP
REPLACE E (DEPT# = 20) WHERE E.NAME = "Smith"
```

c) Move to department 15 all married employees in a department with sales volume over 3m.

```
RANGE OF E IS EMP
RANGE OF D IS DEPT
REPLACE E(DEPT# = 15) WHERE
 E.MARITAL-STATUS = "married" AND D.SALES >3M  AND E.DEPT# =D.DEPT#
```

A5) Q(X,Y) is in conjunctive normal form and has one of the following characteristics:

i) Q(X,Y) is a simple clause in which only the keys of R1 and R2 appear. This would be the case if both EMP and DEPT were keyed on DEPT# in the above example.

ii) Q(X,Y) is a simple clause in which only the key domain of R1 and one non key domain of R2 appear.

iii) Q(X,Y) is a simple clause in which only the key domain of R2 and one non key domain of R1 appear.

iv) Q(X,Y) is a simple clause in which only one non key domain from R1 and one from R2 appear.

v) Q(X,Y) does not satisfy i)-iv).

In the absence of specific information to the contrary we assume that Q(X,Y) is equally likely to be each of the above five cases.

IV ASSUMPTIONS CONCERNING THE PROCESSING OF TRANSACTIONS

A6) In all cases we ignore the cost associated with reading and/or changing qualifying tuples in R1 and R2 since it will be the same for each case.

A7) In all cases we assume that Q1(X) can be evaluated and either a list of qualifying tuples or a list of appropriate domain values can be obtained at the same cost in any of the four cases. Since this step will appear in the processing strategy for each situation, its cost will be ignored. Hence, whatever auxiliary structures are present to expedite this identification are the same. (It is tacitly assumed that these auxiliary structures do not include any of the specific structures dealt with in the four cases to follow.)

A8) Since each structure will be "keyed" in all of the four situations, we will consider as a performance measure the expected number of keyed accesses required to satisfy an interaction (noting clearly assumptions A6 and A7).

V SECONDARY INDICES

The first auxiliary structure we consider (labeled S1) is that secondary indices exist for domains J1 in R1 and J2 in R2. Here, J1 and J2 are the domains mentioned in i)-iv) of assumption A5. The secondary indices are assumed formed as relations and the one for J1 can be obtained as follows:

```
RANGE OF X IS R1
RETRIEVE INTO INDEX(X.J1, X.TID)
```

Here, X is a tuple variable as above and TID is a unique identifier for a tuple. We assume that INDEX is "keyed" on J1 and that

the index for J2 is analogous.

The processing strategy assumed in situation S1 for interaction a) is the following:

i) Evaluate Q1(X) to obtain potentially qualifying tuples in R1. By assumption A7 this cost will be neglected.

ii) For each potentially qualifying tuple satisfying Q1(X) use values from that tuple in interaction a) to obtain a query over relation R2. Evaluate this query using secondary indices if possible and append these partial results to form the desired result. This process is sometimes called "tuple substitution"[3].

In cases i)-iv) one access to the secondary index for R2 is required per X.TID to find tuple id's of potentially qualifying tuples in R2. Using only information from the secondary index and data domains from the given tuple in R1, Q(X,Y) can be evaluated and the pairs of actually qualifying tuples isolated. The cost of accessing these tuples is ignored by assumption A7. Consequently, L1 (the expected number of tuples satisfying Q1(X)) accesses are required to perform this step. (For simplicity we assume that the secondary index is present and used in case 1 even though R2 is keyed on J2 and that it is present in case v. A minor modification to the analysis which follows could drop both assumptions. The conclusions drawn in Section 9 are insensitive to these two assumptions, however.)

In case v) the secondary index for R2 is assumed not useful. Hence, a sequential scan of R2 is required for each X.TID found as above to isolate pairs of qualifying tuples. (See Section 9 for a relaxation of this assumption.)

In interaction b) one must first isolate and change tuples satisfying Q1(X). By assumptions A6 and A7 this cost is neglected. The secondary index for J1 must then be correctly updated.

For each X.TID which satisfies Q1(X) this requires one access to delete a tuple and one access to insert the revised tuple.

In interaction c) one must first execute interaction a) to find qualifying tuples, then one must change tuple values appropriately and update the secondary index for J1. As a result, the number of accesses required for each case is the sum of those required in interactions a) and b).

VI LINKS

Case S2 will be the use of a "link" [2] as an auxiliary structure. The link is assumed to be the following relation.

```
RANGE OF X IS R1
```

```
RANGE OF Y IS R2
RETRIEVE INTO LINK(X.TID, Y.TID) WHERE Q(X,Y)
```

Such a relation stores pairs of tuple identifiers which satisfy the qualification Q(X,Y). This structure is assumed keyed on X.TID.

