INSTITUT FÜR INFORMATIK
Lehr- und Forschungseinheit für
Programmier- und Modellierungssprachen
Oettingenstraße 67, D–80538 München

Ludwig
Maximilians-
Universität
München

Deduktive Datenbanken

François Bry, Rainer Manthey, Heribert Schütz

erscheint in *KI – Künstliche Intelligenz – Forschung, Entwicklung, Erfahrungen*
http://www.pms.informatik.uni-muenchen.de/publikationen
Forschungsbericht/Research Report PMS-FB-1995-1, September 1995

Deduktive Datenbanken

François Bry[1], Rainer Manthey[2], Heribert Schütz[1]

[1]Institut für Informatik der Universität München,
{bry,hschuetz}@informatik.uni-muenchen.de
[2]Institut für Informatik III, Universität Bonn,
manthey@informatik.uni-bonn.de

1 Einleitung

Deduktive Datenbanken stehen in einer interessanten Beziehung zur Logikprogrammierung. Einerseits kann die Fachrichtung "deduktive Datenbanken" als Teil der Logikprogrammierung angesehen werden. Sie ist zusammen mit und fast gleichzeitig wie die Logikprogrammierung entstanden [Min88b,GMN84]. Wie die Logikprogrammierung beruhen die deduktiven Datenbanken auf der Feststellung, daß eine Sprache der Logik erster Stufe (üblicherweise basierend auf definiten Klauseln) als Modellierungssprache verwendet werden kann, wobei die Modelltheorie die Semantik und ein geeigneter Beweiskalkül das Berechnungsmodell liefert. Dementsprechend gelten die theoretischen Grundlagen der Logikprogrammierung auch für deduktive Datenbanken [Min88a]. Andererseits unterscheiden sich deduktive Datenbanken von der eigentlichen Logikprogrammierung in ihren Zielen und Methoden. Dieser Artikel soll in diese besonderen Ziele und Methoden einführen. Er soll insbesondere einen Überblick über die Deduktionsmethoden geben, die während des letzten Jahrzehnts zur Anfrageauswertung, Integritäts- und Änderungsverwaltung in deduktiven Datenbanken entwickelt worden sind. Aus Platzgründen werden nur wenige Literaturhinweise gegeben, die sich im wesentlichen auf Bücher und Übersichtsartikel beschränken, die weitere Referenzen auf spezifische Forschungsliteratur enthalten.

2 Ziele der Forschung über deduktive Datenbanken

Am besten läßt sich die "Philosophie" der deduktiven Datenbanken in Hinblick auf die geschichtliche Entwicklung der Datenbanksysteme erörtern. Entstanden aus dem Wunsch, die Verwaltung von großen oder persistenten Datenmengen zu erleichtern und von den Anwendungsprogrammen zu trennen, hat die Datenbankforschung von Anfang an zwei komplementäre Ziele verfolgt.

Zum einen soll ein Datenbankverwaltungssystem Dienstleistungen im Bereich der Systemverwaltung anbieten, die – andere Akzente setzend – der Prozeß- und Netzverwaltung verwandt sind: Dateiverwaltung, Sicherung der Daten gegen unerlaubte Zugriffe und Systemfehler, nebenläufige Datenzugriffe und Transaktionsverwaltung.

Zum anderen soll ein Datenbankverwaltungssystem unter Einbeziehung von höheren Sprachen die Datenmodellierung, d.h. die Repräsentation von Datenbankanwendungen, erleichtern. Ausgehend von Dateien sind zu diesem Zweck hierarchische, Netzwerk-, relationale, Entity-Relationship- und zuletzt deduktive, aktive und objektorientierte Datenbankmodelle vorgeschlagen worden.

Die eine Strömung der Datenbankforschung ist inhaltlich mit der systemnahen Informatik verwandt. Die andere Strömung liegt nahe bei den Programmier- und Modellierungssprachen. Ein grundlegendes Prinzip verbindet die beiden komplementären Schwerpunkte der Datenbankforschung: Eine Datenbank soll unabhängig von ihrer internen Darstellung einem Anwendungsprogramm gegenüber so erscheinen, als ob sie ausschließlich diesem Programm zur Verfügung stünde. Dies wird einerseits durch die Isolation von Transaktionen, andererseits durch spezialisierte externe Datenbank-Schemata erreicht.

