Application of a Hardware Synthesis Technique for Database Query Optimization*

Vivek Komaragiri, Mitchell A. Thornton
Mississippi State University
Mississippi State, Mississippi

Rolf Drechsler
Siemens AG
Munich, Germany

ABSTRACT

The size of a typical modern Database is on the order of hundreds of *Gigabytes* (GB) and the need for developing systems for processing such huge amounts of data has emerged. A highly efficient information retrieval method is required to make the process quicker and easier. This paper focuses on this aspect and presents a method to optimize an SQL query for efficient information retrieval. It is shown that an SQL query can be optimized using AND/OR graphs. Experimental results, which show the efficiency of this method, are provided.

1.0 Introduction

SQL is a very widely used commercial language for query processing. As a query can be formulated in multiple ways to obtain same set of information, the way it is formulated and presented to the *DataBase* (DB) has a tremendous impact on the efficiency of the corresponding data retrieval.

Query optimization is the process of optimizing a query before it is presented to the database according to some criteria. Here, we focus on data access speed as the optimizing criterion. A query can be represented efficiently using a data structure called an AND/OR graph. The main advantage of AND/OR graph representation is that sub-graph isomorphism frequently occurs as described in detail in [2]. This crucial property coupled with the fact that several powerful CAD techniques developed over the past few years can be used to optimize an SQL query if it is represented using AND/OR graph is the basis for representing the query in this form.

Section 2 gives the background information about digital logic optimization and AND/OR graphs. Section 3 describes the methodology used in this approach. Section 4 gives the experimental results and the conclusion is given in Section 5.

2.0 Background

In this section the concepts of digital logic optimization and relational databases are very briefly introduced. Digital logic circuits are circuits that operate using only two voltages assigned with values '0' and '1'. These digital logic circuits can be viewed as implementations of Boolean functions that operate on a set of binary valued variables. Given a specific functionality, a digital function can be represented in many ways, functions containing a minimum number of literals are hard to find and are considered to be absolute optimal functions. A 'literal' is the occurrence of a variable in an expression. Since finding an absolute optimal function is a provably hard problem many heuristics have been used to achieve near optimal solutions.

A DB is a term used for the storage of data in a particular organization in the computer. A Database Management System (DBMS) is used to access and manipulate data in a DB. Among all the models proposed so far, the "relational" model is the most widely used model for DB's. Information is typically extracted from a database through the use of a query. After a query is presented to a DB, the DBMS initiates a search over its entire set of records and retrieves the information. This may take an increasing amount of time as the size of DB increases and even more time if the query is formulated in an inefficient way.

2.1 AND/OR Graphs

An efficient underlying data structure was a necessity to solve many problems in the area of computer aided design and so several methods based on Decision Diagrams (DD) came into existence [4]. DD's are canonical representations of Boolean function and can be easily manipulated and many operations can be performed on a Boolean function if it is represented using a Decision diagram.

DD's are based on pure OR search and in this kind of search each outgoing edge from a node represents a possible move that can be made at the current state of search process and a solution is formed by traversing the graph using certain strategy. AND/OR graphs were proposed recently as an efficient data structure for representing a Boolean function. In these type of graphs the kind of search technique employed is known as AND/OR search and in this kind of search process, a certain move at a given search

*This work was supported in part by a Research Initiation Award from Mississippi State University, the National Science Foundation under grant INT-0096008 and the DaaD under grant 315/PPP/gü-ab.

state leads to several new problems that all have to be solved. One of the key features of AND/OR search is that it is sufficient to only partially traverse the graph in order to extract valuable information. The main differences between AND/OR graphs and binary decision diagrams are described in [1]. Techniques for building AND/OR graphs from an arbitrary combinational circuit are described in detail in [7].

3.0 Methodology

The experimental setup for achieving SQL query optimization is as shown in Figure 3.0.

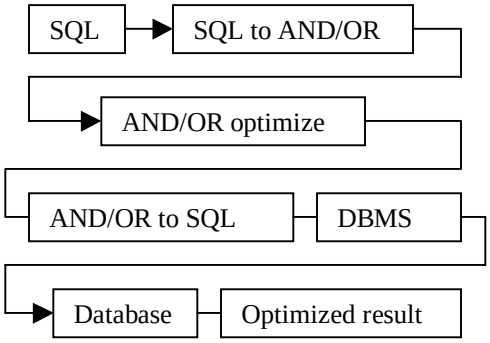

Figure 3.0: Experimental Setup

An AND/OR graph is generated corresponding to the query queried by the user. An SQL parser is interfaced with an AND/OR package to achieve the task of generating AND/OR graphs for the queries. AND/OR package identifies all isomorphic parts and reduces the AND/OR graph (in size) and this reduced graph represents the same query but in most cases it will be more compact than the original query. The process of reduction in AND/OR graph is described in detail in [2]. Two types of isomorphism's occur in an AND/OR graph, they are AND and OR type isomorphism's designated by isomorphic sub-graphs rooted in AND and OR subgraphs respectively. The reduction process involves the use of an open hashing technique which is used to share all the isomorphic sub-graphs in an AND/OR graph. The AND/OR nodes that are addressed by the same list hash to the same hash key. The details about the hashing technique, the hash function used and the algorithms used for the reduction of AND/OR graphs are given in [2]. The query can be reconstructed back from reduced AND/OR graph and will be an optimized query of reduced length.