A LINK relation is very similar to a pointer array implementation for DBTG sets [6]. In fact, if front compression [7] were applied to LINK the two structures would be nearly identical.

We do not treat a chaining implementation of LINK (as in the DBTG prposal) for the following reasons:

a) The desired implementation is quite different if one domain in LINK is functionally dependent [8] on the other (i.e. the relationship expressed by Q(X,Y) is one to one or one-to-many) than if the relationship is many-to-many.

b) There are several possible chaining implementations of one-to-many relationships each with somewhat different characteristics [1]. Hence, the analysis to follow would be unduly complicated.

We assume that LINK is inessential [9] in that it can be discarded without loss of information. There are two reasons for this choice.

a) S4) (to follow) is only possible with inessential LINK's

b) It is impossible to keep an essential LINK updated correctly with the interaction mix we have assumed. A similar point is noted in [10], and the following example illustrates the difficulty.

Example 6.1

Suppose EMP(EMP#,NAME, MARITAL-STATUS,AGE,SALARY, DEPT#) and DEPT(DEPT#, SALES, FLOOR#) are stored using a LINK equating DEPT#'s which is essential. Hence, the stored relations would be:

```
EMP(EMP#, NAME, MARITAL-STATUS, AGE, SALARY, DEPT#)
LINK(EMP.TID, DEPT'.TID)
DEPT'(SALES, FLOOR#)
```

Here, DEPT' is formed from DEPT by projecting on the last two domains.

The following two interactions perform the function of renumbering all departments to the floor number they occupy and then

changing all floor numbers to 1. (This update makes sense if there is only one department per floor and the business is moving to new quarters with several floors only the first of which is occcupied by existing departments.)

```
RANGE OF E IS EMP
RANGE OF D IS DEPT
REPLACE (E.DEPT# BY D.FLOOR#) WHERE D.DEPT# = E.DEPT#
REPLACE (D.DEPT# BY D.FLOOR#, D.FLOOR# BY 1)
```

This sequence of statements is legal in QUEL and will execute correctly when applied to EMP and DEPT. Unfortunately, when applied to EMP, LINK and DEPT', tne first update will empty LINK. Consequently, the second update becomes impossible. Reversing the order of the two REPLACE statements does not alter this situation and ther appears to be no way to correctly deal with this problem.

The processing strategy assumed for case S2 and interaction a) is to evaluate Q1(X) to obtain a list of potentially qualifying tuples in R1 then to use LINK to find pairs which actually qualify. Here, we make an assumption favorable to the use of links which is to ignore the cost of accessing "false drops" (i.e. tuples in R2 which are in LINK for some qualifying X.TID but which do not satisfy Q2(Y)).

For each X.TID which satisfies Q1(X) this requires one access to LINK. (We assume L1 is small compared to the number of pages required to store LINK so that L1 is a good approximation to the required number of accesses.)

In interaction b) no use can be made of the LINK and the processing must proceed either by using the key to address transformation if possible or a sequential scan of R1. The LINK must be appropriately updated when tuples in R1 are altered.

The following discussion summarizes the accesses required.

interaction b) cases i-ii

For each X.TID satisfying Q1(X) the following accesses are required

1 access to delete all (X.TID,Y.TID) which no longer satisfy Q(X,Y)

1 access to find all Y.TID from R2 which satisfy Q(X,Y) with the new domain values for the tuple indicated by X.TID

1 access to add these new pairs of tuple id's to LINK

interaction b) cases iii-v

For each X.TID satisfying Q1(X) the following accesses are

required

1 access to delete all tuples from LINK containing X.TID

A sequential scan of R2 to isolate those Y.TID's which satisfy Q(X,Y) with the new domain values for the tuple indicated by X.TID

1 access to add these new tuples to LINK

interaction c)

For each case the number of accesses is that required by interaction a) plus that required by interaction b).

VII DATA STRUCTURES INDICATED IN S1 AND S2 EXIST TOGETHER

The following discussion summarizes the accesses required in the various situations.

interaction a)

same as interaction a) in S2

interaction b) cases i-iv

For each X.TID which satisfies Q1(X) the following accesses are required

1 access to delete an entry from the secondary index for R1

1 access to add an entry to this secondary index

1 access to delete entries from LINK

1 access to find entries which should be added to LINK from secondary index for R2

1 access to add new entries to LINK

interaction b) case v

This is the same as the above situation except the fourth access must be replaced by a sequential scan of R2.

interaction c)

For each case this is the sum of the accesses in interactions a) and b).

VIII LINKS WITH NO UPDATE

In S4 the same structures exist as in case S2. However, no attempt is made to update LINK when interaction b) or c) occurs. The LINK is rebuilt before processing interactions of type a) or c) if a type b) or c) interaction has occured since the last type a) or c) transaction.

