Datenbank Spektrum (2010) 10: 105–106
DOI 10.1007/s13222-010-0013-7

DATENBANKEN VORGESTELLT

Der Lehrstuhl für Datenbanksysteme am Wilhelm-Schickard-Institut der Universität Tübingen

Torsten Grust

Online publiziert: 29. Juni 2010

Schlüsselwörter Deklarative Datenbanksprachen · Datenbankgestützte Programmausführung · Anfrageverstehen und -debugger

1 Der Lehrstuhl

Der Lehrstuhl der Tübinger "Datenbanker" ist am Wilhelm-Schickard-Institut für Informatik (WSI) der Eberhard Karls Universität Tübingen angesiedelt. Neben Datenbanktechnologie wird am WSI noch in 16 weiteren Arbeitsgruppen Informatik gelehrt und erforscht. Die Tübinger Informatiker profitieren von der vielfältigen Forschungsumgebung einer echten Volluniversität – es bestehen zahlreiche und intensive Kontakte zu Kollegen, die über das WSI hinaus etwa in die Biologie, Medizin, Medien- und Erziehungswissenschaften oder die Psychologie reichen.

Seit einem Wechsel von der Technischen Universität München im August 2008 leitet Torsten Grust den Lehrstuhl für Datenbanksysteme am WSI. Derzeit (Mai 2010) sind sechs weitere Doktoranden und Postdoktoranden in der Arbeitsgruppe beschäftigt:

- George Giorgidze (von Chalmers U / U Nottingham),
- Dr. Melanie Herschel (von HU Berlin / HPI Potsdam),
- Manuel Mayr (von TU München),
- Jan Rittinger (von U Konstanz / TU München),
- Tom Schreiber (von TU München) und
- Jeroen Weijers (von U Utrecht).

T. Grust (✉)
Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Sand 13, 72076 Tübingen, Deutschland
e-mail: torsten.grust@uni-tuebingen.de
url: www-db.informatik.uni-tuebingen.de

2 Forschungsschwerpunkte

Müsste der Lehrstuhl seinen Fokus in der Forschung in 140 Zeichen über Twitter mitteilen, würde die Nachricht wohl *Aspekte der Übersetzung, Einbettung, effizienten Ausführung und des Verstehens (Debugging) ausdruckskräftiger, deklarativer Datenbanksprachen* lauten.[1] Tatsächlich bewegt sich unsere Forschungstätigkeit vor allem in einem spannenden Grenzgebiet, das zwischen relationaler Datenbanktechnologie und den (funktionalen) Programmiersprachen liegt. Während der Lehrstuhlinhaber und die meisten wissenschaftlichen Mitarbeiter einen klassischen Datenbankhintergrund besitzen, sind unsere neuesten Verpflichtungen – George Giorgidze und Jeroen Weijers – gezielt in der *Functional Programming Community* gesucht und gefunden worden. Eine Vielzahl von Techniken, die innerhalb dieser Community entwickelt wurden, sind sehr relevant, wenn es um die effiziente relationale Realisierung von deklarativen Anfrage- und Datenbanksprachen geht. Umgekehrt haben die Programmier sprachler erkannt, wie Datenbanktechnologie beitragen kann, die Ausführung sehr datenintensiver Programme deutlich zu beschleunigen bzw. überhaupt erst zu ermöglichen.

In diesem Spannungsfeld – im positivsten Sinne – befinden sich die meisten der folgenden Forschungsprojekte, die wir hier chronologisch nach ihrem Startzeitpunkt skizzieren.

Pathfinder (seit 2001, www.pathfinder-xquery.org) Mit *Pathfinder* hat die Arbeitsgruppe einen Compiler für XQuery entwickelt, der sich eng am Sprachstandard des W3C orientiert und Aspekte wie Dokumentordnung, Sequenzordnung, Semantik aller XPath-Achsen [2], Knotenkonstrukti-

[1] Die Mitglieder des Lehrstuhls twittern Neuigkeiten zu Lehre und Forschung unter dem Account `DBatUTuebingen`.

on und die orthogonale Kombinierbarkeit der Konstrukte einer ausdrucksorientierten Sprache wie XQuery ernst nimmt. *Pathfinder* generiert Zwischencode in einer einfachen algebraischen Zwischensprache, die sich an den tatsächlichen Fähigkeiten moderner relationaler Datenbankkerne orientiert [3]. Über die Zeit wurden Codegeneratoren für eine Vielzahl – klassischer und exotischer – relationaler Back-Ends entwickelt, darunter IBM DB2, Microsoft SQL Server, PostgreSQL oder kdb+ von kx Systems. Weiteste Verbreitung hat *Pathfinder* aber als *MonetDB/XQuery* gefunden, einem Projekt, in dem wir den Compiler zusammen mit dem Hauptspeicher-optimierten *Column Store MonetDB* (CWI Amsterdam) als Back-End betreiben. Wir sind stolz, dass *MonetDB/XQuery* zu den effizientesten und skalierbarsten XQuery-Prozessoren gehört, die in der bzgl. XML und XQuery sehr aktiven letzten Dekade konstruiert wurden [1]. Unsere Kollaboration mit den Kollegen in Amsterdam (CWI und VectorWise B.V.) ist weiterhin sehr intensiv.

