Implementation of constraint database systems using a compile-time rewrite approach

Diss. ETH No. 11656

Implementation of Constraint Database Systems Using a Compile-Time Rewrite Approach

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY
ZÜRICH

for the degree of
Doctor of Technical Sciences

presented by
ROMAN GROSS-BRUNSCHWILER
Dipl. Informatik-Ing. ETH
born 31. October, 1964
citizen of St.Gallen-Tablat (SG)

accepted on the recommendation of
Dr. R. Marti, examiner
Prof. Dr. M.J. Maher, Prof. Dr. H.-J. Schek, co-examiners

1996

Abstract

The declarative style of database query languages is an important aspect of deductive or relational database systems. However, the declarative spirit is usually restricted to the specification of derived relations (views) and sets of answers satisfying specified queries. Base predicates are explicit enumerations of tuples that satisfy the relations, i.e. the extension of these predicates. Such systems do not allow the representation of infinite extensions, which arise for example in temporal or spatial databases.

Constraint database systems (CDBS), a recent line of research, overcome these limitations by not simply viewing a fact as a tuple of values, but as a conjunction of constraints on the values of the relation's attributes. In doing so, relations may be given implicitly, namely by formulas which specify the tuples which satisfy the relations.

The implementation of constraint database systems is the main topic of this thesis. We propose a program transformation technique, Constraint Lifting, which can be used to rewrite constraint databases at compile-time such that no constraint solving facilities are necessary during query evaluation. I contrast to other rewrite techniques (e.g. Magic Templates) we do not primarily focus on efficiency. Instead, we strive to enhance the power of the query language by supporting constraints which are insufficiently instantiated to be evaluated. Such constraints are propagated up the evaluation tree into depending clauses. To gain efficiency, we only consider the groundness of the variables contained in the constraints instead of the actual values bound to these variables at run-time.

We identify the requirements for constraint solvers which are able to simplify constraints during constraint lifting. Such solvers have to guarantee equivalence between input and output constraint sets only with respect to variables relevant for query answering. For the manipulation of negated literals, we propose a technique based on both constructive negation and delayed evaluation similar to constraint lifting.

On a practical level, we present the design and the implementation of the DeCoR system, a prototype of a deductive DBMS with constraints over reals. The DeCoR system exemplifies how the constraint lifting algorithm enables us to build a constraint database system as a front-end to a deductive or relational DBMS. In doing so, we benefit from the typical database features provided by the underlying DBMS. Moreover, this architecture allows the integration of data represented by constraints with data represented in a traditional style, an ability which is necessary in most realistic applications.

Zusammenfassung

Deklarativität ist ein Merkmal der meisten Abfragesprachen deduktiver und relationaler Datenbanksystemen. Ueblicherweise beschränkt sich die Deklarativität jedoch auf die Spezifikation abgeleiteter Relationen (Views) und Antwortmengen zu Datenbankanfragen. Die Basisprädikate müssen jedoch durch explizite Aufzählung der dazugehörigen Datensätze definiert werden, d.h. es müssen die Extensionen der Prädikate angegeben werden. Deshalb ist es in diesen Systemen nicht möglich Prädikate mit unendlicher Extension, wie sie etwa in temporalen oder räumlichen Datenbanken vorkommen können, darzustellen.

Constraint Datenbanksysteme (CDBS) überwinden diese Einschränkung, indem ein Fakt nicht mehr als einfaches Wertetupel aufgefasst wird, sondern als eine Konjunktion von Bedingungen (Constraints) auf den Attributwerten. Dadurch können Basisprädikate auch implizit definiert werden, d.h. durch Formeln, die spezifizieren welche Tupel dazugehören.

Die Implementation von Constraint Datenbanksystemen ist das Hauptthema dieser Arbeit. Insbesondere stellen wir eine auf Programmtransformationen basierende Methode vor, genannt "Constraint Lifting", die es uns ermöglicht, Constraint Datenbanken zum Zeitpunkt der Uebersetzung von Klauseln so umzuschreiben, dass während der Anfrageauswertung kein Vereinfachen respektive Lösen von Constraints nötig ist. Im Gegensatz zu anderen Transformationstechniken (z.B. Magic Templates) liegt der Schwerpunkt unseres Ansatzes nicht auf der Verbesserung der Effizienz der Anfrageauswertung, sondern auf der Steigerung der Ausdrucksmächtigkeit. Dies wird erreicht, indem auch die Verarbeitung von Constraints unterstützt wird, welche nicht direkt ausgewertet werden können, da sie (noch) nicht genügend instanziert sind. Solche Constraints werden entlang des Auswertungsbaumes in abhängige Regeln propagiert was zu einer Verzögerung der Auswertung führt. Zur Verbesserung der Effizienz dieses Ansatzes abstrahieren wir von den während der Auswertung konkret vorhandenen Werten gebundener Variablen und betrachten nur, ob die Variablen gebunden sein werden oder nicht.

Zur Implementation eines konkreten CDBS ist auch eine Komponente zur Lösung von Constraints (Constraint Solver) erforderlich, die in Verbindung mit dem Constraint Lifting Algorithmus eingesetzt werden kann. Wir identifizieren die Anforderungen, die ein solcher Constraint Solver erfüllen muss. Eine besondere Eigenschaft ist, dass die Aequivalenz zwischen der ursprünglichen und der vereinfachten Menge von Constraints nur mehr bezüglich einer eingeschränkten Menge von Variablen gewährleistet sein muss. Zur Verarbeitung von negierten Literalen präsentieren wir einen Algorithmus, der fallweise eine konstruktive Negation durchführt oder die Auswertung der negierten Literale analog zum Constraint Lifting Algorithmus verzögert.

Als konkrete Anwendung der vorgestellten Techniken haben wir das DeCoR System, einen Prototyp eines deduktiven Datenbanksystems mit Constraints über reellen Zahlen, entworfen und implementiert. DeCoR zeigt exemplarisch wie der Constraint-Lifting Algorithmus verwendet werden kann um ein Constraint Datenbanksystem als Frontend zu einem deduktiven oder relationalen DBMS zu realisieren. Eine solche Architektur ermöglicht es, die im darunterliegenden DBMS vorhandene Datenbankfunktionalität auszunutzen. Ebenso wird eine integrierte Verarbeitung von implizit und explizit dargestelltem Wissen sehr einfach möglich.