Die Fachrichtung "deduktive Datenbanken" gehört im wesentlichen zu dem Teil der Datenbankforschung, der den Programmier- und Modellierungssprachen näher steht. Aus der Sicht der Datenbanken wird sie meist als Erweiterung der relationalen Datenbanken angesehen, weil eine relationale Datenbank hinsichtlich Datenmodellierung und Semantik ein ausschließlich aus **Fakten** bestehendes Logikprogramm darstellt. Eine deduktive Datenbank entspricht demgegenüber einem mit sogenannten **Integritätsbedingungen** ergänzten Logikprogramm. Fakten bzw. nichtatomare Klauseln, **Ableitungsregeln** genannt, sind extensionale bzw. intensionale Definitionen, die zur Anfrageauswertung abgerufen werden. Integritätsbedingungen dienen zur Überwachung von Änderungen der Fakten und Ableitungsregeln: nur solche Änderungen werden zugelassen, die zu Fakten- und Regelmengen führen, die die Integritätsbedingungen erfüllen. Inzwischen werden auch deduktive Erweiterungen anderer, insbesondere objektorientierter Datenmodelle untersucht und implementiert, deren Zusammenhang mit der "klassischen" logischen Programmierung nicht mehr so offensichtlich ist wie im relationalen Fall.

Für die meisten Datenbanken sind intensionale Spezifikationen und Integritätsbedingungen äußerst wichtig. Sie werden derzeit vorwiegend durch spezielle Anwendungsprogramme implementiert, die für jede Anwendung gesondert entwickelt werden müssen. Dabei zeigen sich folgende Nachteile, die von deduktiven Datenbankverwaltungssystemen vermieden werden sollen:

- unsystematische Programmentwicklung
- stark von der Programmierung abhängende Effizienz
- Verwaltung der Programme außerhalb der Datenbank
- imperative statt deklarativer Spezifikationen.

Intensionale Spezifikationen und Integritätsbedingungen sind Bestandteile der Datenbankanwendung und sollen wie die sonstigen Daten gesichert werden sowie von nebenläufigen Transaktionen zugegriffen werden können, d.h. sie sollen als Teil der Datenbank verwaltet werden. Ein ideales deduktives Datenbankverwaltungssystem würde intensionale Regeln und Integritätsbedingungen gleichermaßen wie Fakten verwalten, d.h. für deren Speicherung, Abrufe und Änderungen ähnliche Dienste anbieten wie für die sonstigen Daten. Die Forschungsergebnisse des letzten Jahrzehnts lassen dieses Ziel näher rücken.

Aus der Sicht relationaler Datenbanken ist die Erweiterung einer Datenbank um Ableitungsregeln und Integritätsbedingungen aus mehreren Gründen natürlich. Zum einen stellen Ableitungsregeln lediglich eine Verallgemeinerung des im relationalen Modell vorhandenen Begriffes der "Sicht" (engl.: view) dar. Zum anderen war ein allgemeiner Ansatz für Integritätsbedingungen bereits ein Ziel der relationalen Datenbanken, das bisher nur teilweise realisiert worden ist.

Die in deduktiven Datenbanken eingeführten Erweiterungen stellen für die Datenbankforschung einen Sprung ins Unbekannte dar. Mit den deduktiven Datenbanken wird zum ersten Mal ein Datenbankkonzept vorgeschlagen, das genauso wie eine herkömmliche Programmiersprache uneingeschränkte Berechnungen ermöglicht, weil eine aus Fakten und Ableitungsregeln bestehende Sprache Turing-vollständig sein kann.

3 Das logische Modell deduktiver Datenbanken

Die deklarative Semantik deduktiver Datenbanken unterscheidet sich kaum von jener der Logikprogrammierung [GM92]. Die von einer deduktiven Datenbank spezifizierte Faktenmenge läßt sich sowohl als kleinstes Herbrand-Modell als auch als Fixpunkt eines Folgerungsoperators definieren. Dabei werden Integritätsbedingungen zunächst nicht berücksichtigt.

