SMACS
Schema Mapping Application Compiler System

Exposé einer Diplomarbeit

Guido Draheim
27. Februar 2004

Betreuer Prof. Dr. Ulf Leser
Institut für Informatik, Humboldt-Universität zu Berlin

Zeitraum 27. Februar 2004 – 27. August 2004

Motivation

Die Forschung zur automatisierten Erstellung und Verarbeitung von Abbildungen der Schema relationaler Datenbanksysteme (Schema-Mapping) ist vermutlich so alt wie die Nutzung relationaler Datenbanksysteme selbst, treten diese Prozesse doch immer auf, wenn verschiedene Datenbanken ganz oder teilweise die gleichen Dinge und Konzepte ihres Einsatzfeldes modellieren. Die tatsächliche Umsetzung in Elemente und Relationen der Datenbanken kann dabei so verschieden ausfallen, dass sich Einträge nicht mehr mit einfachen Holen/Einfügen Sequenzen übertragen lassen. Die Elemente und Relationen passen nicht mehr eineindeutig, und Einträge müssen beim Übertrag Transformationen unterworfen werden.

Die durchzuführenden Transformationen können sehr umfangreich und ineinander verschränkt sein. Für die gesicherte und effiziente Umsetzung braucht es ein Ausführungssystem, das Beschreibungen der notwendigen Abbildung interpretiert und in tatsächliche Aktionen umsetzt. Dabei können Aktionen angestoßen werden, die über klassische SQL Operationen hinausgehen. Dies ist etwa bei schematischen Heterogenitäten zu beobachten, die Abbildungen höherer Ordnung erfordern.

Zielsetzung

Die Möglichkeiten zum Schema Mapping sind zahlreich. Die auszuführenden Abbildungen zu finden, ist dieser Arbeit vorgelagert. Hier wird davon ausgegangen, dass die gewünschte Abbildung der Datenbank Schema vorliegt. Es müssen maschinelle Beschreibungsformen und -sprachen gefunden werden, die die Transformationen ausdrücken. Diese werden durch einen Compiler interpretiert und in Programme übersetzt, die die eigentliche Abbildung durch Ausführung von Teiloperationen hinreichend vornehmen.

Dabei ist zu untersuchen, wie Teiloperationen einer Ausführungsumgebung eingebunden werden können. Verschiedene Kombinationen können erweiterte und optimierte Formen der generierten Schema Mapping Applikationen finden. Es ist ein Ausführungssystem zu erstellen, dass eine reale Klasse von Schema Mapping Abbildungen vornehmen kann und auf einer verbreiteten Programmumgebung aufsetzt. Hier wird beispielhaft das Mapping heterogener Schema mit generierten Programmen in SQL/J gewählt.

Es gibt einige Vorarbeiten im Bereich der Durchführung von Abbildungen heterogener Schema, die auf Standard SQL erster Ordnung abbilden. [LSS96][BM01][Gen03][KR01]. Der Gesamtprozess der Ausführungen bindet weitere Teiloperationen ein, etwa die

einer Materialisierung von Strukturen als Tabellen. Dies wird vermutlich auch für andere Schema Abbildungen höherer Ordnung benötigt werden. [Ko00][RD00]. Daher wird diese Problemklasse von Abbildungen heterogener Schema beispielhaft gewählt.

Augenmerk wird vornehmlich auf die Formulierung der Operationen, der Kombination und der Erweiterbarkeit gelegt. Hier werden beispielhaft Skalierungsoperationen mit einbezogen, die Transformationen der Felder selbst darstellen. Die Einbindung erfolgt mit Möglichkeiten von Java Plattformen, weshalb die Wahl auf SQL/J zur Generierung der ausführbaren Programme der Schema Abbildungen fiel. Die generierten Programmtexte sind voraussichtlich leicht lesbar, prüfbar und potentiell in nachgelagerte Prozesse integrierbar und/oder dort veränderbar.

Vorgehen

Die im Compilerkontext und/oder Applikationskontext einzubindenden Teiloperationen können sehr vielfältig sein. Mindestens will man Beispiele der oben genannten Problemklassen verarbeiten können. Aus den Vorarbeiten (siehe Verweise insbesondere [Gen03]) erschließen sich folgende Operationen, von denen einige in der Arbeit näher untersucht werden:

Sql Rule Inspection: Schema Abbildungen in SQL und davon abgeleiteten Dialekten sind auf Merkmale wie spezielle Variablen und Funktionsaufrufe zu untersuchen, die entsprechenden Stellen als Referenzen zu markieren und bearbeitbar zu speichern.

