Extensions of the Reduction Process in the Wong-Youssefi Strategy for Query Processing*

Gary Haggard**

Dale Skeen

August 1982

TR 82-507

Department of Computer Science
Cornell University
Ithaca, New York 14853

**On Leave From:
Department of Computer Science
University of Maine at Orono
Orono, Maine 04469

*This work was supported under NSF Grant SPI-8165050

EXTENSIONS OF THE REDUCTION PROCESS IN THE WONG-YOUSSEFI STRATEGY FOR QUERY PROCESSING

Gary Haggard*
Dale Skeen

TR 82-507
August 1982

Department of Computer Science
Upson Hall
Cornell University
Ithaca, New York 14853

*Visiting Professor; permanent address: Department of Computer Science, University of Maine at Orono, Orono, Maine 04469.

This work was supported under NSF Grant SPI-8165050

Extensions of the Reduction

Process in the Wong-Youssefi Strategy for Query Processing

I. Introduction

The efficient operation of a non-procedural data sublanguage requires an optimization of the query processing. Wong and Youssefi describe the implementation strategy of the query processing algorithm for the relational data base system INGRES [3]. The Reduction step in their algorithm decomposes the query into a set of irreducible components. The query is then executed by processing the irreducible components sequentially. Because of the method used to execute an irreducible component, it may be necessary to repeat this Reduction process several times while a query is being executed. This paper gives an extension of the Wong-Youssefi model for a query. The extended model makes visible more of the information gained about the query during the Reduction process. This additional information can be used for ordering the irreducible components for execution. The order chosen is important because some relations occur in more than on irreducible component and so the execution of one component can effect the efficiency of the execution of the remainder of the query. The interaction between irreducible components is clearly represented in this extended model but is not represented in the Wong-Youssefi model. If parallel processing is considered for a fully distributed data base, the extended model makes clear which portions of the query can be executed in parallel.

The Reduction process of Wong-Youssefi use $O(n^2)$ algorithms. The Reduction process based on the extended model presented here uses O(n) algorithms.

Finally the extended model is used to demonstrate a difficulty in query processing that is not yet satisfactorily resolved by any of the algorithms that have been proposed for the optimization of query processing.

II. Queries in QUEL

The data language for INGRES is called QUEL. A complete description of QUEL can be found in [2]. The only command of QUEL needed for purposes of this paper is RETRIEVE that has the form:

RANGE OF {Variable} IS {Relation}

RETRIEVE INTO Result-Name (Target-list) WHERE Qualification

The list {Variable} is a set of variables representing the physical relations listed in {Relation}. As the elements of Variable range over the tuples in the relations they represent the tuples that satisfy the conditions in Qualification are used to form the relation Result-Name which has as attributes those listed in Target-list. We assume that the Qualification term of a query is given as a conjunction of terms of the form P.A rela S.A or P.A = c where P and S are relations; A is a variable; c is a constant and rela is a permissible comparison operator. A term of the form P.A = c is called a 1-variable query. We also assume the Qualification term does not contain any aggregate operators.

III. The Query Graph and Irreducible Components

The first step in modeling the query is to define the following graph.

DEFINITION 1. Let Q be the Qualification term of a query **Q**. The query graph G(**Q**) is the undirected graph consists of the following vertices and edges:

(i) v ϵ V(G) if and only if v is a relation that occurs in Q or v is a constant occurring in a 1-variable term of Q.

(ii) (v,w) ϵ E(G) if and only if there is a term in Q of the form v rela w where rela is a permissible comparison operator.

It is sometimes useful to denote an edge of G(**Q**) as a triple (P,S,A) when the term of Q it represents is of the form P.A rela S.A or (P,c,A) when the term if of the form P.A rela c. We now give an example of G(**Q**) for the query used in [1].

RELATIONS IN THE DATA BASE

Supplier (S#,Sname,City)

Parts (P#,Pname,Size)

Supply (S#,P#,Quantity)

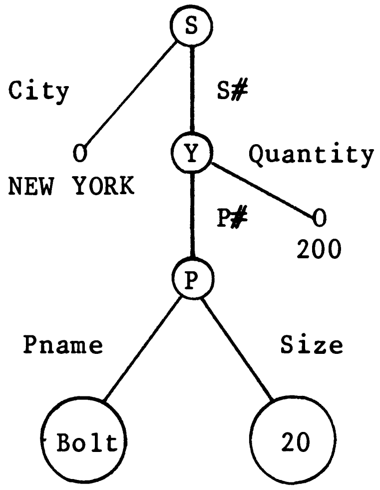
RANGE OF (S,P,Y) IS (Supplier,Parts,Supply)

```
RETRIEVE (S.Sname) WHERE (S.City = 'NEW YORK')
                   AND  (P.Pname = 'Bolt')
                   AND  (P.Size = 20)
                   AND  (Y.S# = S.S#)
                   AND  (Y.P# = P.P#)
                   AND  (Y.Quantity ≥ 200)
```