The number of accesses needed is the following.

interaction a) case i

This is the same as case i in S2 unless LINK must be rebuilt. If so, a scan of R1 and R2 is required.

interaction a) cases ii-v

This is the same as S2 unless LINK must be rebuilt. If so, the cross product of R1 and R2 must be examined.

interaction b)

No extra overhead is required

interaction c)

The sum of the accesses for interactions a) and b) is required for each case.

IX COMPARISON OF THE FOUR CASES

Table 1 indicates the number of accesses required for each case.

	S1	S2	S3	S4
ai	L1	L1	L1	L1 or NP+MP
aii	L1	L1	L1	L1 or NP*MP
aiii	L1	L1	L1	L1 or NP*MP
aiv	L1	L1	L1	L1 or NP*MP
av	L1*NP	L1	L1	L1 or NP*MP
bi	2*L1	3*L1	5*L1	0
bii	2*L1	3*L1	5*L1	0
biii	2*L1	L1(NP+2)	5*L1	0
biv	2*L1	L1(NP+2)	5*L1	0
bv	2*L1	L1(NP+2)	5*L1	0
c ((In each case c requires the sum of the accesses of a and b))				

A Summary of the Required Accesses
Table 1

Note in all cases that a sequential scan of R2 for each X.TID (i.e. wherever NP appears in the table) can be avoided by batching a group of X.TID's for processing in a single scan. This would have the effect of dividing NP by the batch size in the above table. If this were done NP could be considered as the weighted relation size and the analysis to follow is still correct.

It should also be noted that the analysis to follow is relatively insensitive to the assumption that each of the five forms of Q(X,Y) is equally likely but quite sensitive to the interaction mix chosen. The particular mix was considered because it was thought to be generally favorable to links (the reader should note the obvious problem with links if interaction b) can range

over either R1 or R2 instead of just R1). However, the methodology can be applied to any given mix.

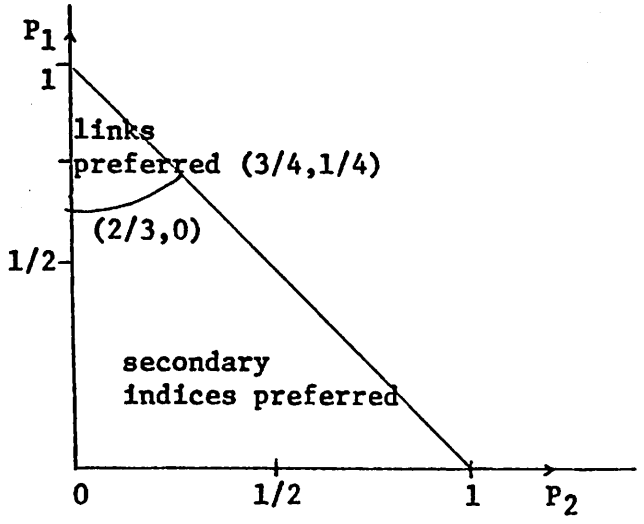
Let E(Si) be the expected number of accesses using structure Si. In this situation algebraic manipulation produces the following:

E(S2)-E(S1)=

$$(L1/5)[NP(2-3P1+P2) +3-2P1-P2]$$

This expression is guaranteed to be positive if $P1 < 2/3$ regardless of the value of P2. For $P1 > 3/4$ the term multiplying NP is guaranteed to be negative for any value of P2. In this situation, the expression is either negative or at most $(L1/5)(3/2)$.

Consequently, if the retrieval probability is less than 2/3 secondary indices are preferable to links. If the retrieval probability is greater than 3/4, links are usually preferred. In the middle range the choice depends on both P1 and P2. Figure 1 indicates this fact pictorially.



Tradeoff between Links and Secondary Indices
Figure 1

We now turn to comparing situation S3 with S1 and S2. The following two equations are readily shown:

E(S3)-E(S1)=

$$(L1/5)[NP(P2-P1) +15-14P1-P2]$$

E(S3)-E(S2)=

$$(L1/5)[NP(2P1-2) +12 -12P1]$$

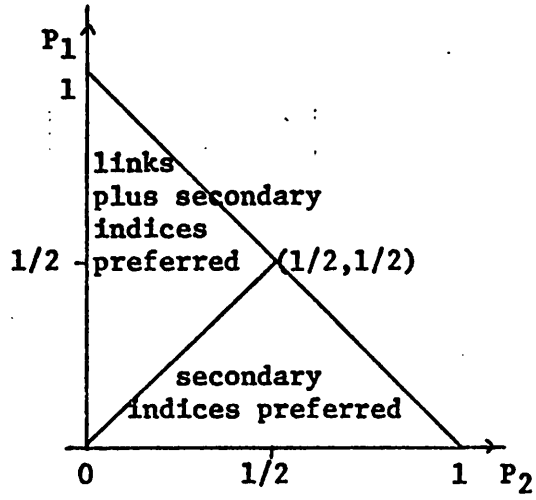
$$= (L1-5)(2NP-12)(P1-1)$$

The following conclusions can be easily drawn:

1) Secondary indices are preferred to secondary indices plus links if P2>P1.