This process can be explained with the help of an example. Consider the following SQL query:

Select pname from project where (Pteam= CAD and Pcity=Starkville and P#=P1006) or (Pcity=Starkville and EmpName=John and Pteam = CAD) or (Pteam=CAD and Pleader= chris and Pcity = Starkville);

This query is represented in the form of a digital circuit as shown in Figure 3.1. This circuit represents the case where the query will not succeed (***Pname=0***).

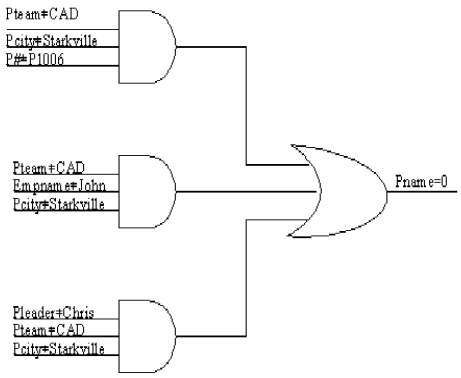

Figure 3.1: Circuit corresponding to example query

At this stage, optimizing the query can be considered as optimizing the corresponding digital logic circuit. In this way, the query optimization problem can now be thought of as the problem of digital logic optimization. As finding an absolute minimal expression for a digital logic function is hard, it is also not always possible to get an absolute optimal query.

The AND/OR graph generated from the above digital circuit is shown in Figure 3.2.

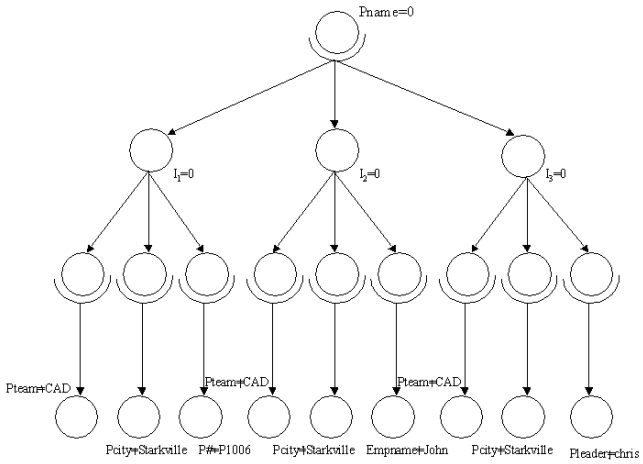

Figure 3.2: AND/OR graph of example query

The above AND/OR graph contains some redundant information and this redundancy is removed in the AND/OR package resulting in the graph shown in Figure 3.3 and this graph contains a fewer number of nodes and also the crucial information that "***Pteam=CAD***" and "***Pcity = Starkville***" are necessary conditions for the success of the query. If either of these conditions fail then the query fails irrespective of other conditions. If we build a digital logic circuit from the reduced AND/OR graph then the circuit obtained is shown in Figure 3.4.

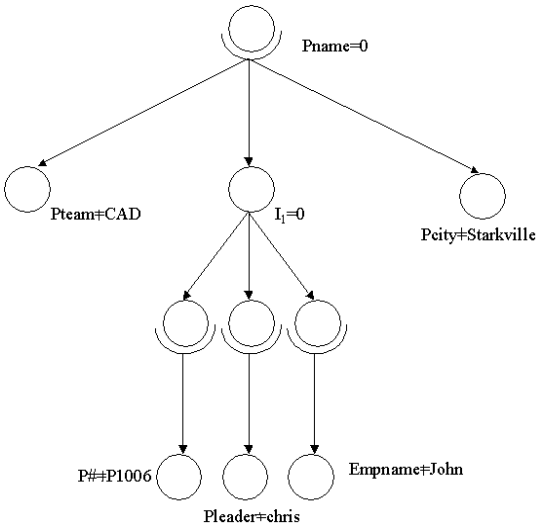

Figure 3.3: Reduced AND/OR Graph

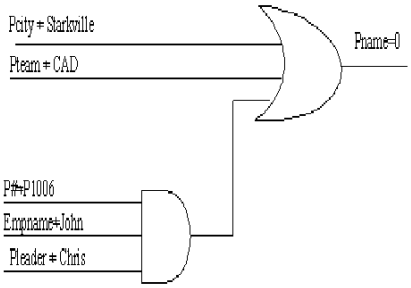

Figure 3.4: Ciruit Corresponding to Reduced AND/OR Graph

The circuit shown in Figure 3.4 represents the case of a query failure and since we are interested in probable success of the query and not failure, the circuit in figure 3.4 is converted to a circuit that represents the success of the query (Pname=1). This is obtained by placing inverters at the outputs of OR gate and at all the inputs since we want equalities of field values, not inequalities.