Schema Materialization: Schema von Datenbanken sind selbst als Tabellen zu speichern, und für spätere Operationen abfragbar bereitzustellen. Eine Reihe von Zwischenstrukturen können als Listen gespeichert werden. Die Zwischendaten müssen voraussichtlich im Compiler und der Schema Mapping Applikation verwaltet werden.

Sql Rule Unroll: Enthaltene Teile von Abbildungen (gewöhnlich in SQL oder äquivalenten Beschreibungen) müssen mit eingebundenen Funktionen und Zwischentabellen verbunden werden. Dazu werden markierte Stellen erweitert mit Ausformungen von Code Stücken aus Schablonen. Anderes Ausrollen erfordert das Vervielfältigen in ähnliche Teilabbildungen und Einbettung in Java Verbundstücke.

Template Macro System: Die Schablonen für eingebunde Funktionen und Bindung an Zwischendaten sind in einer (separaten) Beschreibungsform zu halten. Es ist dazu ein Interpretersystem (im Compiler) zu finden, das diese ausformen kann und in Teile von Abbildungen einbetten kann, die eine ausführbare Applikation ergeben.

Foreign Function Handling: Externe (Java) Funktionen müssen eingebunden werden. Diese müssen in ihren Eigenschaften beschrieben werden, insbesondere wo sie im SQL Zielsystem vorliegen, durch einen SQL Interpeter gerufen werden können oder ob die Operationen auf den Datensätzen in Java Schleifen einzeln verarbeitet werden müssen.

Rule Head Matching: Das Ausrollen mit Ausformungen der Schablonen unter Nutzung von Funktionen erfordert das Untersuchen von Abbildungen und darin erreichbarer und/oder markierter Stellen. Dazu ist eine geeignete Adressierung der Stellen und eine Ausdruckssprache zu Bewertungen zu finden, die Bedingungen hervorbringt, an denen Compiler Operationen angestoßen werden.

Generation Variation: die bearbeiteten Abbildungen im Compiler werden in einer Applikationssprache ausformuliert, hier SQL/J. Das Ausführungssystem kann in verschiedenen Varianten repräsentiert werden. Dies wären beispielsweise unterschiedliche Behandlung von Records, die Fehler in eingebundenen Teiloperationen bewirk(t)en. Variationen können etwa Logging, Exception Callbacks oder Recovery über Alternativen im Ausführungsblock sein.

Verweise

[LSS96] L.V.S. Lakshmanan, F. Sadri, S.N. Subramanian:
„SchemaSQL: An extension to SQL for multidatabase interoperability“
VLDB'96 - Conference on Very Large Databases, 1996
http://citeseer.ist.psu.edu/lakshmanan96schemasql.html

[BM01] Francois Barbancon, Daniel P. Miranker:
„Implementing Federated Databases Systems by Compiling SchemaSQL“
Submission Report, University of Texas, Dept. Of Computer Science, 2001
IDEAS'02 – International Dabase Engineering and Applications Symposium, 2002
http://www.cs.utexas.edu/users/francois/ideas05.pdf http://citeseer.nj.nec.com/535654.html

[Gen03] Li Geng:
„Compiling SchemaSQL into SQL“,
Project Report, University of Texas, Dept. Of Computer Science, 1/2003
http://www.cs.utexas.edu/users/ligeng/Projects/SchemaSQL/CompilerReport.doc

[KR01] Andreas Koeller, Elke Rundensteiner:
„Incremental Maintenance of Schema-Restructuring Views in SchemaSQL“
Technical Report WPI-CS-TR-00-25, Worcester Polytechnic Institute, Dept. Of Computer Science, 2002
EDBT'02 – International Conference On Extending Database Technology, 2002, Prag
http://www.cs.wpi.edu/Resources/techreports/[WPI-CS-TR-00-25] http://davis.wpi.edu/~dsrg/MEMBERS/KOELLER/SchemaSQL.html

[Ko00] Charibeth Y. Ko:
„Three Approaches To A Multidatabase System“
PCSC'00 – Philippine Comuting Science Congress, 2000
http://www.pst.dlsu.edu.ph/ces/capt_CSP/docs/proceedings/presentation/ThreeApproaches.pdf

[RD00] Eberhard Rahm, Hong-Hai Do:
„Data Cleaning: Problems and Current Approaches“
IEEE Bulletin of the Technical Committee on Data Engineering, Vol 23 No. 4, 2000
http://dol.uni-leipzig.de/pub/2000-45 http://citeseer.nj.nec.com/context/1770019/0