2) Links are never preferred to secondary indices plus links. Hence, links alone are not a good idea. This is true as long as NP>6.

These conclusions are summarized in Figure 2



Tradeoff Between Links plus Secondary Indices and Secondary Indices
Figure 2

The following equation is easily derived for E(S4).

$$E(S4)= (1-P2)(P1*L1 +(1-P1)(NP+MP+4*MP*NP)/5)$$

Moreover:

$$E(S4) > (L1/5)[5P1*(1-P2) +(1-P1)(1-P2)(4NP*MP/L1)]$$

and:

$$E(S4)-E(S3) > (L1/5)[-29+24P1+NP(1-P1)[(4MP/L1)(1-P2)-1]]$$

$$E(S4)-E(S1) > (L1/5)[-9+5P1+4P2+NP(1-P2)[(1-P1)(4MP/L1)-1]]$$

It is also easily shown that:

$E(S4)>E(S1)$ if $P1< 1- L1/4MP$

or if P2 is not approximately 1

and

E(S4)>E(S3) if P2 < 1- L1/4MP

or if P1 is not approximately 1

The following conclusions are readily drawn:

1) S4 is the preferred structure only if P1 is essentially 1, i.e. in retrieve only situations. In this case there is no loss by not updating redundant information because it never changes.

2) S4 is also preferable if P2 is approximately 1, i.e. in update only situations. In this case it is not faced with keeping redundant information which speeds retrieval current.

3) As noted in Figures 1 and 2, S2 is NEVER preferred to both S1 and S3. As a result, the sole use of links is questionable as a performance oriented access path.

4) As noted in Figure 2, the use of both links and secondary indices is a winner only in "retrieve mostly" situations. Since supporting both structures increases system complexity, a reasonable choice is the sole support of secondary indices. This choice was taken in at least two relational implementations [11,12].

5) Lastly note that no conclusions concerning chained implementations of LINKS (i.e. one-to-many DBTG sets) can be drawn. It is not clear how to analyze the performance of sets using the above methodology.

ACKNOWLEDGEMENT

Research Sponsored in part by National Science Foundation Grant DRC75-03839

REFERENCES

1) Committee on Data Systems Languages, "CODASYL Data Base Task Group Report", ACM, New York, 1971.

2) Tsichritzis, D., "A Network Framework for Relational Implementation", University of Toronto, Computer Systems Research Group Report CSRG-51, Feb. 1975.

3) Held, G., Stonebraker, M. and Wong, E., "INGRES-A Relational Data Base Management System", Proc. 1975 National Computer Conference, AFIPS Press, 1975.

4) Codd, E.F., "A Data Base Sublanguage Founded on the Relational Calculus", Proc. 1971 ACM-SIGFIDET Workshop on Data Description, Access and Control, San Diego, Ca., Nov. 1971.

5) Astrahan, M. et al, "SYSTEM R: A Relational Approach to Data Base Management", ACM Transactions on Data Base Systems, Vol 2, No. 2.

6) Taylor, R., "When are Pointer arrays better than Chains?" Proc. 1974 ACM National Conference, San Diego, Ca., Nov. 1974.

7) Gottlieb, D., et.al. "A Classification of Compression Methods and their Usefulness in a Large Data Processing Center" Proc. 1975 National Computer Conference, AFIPS Press, May, 1975.

8) Codd, E., "A Tutorial on Normal Forms", Proc. 1971 ACM-SIGFIDET Workshop on Data Description, ACcess and Control, San Diego, Ca. Nov. 1971.

9) Codd, E.F. and Date, C.J., "Interactive Support for Non-Programmers, The Relational and Network Approaches", Proc. 1974 ACM-SIGFIDET Workshop on Data Description, Access and Control, Ann Arbor, Mich., May 1974.

10) Kay, M., "Implementing a Relational Schema on a Network Schema", Proc. IFIP TC2 Conference on Data Definition Languages, Namur, Belgium, January, 1975.

11) Held, G. and Stonebraker M., "Storage Structures and Access Methods for the Relational Data Base System, INGRES", Proc. ACM-PACIFIC-75, San Francisco, Ca., April, 1975.

12) Astrahan, M. and Chamberlin, D., "Implementation of a Structured English Query Language", Proc. 1975 ACM-SIGMOD Workshop on Management of Data, San Jose, Ca., June 1975.