# A Practical Issue Concerning Very Large Data Bases: The Need for Query Governors

Gerald Cohen, President
Information Builders, Inc.
1250 Broadway
New York, NY 10001 USA

**SUMMARY:** The popularity of client/server technology coupled with the concept of the 'Information Warehouse' as a central place where end users come to access large organization data bases has some problems that have not received much study. One of these problems is the danger of inexperienced users formulating 'run away' queries which consume excessive computer resources while impacting all other users. The construction of query governors to prevent this is a major practical problem, as well as theoretical problem.

## The Problem

The concept of the information warehouse as a place to keep the information assets of an organization implies by the word 'warehouse' that it is also the place for distribution of these assets. But access to these assets cannot be distributed without some controls.

One of the least analyzed areas of very large data base problems is the problem of unlimited user access. With the prevalence of work stations and connectivity software and networks the desk top worker is now the 'empowered' worker. Ad hoc query is the watchword for these users. We at Information Builders sell precisely the software, sometimes called 'middleware', to connect these end users with the largest databases. Our EDA/SQL (Enterprise Data Access/SQL) product allows all of the work station software products which use SQL to access all of the relational and non-relational data bases on large IBM, VAX and Unix computers. About 200 software products now have the linkage to EDA/SQL. For example, you can sit behind a PC running Microsoft Windows Excel spreadsheet and request data from VSAM files on an IBM mainframe computer.

The danger of submitting a 'ill-structured' query which runs away with the host computer is real. It soon became apparent to us that while the desire to distribute data was genuine most organizations had never considered the effects of uncontrolled end user access. If we were to continue to sell our product we had to help them govern the query process so as to prevent 'run away' queries which ate up machine resources and froze out other users. Hence, we developed our SmartMode query governor.

## Expert System Approach

In the mid eighties we entered the Expert Systems business with a product named Level5. It achieved modest success and has suffered like all artificial intelligence products by lack of market interest. However, complex problems require the deep functionality software offered by expert systems, so we were positioned to tackle query governing. The engineering side of the problem requires that the SQL query be intercepted, parsed, and passed to a governor for acceptance or rejection. The governor is itself a database, but of rules about the resources used by the elements of queries.

There are three ways to collect these rules. The first is to construct them from the knowledge of experienced users of the particular data base. This is a traditional expert system methodology, and may be appropriate for specific data bases in some environments. It won't work in a mass market because the human resources are not trained or available. The second approach is to use induction technology. This requires that we use an 'engine' which watches the traffic of queries and collects the costs and 'learns' what elements or lack of them are



Proceedings of the 19th VLDB Conference
Dublin, Ireland, 1993

costly. Two technologies are available for this; neural networks, and rules induction. We chose the later technique, but believe the former should be investigated. This paper discusses some of the aspects of query governing with an induced rule based knowledge engine.

## Governing Engine

A rule based language takes the form of true or false antecedent conditions for each conclusions. The rules can be programmed to operate in a backwards or forwards chain. In backwards chaining the PRL engine attempts to find those true conditions which give rise to conclusions which themselves are needed as antecedents in other rules. The end condition being the final conclusion or goal of the process. In a data base query situation a rule might be ...

> IF sort by field IS detailed AND selection IS NOT aggregate AND no index IS TRUE THEN costly.

Another rule to find the values of the antecedents in the first rule might be ...

> IF field type IS numerical OR field name IS customer name THEN detailed

Notice that the first rule says that if we sort a query by a field with many different values it is an indicator of larger cost. Obviously sorting by a field like MONTH would result in a maximum of 12 lines of output. But sorting on a field like CUSTOMER NAME depends on the data itself. We could have 100 customers or millions of them in a mail order application. Induction is the process of finding out information like this from the data itself.

There are three phases to the induction process. The first is the collection phase. In this phase every query component is logged as an element of a data record, along with the resulting cost (in appropriate units, i.e., cpu seconds, excp's, etc.). In the second phase this collected file is submitted to the induction algorithm. We use the ID3 algorithm[1], which is based on an information entropy measurement. The resulting rules are then compiled in the Level5 production rule compiler, and the knowledge base established.

## System Maintenance and Operation

Periodically the collector file as it gathers more experience rebuilds the knowledge base. This is the basis for the claim that the system gets 'smarter' the longer it is used. It would seem that the 'outlier' queries would be the most important, and that only after one turned up could others be identified. This is partly true, but not as important as it seems, as the normal flow of queries with their statistical distribution of costs rapidly sets up the parameters needed for effective governing. The reason for this, is that the governor is not trying to predict the cost of a query, but rather whether the query is above a threshold. Upon system installation, a calendar is established with the acceptable costs on each shift each day of the week, for either batch or online operation. This results in about six cost regions. Grouping the samples in this way gives a larger number of observations to each region. The coarser grid is satisfactory because we are only interested in identifying a costly query and not estimating its exact cost.

## Future Investigations

The ID3 algorithm is capable of assigning a degree of confidence to a result. This occurs when seemingly similar samples result in different costs. The frequency of one result over another is the basis for the confidence factor. Today we discard any rule whose result is below a set of confidence limit. Another study area is the use of separate knowledge bases for each factor in the cost equation. That is, one set of rules for I/O usage, and another for CPU usage, with separate thresholds for each, or a combined one. A third area is how to most quickly prime the knowledge base. Could artificial queries be constructed? Should the collector phase while still operating during the governing phase only store the queries whose estimates were wrong?

---

1 Quinlan J.R. *A Case Study in D. Michie Expert Systems*. Edinburg University Press. 1979.