Eine deduktive Datenbank wird **widerspruchsfrei** genannt, wenn sie alle ihre Integritätsbedingungen erfüllt. Integritätsbedingungen können als geschlossene Formeln, d.h. Ja/Nein-Anfragen, definiert werden, die nach jeder Datenbankänderung von der Fakten- und Regelmenge erfüllt werden müssen. Ein wesentlicher Unterschied zwischen deduktiven Datenbanken und Logikprogrammen liegt darin, daß eine deduktive Datenbank wegen ihrer Integritätsbedingungen widersprüchlich oder gar unerfüllbar sein kann. In Abschnitt 5 wird erläutert, wie diesem Umstand Rechnung getragen werden kann.

Deduktive Datenbanken verlangen in der Regel von Fakten, Ableitungsregeln, Anfragen und Integritätsbedingungen **domänenunabhängig** zu sein. Eine Formel heißt domänenunabhängig, wenn sich ihre Gültigkeit bzgl. einer Interpretation durch eine Erweiterung der Domäne nicht ändert. Für Fakten bedeutet die Domänenunabhängigkeit lediglich, daß sie variablenfrei sind. Da die Domänenunabhängigkeit aber eine unentscheidbare Eigenschaft ist, sind hinreichende syntaktische Bedingungen, vor allem die Bereichsbeschränkung, vorgeschlagen worden. Eine Ableitungsregel heißt **bereichsbeschränkt**, wenn jede im Kopf oder in einem negativen Rumpfliteral vorkommende Variable ebenfalls in einem positiven Rumpfliteral vorkommt. Für Integritätsbedingungen und Anfragen läßt sich diese Eigenschaft geeignet verallgemeinern. In deduktiven Datenbanken wird die Bereichsbeschränkung in der Regel vorausgesetzt. Universelle Spezifikationen werden in deduktiven Datenbanken im Gegensatz zur eigentlichen Logikprogrammierung kaum berücksichtigt.

Der Nutzen rekursiver Spezifikationen für Datenbanken wurde zeitweise bezweifelt. Inzwischen ist die Bedeutung der **Rekursion** für die Modellierung von Datenbanken jedoch nicht mehr umstritten. Graphtraversierungen, wie sie in einer Vielzahl von Anwendungen (Energie- oder Kommunikationsnetze, Straßen- oder Bahnverbindungen, etc.) vorkommen, lassen sich deklarativ mittels Rekursion natürlich und elegant spezifizieren. Sogar die relationale Anfragesprache SQL wird derzeit um rekursive Sichten erweitert (SQL 3). Aus Effizienzgründen wird oft die Einschränkung auf lineare Rekursion vorgeschlagen. Obwohl diese Einschränkung meist möglich ist, kann sie bisweilen die Modellierung wesentlich erschweren.

Für deduktive Datenbanken wie in der Logikprogrammierung wird eine von der klassischen Logik abweichende Negationsart verwendet, die nichtmonoton ist und zur Modellierung der meisten Anwendungen besonders gut geeignet ist: Aussagen, die nicht bewiesen werden können, werden als falsch betrachtet ("**negation-as-failure**"). Die deklarative Semantik dieser Negationsart ist immer noch nicht abschließend geklärt. (Vgl. Abschnitt 2.1 des Beitrags von T. Eiter und G. Gottlob in diesem Heft.) Viele deduktive Datenbankverwaltungssysteme setzen **Stratifikation** – d.h. Abwesenheit von Negation in Rekursionszyklen – voraus, um sich auf Datenbanken mit einer klaren Semantik einzuschränken [Bid91].

Aus praktischen Gründen haben in deduktiven Datenbanken sogenannte **Aggregationsoperatoren** (z.B. arithmetischer Durchschnitt) eine größere Bedeutung als in der herkömmlichen Logikprogrammierung. In Verbindung mit rekursiven Regeln ermöglichen sie insbesondere einfache und anschauliche Spezifikationen von sogenannten Stücklisten ("bill of materials"), die von großer praktischer Relevanz sind. Auch bei der Aggregation treten Probleme des nichtmonotonen Schließens auf.