V(G) = {'NEW YORK','Bolt', 20, 200, P, S, Y}

E(G) = {(S,'NEW YORK',City), (P,'Bolt',Pname), (Y,S,S#),(Y,P,P#),

(Y,200,Quantity), (P,20,Size)}



One of the relations holding a variable from the Target-list is usually indicated as the root of this graph. A depth first search starting at the root will identify both the connected components and the irreducible components of the query in O(n) time where n = |E(G)| provided the graph is represented by an adjacency list structure. In this example each edge is an irreducible component. There is no reason at this point to treat the irreducible components that are 1-variable terms differently.

The irreducible components constructed are the same as the irreducible components of [3].

IV. The Query Processing Graph

Joining variables are defined in [3] to be those relations whose deletion causes the query graph to become disconnected. The irreducible components of the query graph are the maximal connected subgraphs that contain no joining variables. We now define a graph that gives a representation for a query that can be used in determining the ordering for the processing of the irreducible components. This graph gives joining variables a visible role so that they can be considered in determining the order of execution for the components containing them.

DEFINITION 2. Let G be the query graph of a query **Q**. The query processing graph GPG is the undirected graph consists of the following vertices and edges:

(i) $v \in V(QPG)$ if and only if v is a irreducible component of G or v is a joining variable of G.

(ii) $(v,w) \in E(QPG)$ if and only if v is a joining variable of G and w is an irreducible component of G containing the joining variable v.

A well known theorem in graph theory [1] gives a characterization of the query processing graph. In graph theory terminology a joining variable is a cut point and an irreducible component is a block.

Theorem. A graph G is the block-cutpoint graph of some graph H if and only if it is a tree in which the distance between any two endpoints is even. In a query processing graph we use the convention that the vertices representing irreducible components are named with lower case letters and the vertices representing joining variables are named with capital letters.

We now give an example of the QPG for the query graph shown in II.

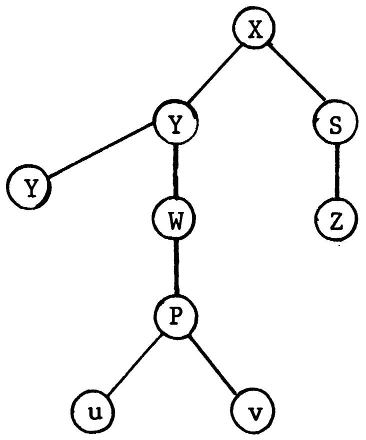
QPG

V(QPG) = {P,Y,S, u=(P,'Bolt',Pname), v=(P,20,size),

w = (P,Y,P#), x=(Y,S,S#), y=(Y,200,Quantity),

z = (S,'NEW YORK',City)}

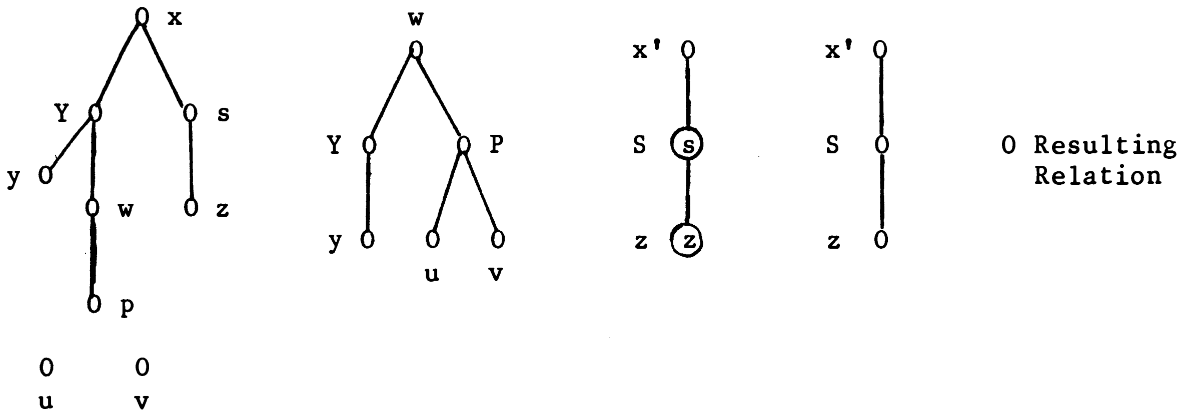
E(QPG) = {(P,u), (P,v), (Y,y), (Y,x), (Y,w), (S,z), (P,w), (Y,x), (S,x)}



The leaves of the query processing graph will be irreducible components of the query. The joining variables will always have degree at least 2 in the query processing graph. If is worth remarking again that the joining variables S,Y and P play a visible role in this representation of the query.