Ferry (seit 2008, www.ferry-lang.org) *Ferry* untersucht die effiziente Übersetzung und Ausführung listenbasierter funktionaler Programmiersprachen mit Unterstützung relationaler Datenbanktechnologie. Letztendlich verfolgen wir das Ziel, das Laufzeitsystem von Programmier- und Skriptsprachen zu entlasten, indem Fragmente datenintensiver Programme direkt vom Datenbankkern ausgeführt werden. Sowohl für Datenstrukturen wie etwa beliebig tief verschachtelte Listen, Arrays oder Hashes als auch für typische Operationen auf diesen Strukturen findet *Ferry* effiziente relationale Repräsentationen. Grundlegende Arbeiten aus dem *Pathfinder*-Projekt helfen uns, derartige Programmfragmente in effizienten und kompakten algebraischen Code zu übersetzen [4].

Derzeit bestimmen vor allem unsere Arbeiten an *Ferry* die Forschungstätigkeit des Lehrstuhls. Auf Basis von *Ferry* konnten wir bereits neue mächtige *Provider* für Microsofts *LINQ* (*Language-Integrated Query*) konstruieren. Ein neuer SQL-Codegenerator für Philip Wadlers funktionaler Multi-Tier-Programmiersprache *Links* (U Edinburgh) ist nahezu komplett. Darüberhinaus entsteht eine nahtlose Einbettung von *Ferry*-basierten Übersetzungstechniken in *Ruby*, die einem Framework wie *Ruby on Rails* einen Grad an Datenbankfunktionalität verleiht, der deutlich über die aktuellen Technologien *ARel* oder *ActiveRecord* hinaus geht.

Die Arbeiten an *Ferry* stehen der *Functional Programming Community* besonders nahe und es besteht ein lebhafter Austausch. Ganz aktuell hat sich eine Verbindung zwischen *Ferry* und *Data Parallel Haskell* gezeigt, die wir in der kommenden Zeit ausloten werden.

Nautilus (seit 2009, www.nautilus-system.org) Entwickler komplexer Datentransformationen werden mit Hilfe von *Nautilus* in die Lage versetzt, ihre Transformationen zu analysieren, insbesondere um Fehler aufzudecken. *Nautilus* verfolgt somit ähnliche Ziele wie *Debugger*, die für prozedurale Programmiersprachen eingesetzt werden. Allerdings verwendet *Nautilus* aufgrund der deklarativen Spezifikation von Datentransformationen einen neuartigen Ansatz, der auf der *Datenherkunft (Data Provenance) erwarteter Daten* im Ergebnis der Transformation basiert. Zusätzlich zur Debugging-Funktionalität untersuchen wir, wie *Nautilus* Transformationsänderungen vorschlagen kann, die defekte Transformation reparieren können. Schließlich verwaltet *Nautilus* die Historie des iterativen Debuggingprozesses und verwendet diese in Folgeiterationen, um bessere Analyseergebnisse und Änderungsvorschläge zu erzielen.

Bisher lag der Forschungsschwerpunkt auf der Analyse-Funktionalität [5]. Ein wesentliches Ergebnis sind die Definition und Evaluation effizienter Algorithmen, die für eine signifikante Teilmenge von SQL-Anfragen erklären, warum erwartete Daten *nicht* im Ergebnis einer SQL-Anfrage vorkommen.

Der Grundstein für *Nautilus* wurde in Kooperation mit IBM Almaden und der UC Santa Cruz in Kalifornien gelegt. Diese Kollaboration ist weiterhin sehr aktiv.

Exploration sozialer Netzwerke (seit 2010, www.wissenschaftscampus-tuebingen.de) In Kooperation mit Tübinger Medienwissenschaftlern und Psychologen, gefördert im Rahmen des Wissenschaftscampus Tübingen, untersuchen wir die Konstruktion domänen-spezifischer Sprachen (DSLs) für die deklarative Exploration sehr großer sozialer Netzwerkdatenstrukturen, die mittels relationaler Datenbanksysteme repräsentiert sind. Tief eingebettet in die funktionale Programmiersprache *Haskell* wird Domänen-Experten ein erweiterbares Vokabular vertrauter Objekte und Verben an die Hand gegeben, mit dem komplexe Analysen graphförmiger Strukturen sozialer und medialer Interaktion beschrieben werden können. Eine direkte Ausführung dieser Analyseprogramme würde zu iterativen (ineffizienten) Datenbankzugriffsmustern führen. Auch hier setzen wir datenbankspezifische Compilationstechniken ein, die derartige Iteration in daten-parallele, effiziente Anfragen transformiert.

Literatur

1. Boncz P, Grust T, van Keulen M, Manegold S, Rittinger J, Teubner J (2006) MonetDB/XQuery: a fast XQuery processor powered by a relational engine. In: Proc. SIGMOD, Chicago, USA, pp 479–490
2. Grust T (2002) Accelerating XPath location steps. In: Proc. SIGMOD, Madison, USA, pp 109–120
3. Grust T, Sakr S, Teubner J (2004) XQuery on SQL hosts. In: Proc. VLDB, Toronto, Canada, pp 252–263
4. Grust T, Mayr M, Rittinger J, Schreiber T (2009) Ferry: database-supported program execution. In: Proc SIGMOD, Providence, RI, USA, pp. 1063–1066
5. Herschel M, Hernández MA, Tan WC (2009) Artemis: a system for analyzing missing answers. PVLDB 2(2):1550–1553