Chapter VIII
ROL2: Towards a Uniform Deductive Object–Oriented Database Language

Mengchi Liu
Wuhan University, China

ABSTRACT

Deductive object-oriented databases are intended to integrate the deductive and object-oriented database techniques to combine the best of two approaches and to overcome their inherent shortcomings, with a number of deductive object-oriented database languages proposed. However, most of these languages are only structurally object-oriented. Important behaviorally object-oriented features such as methods and encapsulation common in object-oriented database systems are not properly supported. This chapter presents a novel deductive object-oriented database language called ROL2 that extends structurally object-oriented database language ROL with all behaviorally object-oriented features. It supports in a rule-based framework all important object-oriented features such as object identity, complex objects, typing, information hiding, rule-based methods, encapsulation of such methods, overloading, late binding, polymorphism, class hierarchies, multiple structural and behavioral inheritance with overriding, blocking, and conflict handling. It is so far the only deductive object-oriented database language that supports all these features in a uniform rule-based framework.

INTRODUCTION

Deductive databases and object-oriented databases are two important extensions of the traditional database technology. Deductive databases result from the integration of relational database and logic programming techniques. The main attraction of the relational database technique is that it is built around simple and natural mathematical structure — the relation that allows efficient sec-

ondary storage access, and set-oriented, high-level query languages, with rigorous mathematical foundations (Codd, 1970).

Logic programming is direct outgrowth of earlier work in automatic theorem proving and artificial intelligence. It uses logic to represent knowledge and uses deduction to solve problems by deriving logical consequences. The most well-known and widely used logic programming language is Prolog (Colmerauer, 1985; Kowalski, 1988; Sterling & Shapiro, 1986), which uses the Horn clause subset of first-order logic as programming language and the resolution principle as a method of inference with well-defined model-theoretic and proof-theoretic semantics (Lloyd, 1987).

Important studies on the relations between logic programming and relational databases have been conducted since 1970s, mostly from theoretical point of view (Gallaire & Minker, 1978; Gallaire, 1981; Jacobs, 1982; Ullman, 1982; Maier, 1983). Relational databases and logic programming have been found quite similar in their representation of data at the language level. They have also been found complementary in many aspects. Relational database systems are superior to the standard implementations of Prolog with respect to data independence, secondary storage access, concurrency, recovery, security and integrity (Tsur Zaniolo, 1986). The control over the execution of query languages is the responsibility of the system, which uses query optimization and compilation techniques to ensure efficient performance over wide range of storage structures. However, the expressive power and functionality of relational database query languages are limited compared to that of logic programming languages. Relational languages do not have built-in reasoning capabilities. Also, relational query languages are often powerless to express complete applications, and are thus embedded in traditional programming languages, resulting in impedance mismatch (Maier, 1987) between programming and relational query languages. Prolog, on the other hand, can be used as general-purpose programming language. It can be used to express facts, deductive information, recursion, queries, updates, and integrity constraints in uniform way (Reiter, 1984; Sterling &Shapiro, 1986).

Deductive databases combines the benefits of both logic programming and relational databases, such as representational and operational uniformity, reasoning capabilities, more expressive declarative query language, efficient secondary storage access, etc. The function symbols of Prolog, which are typically used for building recursive functions and complex data structures, have not been found useful for operating over relational databases made up of flat relations. As result, restricted form of Prolog without function symbols called Datalog (with negation), with well-defined declarative semantics based on the work in logic programming, has been widely accepted as the standard deductive database language (Ceri et al., 1990; Ullman, 1989).

Object-oriented databases extend the data modeling power of the traditional databases by means of number of novel data modeling mechanisms such as object identity, complex objects, classes, class hierarchy, and inheritance. They integrate both structural and behavioral parts into uniform framework and provide better way to organize and manipulate structured objects. Examples of object-oriented languages and systems are Iris (Fishman et al., 1987), Exodus (Carey et al., 1988), GemStone (Butterworth et al., 1991), Orion (Kim, 1990), O2 (Deux et al., 1991), ObjectStore (Lamb et al., 1991), ONTOS (Soloviev, 1992), Jasmine (Ishikawa et al., 1993). ODMG-93 (Cattell, 1996), and ODMG 2.0 (Cattell & Barry, 1997).

However, both deductive databases and object-oriented databases have shortcomings. Deductive databases lack the powerful data modeling mechanisms offered by object-oriented databases, while object-oriented databases lack built-in reasoning capabilities and expressive declarative query language with firm logical foundation.