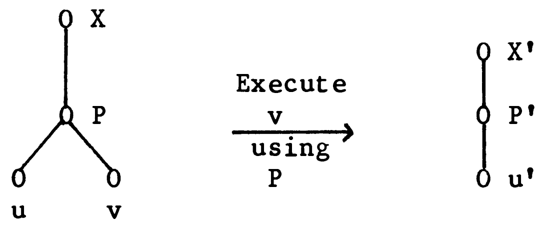
IV. Processing Strategies Based on the Query Processing Graph

The rules for ordering the irreducible components for sequential execution given in Wong and Youssefi treat 1-variable queries as a special case. A 1-variable query is always executed along with the irreducible component that contains it. With the additional design decision that the component that contains the variables in the Target-list should be executed last, the execution steps in the example so proposed by Wong and Youssefi are the ones expected.

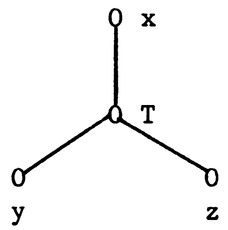


QPG_o Step 1 QPG_1 Step 2 QPG_2

Since each 1-variable query is an irreducible component, it may be useful to execute a 1-variable query after executing the component containing the joining variable in the 1-variable query. The query processing graph is easily modified to reflect this in a way similar to what is pictured below:

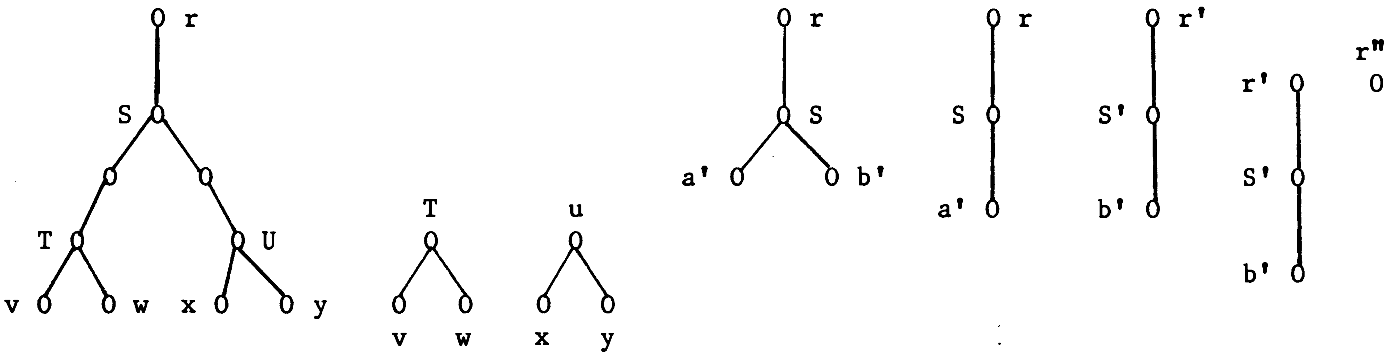


It is not hard to construct a query processing graph in which the execution steps are not so well defined and the order of execution can become much more critical. For example, consider the following query processing graph:



Suppose that none of the query terms x,y or z are 1-variable queries. The Wong-Youssefi algorithm picks one of the two queries y and z for execution first. The choice is determined by finding the first row in the matrix that represents an irreducible components that has more than a single non-zero entry. It seems it would be more effective to do some sort of a look ahead at a vertex of the query processing graph representing a joining variable before ordering the components that contain it. One option to consider in such a look ahead at a joining variable is whether or not induces exist that could make it more efficient to execute one branch before another. The current algorithm used in INGRES acts as if no indices exist and could actually execute a query term without an index before a query term involving the same joining variable that had an index. In such a case the execution may be much less efficient than if the branches were executed in the other order. Since there are so many efficient tree transversal algorithms, some analysis at cut vertices could be incorporated into the process of determining the order of execution for the irreducible components.

