DLV^{DB}

A System for Advanced Database Applications

Ph.D. Thesis

Vincenzino Lio

December 22, 2008

Abstract

One of the most fundamental uses of a Database is to store and retrieve information, particularly when there is a large amount of data to be stored. Mining information and knowledge from large databases has been recognized by many researchers as a key research topic in database systems and machine learning fields, and by many industrial companies as an important area with an opportunity of major revenues.

Moreover, our capabilities of both generating and collecting data have been increasing rapidly. The widespread use of internet applications for most commercial activities, the computerization of many business and government transactions, and the advances in data collection tools have provided us with huge amounts of data. Millions of databases have been used in business management, government administration, scientific and engineering data management, and many other applications. It has been observed that the number of such databases keeps growing rapidly because of the availability of powerful and affordable database systems. This explosive growth in data and databases has generated an urgent need for new techniques and tools that can intelligently and automatically transform the processed data into useful information and knowledge.

In this scenario, a mounting wave of data intensive and knowledge based applications, such as Data Mining, Data Warehousing and Online Analytical Processing (OLAP) have created a strong demand for more powerful database languages and systems. This led to the definition of both several data model extensions (e.g., the Object Relational model), and new language constructs (e.g., recursion and OLAP constructs), and various database extenders (based, e.g., on user defined functions), to enhance the current Database Management Systems DBMSs.

However, state of the art DBMSs are still not powerful and general enough for many advanced database applications. Indeed, there is a variety of systems and efficient techniques to store and retrieve data, but the problem of understanding and interpreting this large amount of information is still complex, particularly when the data belong to complex domains. Moreover, a number of applications that have a database ”flavor”, are not well addressed by conventional database management systems. The explosive growth of new database applications has, in several cases, outpaced the progress made by database technology. The great success of new application areas often serves as a grim reminder of the limitations from which DBMSs still suffer in terms of expressive power. For instance, the newly introduced Object Relational (O-R) systems offer great improvements in generality, extensibility, and query power; yet O-R systems do not fully support data mining applications. Summarizing, current Database Systems don’t have reasoning modules in order to extract complex and more complete knowledge from the underlying data. To solve this problem, a mechanism for reasoning about the stored information is necessary. This desirable mechanism must be capable of managing and handling very large amounts of information, as well as of performing sophisticated inference tasks, and of drawing the appropriate conclusions.

These reasoning capabilities can be provided by logic-based systems. In particular, logic programming provides a powerful formalism capable of easily modelling and solving complex problems. While research in this area initially had mainly a theoretical impact, the recent development of efficient Anwer Set Programming (ASP) systems like DLV [3] and Smodels [6] has renewed the interest in the area of non-monotonic reasoning and declarative logic programming for solving real world problems in a number of application areas. As a consequence, they can provide the powerful reasoning capabilities needed to solve novel complex database problems.

However, many of the interesting problems are either "data intensive", such as the automatic correction of census data [4], or they produce huge ground programs that can not be handled in a typical logic programming main-memory data structure. As an example, elaboration on Scientific Databases in order to detect or extract chemical structure (e.g., cellular interactions or to analyze the medical histories in order to verify the possible causes of diseases (e.g., radioactivity or genome malformation) are typical data intensive applications that cannot be efficiently managed neither by main memory logic systems nor by traditional databases.

Summarizing:

- Logic-based systems evaluate logic programs efficiently in main memory, but are tuple-at-a-time, and therefore inefficient with respect to disk accesses.
- In contrast, database systems can implement only a non-recursive subset of logic programs (essentially described by relational algebra), but do so efficiently with respect to disk accesses (set-at-a-time).
- The expressive power of logic-based systems allows to easily write declarative solutions to complex problems which cannot be solved by database systems.
- In typical database applications, the amount of data is sufficiently large that much of it is on secondary storage. Efficient access to this data provided by database systems is, then, crucial to achieve good performance.

The considerations above put into evidence that efficient and effective data management techniques that combine Logic Inference Systems with Database Management Systems, are mandatory. In particular, there is the need of combining the expressive power of logic-based systems with the efficient data management features of DBMS*s*. Indeed, logic-based systems (such as the ASP ones) provide an expressive power that goes far beyond that of SQL3, whereas good DBMS*s* provide very efficient query optimization mechanisms. In the literature Deductive Database Systems have been proposed to combine these two realities. In practice, deductive database systems are an attempt to adapt typical Datalog systems, which have a "smalldata" view of the world, to a "largedata" world via intelligent interactions with some DBMSs.

In more detail, Deductive Database Systems are advanced forms of database management systems whose storage structures are designed around a logical model of data and inference modules are designed on logic programming systems. Deductive databases not only store explicit information in the manner of a relational database, but they also store rules that enable inferences based on the stored data. Using techniques developed for relational systems in conjunction with declarative logic programming, deductive databases are capable of performing reasoning based on that information.

