Abstract

An important issue in extending DBMS functionnalities is to allow the expression and execution of flexible queries in order to make these systems able to satisfy user needs more closely. To deal with this problem it is necessary to define new evaluation methods since standard (crisp) algorithms and access paths have proved to be inappropriate. A challenge in solving this problem is to keep the additional costs entailed by these new querying capabilities at a reasonable level. For this purpose we propose a set of query evaluation algorithms based on techniques expected on the one hand to restrict the amount of tuples to be transferred from disk to main memory and on the other hand to limit the computations which apply to these tuples. This work only concerns the evaluation of basic extended relational operators and may be seen as a first step towards the design of a fuzzy query optimizer. For each of the proposed methods, we suggest some useful measures for determining to what extent the corresponding algorithms are acceptable. Finally, we present an experimental framework for measuring the performance of these algorithms.

Keywords

DBMS, imprecise queries, query processing.

1. INTRODUCTION

A stream of growing interest in database research aims at increasing the usability of DBMS's by "fuzzy" querying systems which provide more flexibility, and therefore also allows more accuracy in characterization of information needs. Here, we view the situation in the framework of relational databases where the data are "crisp" and fully known. One criterion can be expressed as a combination of primary terms such as young, big, more or less equal, whose interpretation is based
on the fuzzy sets theory. Thus the challenge is no longer one of selecting tuples which reply in a precise way to a question, but is rather one of determining to what extent each tuple satisfies it, by evaluating their degree of satisfaction [BOSC 88b].

It seems that the conceptual and logical aspect of fuzzy querying have almost exclusively commanded the attention of researchers while the aspects of implementation have been neglected. As regards relational DBMS's, the performance factor, summarized by the response time, remains the determining measure. This factor is especially crucial in flexible querying systems since the performance may become unacceptable due to the highly numerical nature of the methods envisaged.

Indeed, the practice of qualifying data by using fuzzy predicates is based on numerical operators, dedicated to the measurement of a satisfaction degree. These operators lead to a considerable increase in the computation time for examining each tuple. Thus it becomes all the more advantageous to have techniques available which can limit the volume of tuples to be transferred from the disk to the central memory, and in the same way reduce the number of calculations to be carried out.

In answer to this problem, one solution lies in implementing associative rather than sequential access paths. Classical access accelerators (indexing, hash-coding, ...), whose principle is based on access to tuples in a relation according to the values of one or more attributes, cannot be directly adapted to fuzzy querying. Indeed, contrary to boolean criteria, fuzzy queries do not refer directly to entry values of the index. For example, the condition $\text{height} > 1.90\text{m}$ offers direct access to well identified values, which is not the case for the condition $\text{very tall}$.

Thus the problem is on the one hand to define access paths adapted to the evaluation of fuzzy query operators (restriction, join, ...), and on the other hand, to study the impact of fuzzy predicates on standard evaluation techniques, in order to define appropriate algorithms for evaluating imprecise queries. The final objective is to propose several operating mechanics of evaluation for each category of fuzzy queries. In a classic DBMS, the evaluation of relational operators is obtained by multiple algorithms using diverse strategies, depending notably on the type of the considered query and on the existing access paths. Similarly, in the case of a fuzzy querying system, it is desirable to have different algorithms for each class of question in order to be able to choose the most appropriate one at the time of evaluation. This paper provides a first step towards the design of a fuzzy query optimizor.