Besides possible improvements in the Wong-Youssefi algorithm for ordering for execution the irreducible components of the query graph, the query processing graph can be used to implement query processing in a fully distributed data base environment. An example illustrates this:



$QPPG_0$ Step 1 in parallel GPG_1 Step 2 GPG_2 Step 3 GPG_3

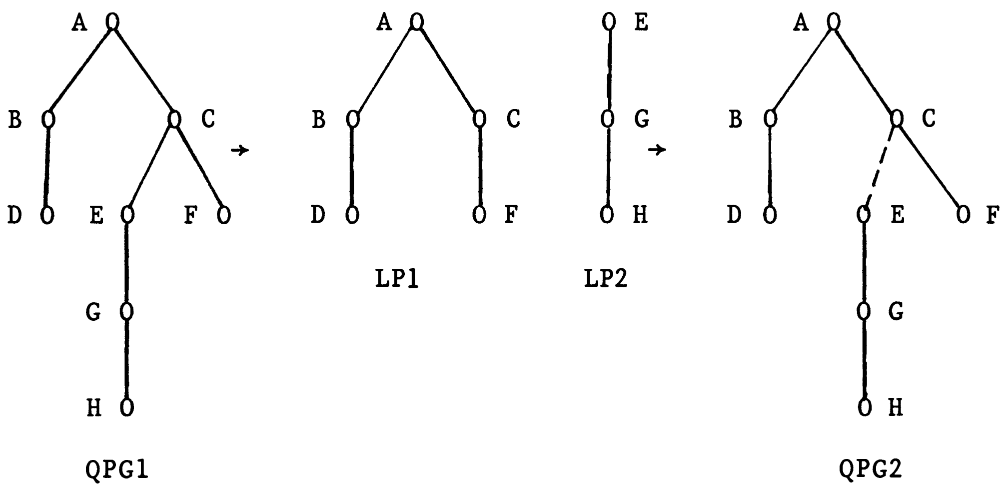
The subtree with root T can clearly be executed independent of the execution of the subtree with root U.

V. The Query Processing Graph and Query Optimization

Possible query processing optimization techniques include: (i) linear programming; (ii) exhaustive search; (iii) greedy algorithm; and (iv) tuple substitution. Careful examination of these strategies in terms of the representation of the query by the query processing graph makes clear a difficulty none of these algorithms satisfactorily resolve.

Rather than consider each of these strategies separately, we will only consider the linear programming approach because the key difficulty in this approach is really the same difficulty encountered in the others. We do not focus on the difficulty of finding an optimal way to process an irreducible component of a query but rather on how the interrelation among irreducible components effects execution. The problem of how to design an efficient algorithm for executing irreducible components remains a separate thought not unrelated problem.

An optimization algorithm using linear programming assumes that by examining each possible partition of the query into two connected pieces, it is possible to find a partition such that the optimal strategy for executing the query is the result of executing the optimal strategies for the two parts of this partition and then joining the results of these two subqueries in a straightforward manner. The partitioning process is continued recursively on each of the two pieces of the partition being considered until a partition consists of a single irreducible component. The assumption is that the optimal way of executing an irreducible component is not effected by the way it enters into the optimal execution of the query itself. Thus it is possible to determine an optimal way of executing each piece of the original partition. By choosing the minimum over all such partitions an optimal strategy for evaluation the query is determined. The figure below will help explain the unresolved difficulty with this approach.



The original query processing graph is QPG1. Assume that LP1 and LP2 form a partition of the query that leads to an optimal solution. Notice that LP2 has no edge joining E to C. The relation C is still in LP2 but it is not a joining variable when LP2 is considered as a subquery. The linear programming optimization does not take into account that C is a joining variable in QPG1. Similarly the optimization of LP1 does not take into consideration that C is a joining variable for the irreducible component E as well as the irreducible components A and F. It is clear though that to put the solutions of LP1 and LP2 together will require the solution LP2 to preserve enough of C so that the edge (C,E) can be replaced. This already says that it is not enough just to optimize LP1 and LP2 separately to optimize the execution of the query.

The interrelationship between the role of a joining variable in a particular irreducible component and its role of joining two or more components together is not very well understood. Much work remains to understand the role of joining variables in determining an efficient ordering for execution of the irreducible components of a query. The extended model of a query described gives a new tool for experimenting with the implementation of query processing strategies that capture more information that effect the execution of a query. A better understanding of the role of joining variables may also shed additional light on how to design efficient algorithms for executing the irreducible components of a query.

Bibliography

[1] Harary, F., Graph Theory, Addison Wesley, 1969.

[2] Held, G.D., Stonebraker, M. and Wong E. INGRES--a relational data base management system. Proc. AFIPS 1975 NCC, Vol. 44, AFIPS Press, Montvale, New Jersey, pp. 409-416.

[3] Wong, E. and Youssefi, K., "Decomposition - A strategy for Query Processing," TODS, 1, 1976, 223-241.