There are many application areas for deductive database technology. In general, deductive database technology allows to analyze more efficiently large amounts of information. This property, makes deductive systems a good solution for several data intensive problems. Deductive database technology is an appropriate solution also to decisional problems. Decision support systems are needed by the organizations in order to reason effectively about plans for the present and future product activities. Another application area, with similar properties of the former one, concerns Planning systems. For example, guidance systems require the ability to generate round-the-world trip and the possibility to explore alternatives and hypotheses solutions. Another fruitful application area is that of expert systems. As the former problems, expert systems require to manage large amounts of data but, in this case, the amount of facts can be distilled by a simple yet tedious analysis to prevent the user to explore irrelevant problem solutions (based on past experience).

Despite their potential, the development of deductive database systems did not receive much attention in the literature, mainly due to the high difficulties in obtaining efficient and effective systems.

Objectives and contributions

This work provides a first, initial contribution in the area of Deductive Database systems, bridging the gap between ASP systems and DBMS*s*; indeed, it first provides an overview of the state of art datalog systems and of various technics that led to the successful implementation of some deductive database systems; in particular, we point out the problems concerning code optimization, program rewriting, efficient plan for query execution and other operations on the data which are crucial for a successful development of a deductive database system. After this we present the deductive database system DLV^{DB} , developed in this work.

DLV^{DB} has been conceived to increase the cooperation between ASP systems and databases. As such, it allows substantial improvements in both the evaluation of logic programs and the management, within an existing ASP system (namely, DLV), of input and output data distributed on several databases.

DLV^{DB} has been developed as an extension of the well known ASP system DLV and, as such, it combines the experience in optimizing logic programs gained within the DLV project with the well assessed data management capabilities of existing DBMS*s*. This makes it well suited to be applied on both complex and data intensive applications. Moreover, its architecture has been designed so as to allow further extensions to the more powerful capabilities of the DLV language, such as disjunctions.

Presently, DLV^{DB} allows for two typologies of execution: *(i)* direct database execution, which evaluates logic programs directly on the databases, with a very limited usage of main memory, and *(ii)* main memory execution, which loads input data from different (possibly distributed) databases and executes the logic program directly into the main memory.

From this considerations it is possible to observe that, DLV^{DB} provides a well established infrastructure for the interoperation with databases and allows the application of several optimization techniques already developed or under development in the DLV project (such as magic sets [1, 2, 5, 7, 8]). Moreover, as it will be shown in the experimental evaluation, it gives both important speed ups in the running times for several relevant problems and the capability to handle greater data sets, w.r.t. previously developed, both logic-based and database, systems.

Currently, the DLV^{DB} system is successfully exploited as the core reasoning module within the Infomix System, a system for Information Integration developed within the project funded by the European Commission IST-2001-33570 INFOMIX project (see chapter **??** for a complete description of this Information Integration System).

Summarizing, the overall contributions of this work are the following:

- A comprehensive survey of existing logic-based systems;
- The development of a fully fledged system enhancing in different ways the interactions between logic-based systems and $DBMS_s$;

- The development of a novel evaluation strategy for logic programs allowing to minimize the usage of main-memory and to maximize the advantages of optimization techniques implemented in existing $DBMS_s$;
- The definition of a framework for carrying out an experimental comparative analysis of the performance of the existing systems (both ASP, DDS and DBMS) and of DLV^{DB} .

Bibliography

[1] F. Bancilhon, F. Maier, Y. Sagiv, and J. Ullman. Magic sets and other strange ways to implement logic programs. In *Proc. of the ACM Symposium on Principles of Database Systems (PODS'86)*, pages 1–16, Cambridge, Massachusetts, 1986. ACM Press.

[2] C. Beeri and R. Ramakrishnan. On the power of magic. *J. Logic Programming*, 10(3/4):255–299, 1991.

[3] T. Dell'Armi, W. Faber, G. Ielpa, C. Koch, N. Leone, S. Perri, and G. Pfeifer. System Description: DLV. In T. Eiter, W. Faber, and M. aw Truszczyński, editors, *Logic Programming and Nonmonotonic Reasoning — 6th International Conference, LPNMR'01, Vienna, Austria, September 2001, Proceedings*, number 2173 in Lecture Notes in AI (LNAI), pages 409–412. Springer Verlag, September 2001.

[4] E. Franconi, A. L. Palma, N. Leone, S. Perri, and F. Scarcello. Census Data Repair: a Challenging Application of Disjunctive Logic Programming. In *In* Proc. of LPAR 2001, pages 561–578, 2001.

[5] I. Mumick, S. Finkelstein, H. Pirahesh, and R. Ramakrishnan. Magic conditions. *ACM Trans. Database Systems*, 21(1):107–155, 1996.

[6] I. Niemelä, P. Simons, and T. Syrjänen. Smodels: A System for Answer Set Programming. In C. Baral and M. Truszczyński, editors, *Proceedings of the 8th International Workshop on Non-Monotonic Reasoning (NMR'2000)*, Breckenridge, Colorado, USA, April 2000.

[7] K. Ross. Modular stratification and magic sets for datalog programs with negation. In *Proc. of the ACM Symposium on Proncïples of Database Systems*, 1990.

[8] D. Saccà and C. Zaniolo. Magic counting methods. In *Proc. of the ACM SIGMOD Annual Conference on Management of Data (SIGMOD '87)*, pages 49–59, San Francisco, CA, 1987. ACM Press.