A (Gentle) Introduction to Deductive Databases

Shalom Tsur
University of Texas System, Center for High Performance Computing,
Austin, TX, USA

Abstract. This paper is intended as an introduction to deductive databases for the practitioner. After a brief historical overview it discusses the tradeoff between declarative and procedural programming and introduces the topic of deductive databases as an instance of this issue. The paper proceeds to expose some of the features of DD technology by means of the LDL system, shows some typical DD applications and concludes with a discussion of the state of the art and the relation of this technology to that of Object Oriented systems.

1 Introduction

The term deductive database is used to denote a technology that extends relational database technology and combines it with certain features of logic programming. The deductive database technology provides a powerful query system that subsumes the current practice of application development by means of a combination of procedural programs and embedded queries. Instead, it provides one uniform language that is powerful enough to specify the problem in its entirety and thus, enables global optimization, both over the application as well as the query portions of the program.

Deductive Databases (DD) evolved during the 80's. The background that led to this development was the maturing of the relational database technology and an appreciation of its strength as well as its weakness. The strength of the relational approach is clearly the declarative method of query specification as embodied in such query languages as SQL. The user specifies what subset of data needs to be retrieved from the database and leaves the how of it to the system itself. A user is thus absolved from the need to plan and optimize the access to the dataset of interest and this function is executed on her behalf by the database system itself. The result is a significant increase in the data independence and maintainability of the data over an extended time. On the downside, users became painfully aware of the limited expressive power of relational query languages and their inability to express certain classes of queries, in particular recursive queries.

Another influence on the evolution of DD technology was the parallel development in the field of logic programming, particularly the use of Horn clauses as a programming device. This lead to the idea of using logical rules
as an extension to the relational data model: rule heads express derived relations that can be used to express queries. The abstract queries that can be expressed using these rule extensions are compiled and optimized by the system into extended programs and queries against the underlying data. The rule extensions allow e.g., to express recursive queries. The seminal example is that of the ancestor relation shown in section 3.

This paper is intended as an introduction for the practitioner to the topic area of deductive databases. We provide an overview and illustrate some of the issues by means of examples. No attempt is made here to treat the formal aspects of the theory underlying the technology or the compilation/optimization methods that have been developed for it. After a general discussion of issues pertaining to procedural and declarative programming we move to an exposition of one particular DD system. The other two topics that will be briefly discussed are typical applications of the DD technology and the relation to Object Oriented systems. The LDL++ system, presently under development, combines logical rules with external data types defined as C++ classes.

The present status of technology today is that some of the prototype systems have reached a strength of pre-commercial deployment and at least one vendor has serious plans to develop it into a commercial product.

2 Declarative vs. Procedural Programming

Deductive database technology is one aspect of the larger issue of declarative vs. procedural programming. The latter style is the more common and is embodied in such programming languages as C, FORTRAN, COBOL and others. Essentially, procedural programming is a style of specification that requires how a problem is to be solved. In contrast, declarative programming is a style that requires what problem is to be solved and leaves the "how" of it to the system itself. Consequently, the procedural programming languages require the programmer to specify the problem of interest in terms of a sequence of steps, leading from the data to the result. These languages are featured with control statements that enable the programmer to modify the order in which the steps are executed. Using this style, the meaning of the program is determined by the order of specification and different sequences of the same steps have different meanings.

Declarative programming is perhaps best known in the form of query languages to relational database systems such as SQL. Using this language, a programmer specifies which set of the data is to be retrieved from the database and leaves the creation of an access plan and its optimization to the DBMS itself. To illustrate the differences in these programming styles