The circuit which is obtained for Pname =1 is shown in Figure 3.5 which reduces to Figure 3.6 and the query can be constructed back from the circuit in Figure 3.6 and doing so yields the following:

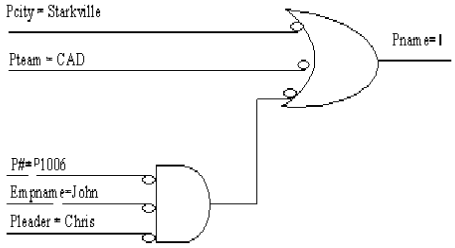

Figure 3.5: Negated Circuit Used to Obtain Successful Query

Select Pname from project where Pteam = CAD and Pcity=Starkville and (P#=P1006 or Empname=John or Pleader=Bryan);

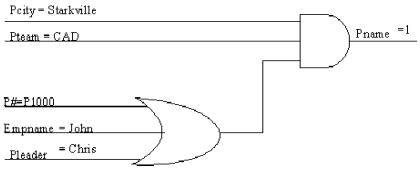

Figure 3.6: Circuit Corresponding to Optimized Query

This is a trivial example but it clearly indicates how AND/OR graphs can be used to reduce the length of the query. Most of the redundant information is discarded from the query if represented by an AND/OR graph.

4.0 Experimental Results

The execution time taken for optimized queries (using AND/OR graphs) using the ORACLE DBMS is compared with that of original query. The total execution time for the optimized query is the sum of execution time taken in the AND/OR package and the execution time taken by the DBMS to process the resulting query. The details of implementation of AND/OR package used in this method are described in detail in [2].

Table 4.0: Experimental Results

Query	Orig.	AND/OR	Opt. query	total
1	0.29	0.04	0.09	0.13
2	0.10	0.02	0.05	0.07
3	0.08	0.03	0.04	0.07
4	0.12	0.03	0.03	0.06
5	0.13	0.02	0.09	0.11
6	0.09	0.03	0.04	0.07
7	0.09	0.03	0.03	0.06
8	0.36	0.09	0.13	0.22

Table 4.0 shows the experimental results. "Orig" is the execution time taken by the original query, "AND/OR" is the time taken for building an AND/OR graph for the query in the AND/OR package, "opt. query" is the time taken by the DBMS to process the optimized query. All results are given in seconds of CPU runtime. These results indicate that this technique is very effective for some queries. This technique is very effective for queries having similar sub-queries. In almost all the queries tested the execution time taken by the optimized query is either equal to or less than original query but not more.

5.0 Conclusions and Future Work

A method is introduced for optimizing an SQL query with the use of AND/OR graphs as an intermediate data structure. The experiments indicate that in most cases the execution time of the query is reduced at least by 30%.

Future directions in this research include implementation to pass a set of queries simultaneously to the AND/OR generation software and then to identify sub-graph isomorphism among the queries. This would result in an inter-query optimization where we have only examined intra-query optimization here. Another possible extension to this research would be to use a "Multivalued decision diagram" (MDD) instead of an AND/OR graph as a data structure for representing the query. AND/OR graphs can be used to represent a subset of queries only in Boolean domain whereas if an MDD is used to represent the query then all possible sets of queries can be optimized.

REFERENCES

[1] R. Drechsler, W. Kunz, A. Zuzek and D. Stoffel, *"Decision Diagrams and AND/OR graphs for Design Automation problem"*, Proceedings of the International Conference on Information, Communication and Signal Processing, 1997, pp 67-72.

[2] A. Zuzek, R. Drechsler and M. A. Thornton, *"Boolean Function Representation and Spectral Characterization using AND/OR graphs"***.** INTEGRATION, The VLSI journal. Vol. 29, September 2000, pp 101-106

[3] D. Stoffel, W. Kunz and S. Gerber, *"AND/OR reasoning graphs for determining prime implicants in Multilevel combinational circuit,"* in Proceedings of the ASP Design Automation Conference, 1997, pp 25-32.

[4] B. Becker and R. Drechsler, "*Decision Diagrams in Synthesis, algorithms , applications and extensions"*, **in** VLSI Design Conference, 1997, pp 46-50.

[5] R. E. Bryant, "*Binary Decision Diagrams and beyond*: Enabling techniques for formal verification", in Int'l Conf on CAD, 1995, pp 236-243.

[6] R. E. Bryant, *"Graph Based Algorithms for Boolean function manipulation"*, IEEE Transactions on computers, 1986, pp 677-691.

[7] D. Stoffel, W. Kunz, S. Gerber," *AND/OR Graphs"*, Technical Report, MPI-I-95-602, 1995.

[8] H. Korth and A. Silberschatz, **Database System Concepts**, McGraw-Hill, Inc., New York, NY, 2nd edition, 1991.

[9] W. Kunz and D. K. Pradhan, "*Recursive Learning, A New Implication Technique for efficient solutions to CAD problems: Test, Verification and Optimization*", IEEE Transactions on CAD, 1994, pp 1143-1158.

[10] K.S Brace, R.L Ruddell, and R. E. Bryant, *"Efficient implementation of a BDD package,"* in Proceedings of design automation Conf., 1990, pp 40-45.