Der Mangel an Strukturierungsmöglichkeiten in der Logikprogrammierung erweist sich bei der Modellierung von deduktiven Datenbanken als besonders ungünstig. Um diese zu verbessern, aber auch um Konzepte wie Objektidentität oder Vererbung als Modellierungswerkzeuge zur Verfügung zu stellen, sind logische Sprachen vorgeschlagen worden, die einige Merkmale der Objektorientierung anbieten. Sie stellen eine interessante Erweiterung der herkömmlichen Logikprogrammierung dar, auch in theoretischer Hinsicht, da sie beispielsweise zyklische Terme oder Verwendung von Skolemtermen in Regelköpfen ermöglichen [KL89].

4 Anfrageauswertung

In ihrer prozeduralen Semantik weichen die deduktiven Datenbanken von der eigentlichen Logikprogrammierung deutlich ab. Die Tiefensuche, auf der üblicherweise Sprachen der Logikprogrammierung beruhen, wird für deduktive Datenbanken aus zwei Gründen in Frage gestellt: zum einen, weil sie die Terminierung der Anfrageauswertung nicht sicherstellt; zum anderen, weil sie zu einer fakten- statt mengenorientierten Anfrageauswertung führt.

Deduktive Datenbanken stellen insofern höhere Anforderungen an die Anfrageauswertung als die Logikprogrammierung, als Terminierung soweit möglich vom Auswertungsverfahren garantiert werden soll. Aus mehreren Gründen kann nicht von Benutzern einer deduktiven Datenbank verlangt werden, für Terminierung zu sorgen:

- Bei großen Fakten- und Regelmengen ist es kaum möglich, die Reihenfolge der Klauseln zu überschauen und so zu berücksichtigen, daß nur terminierende Anfragen gestellt werden.

- Aus Effizienzgründen ist es meist günstiger, im Sekundärspeicher Fakten- und Regelmengen ungeordnet zu verwalten.

Es wird also ein noch höheres Maß an Deklarativität gefordert. Für deduktive Datenbanken sind Anfrageauswertungsverfahren entwickelt worden, die diese besonderen Anforderungen erfüllen.

4.1 Die relationale Abstraktion: mengenorientierte Zugriffe auf Fakten

Für die Auswertung von Anfragen an relationale Datenbanken wurden die bekannten Operatoren der relationalen Algebra (Projektion, Selektion, Join usw.) eingeführt. Die relationale Algebra stellt die Relation bzw. Menge als abstrakten Datentyp zur Verfügung. Sie ermöglicht die Beschreibung von Anfragen unabhängig von den Datenstrukturen, durch die die betroffenen Relationen repräsentiert werden.

Ein Ausdruck der relationalen Algebra kann auf zwei Ebenen optimiert werden, insbesondere um die Zahl der Sekundärspeicher-Zugriffe zu verringern: Zunächst wird er durch einen algebraisch äquivalenten Ausdruck mit möglichst kleinen Zwischenergebnissen ersetzt. Danach werden für die einzelnen Operatoren geeignete Algorithmen und Datenstrukturen ausgewählt.

Solche Optimierungen wurden im Gebiet der relationalen Datenbanken besonders für positive, konjunktive Anfragen untersucht. Da an deduktive Datenbanken wegen ihrer Ableitungsregeln und Integritätsbedingungen oft allgemeinere Anfragen (einschließlich Negation, Disjunktion und Rekursion) gestellt werden, wurde die relationale Algebra um einige neue Operatoren und Optimierungsmethoden erweitert.

4.2 Vorwärtsschließende Anfrageauswertungsverfahren

In der Logikprogrammierung wird das Vorwärtsschließen fast ausschließlich theoretisch, zur Definition der Fixpunktsemantik verwendet. In deduktiven Datenbanken wird es hingegen auch praktisch, zur Implementierung der Anfrageauswertung verwendet [NR91].

Für Datenbanken, bei denen unterschiedliche Anfragen sehr häufig ausgewertet werden, kann es sinnvoll sein, die Menge aller ableitbaren Fakten zunächst zu generieren, um sie dann als relationale, nichtdeduktive Datenbank zu verwalten. Dieser Ansatz wird **Materialisierung** genannt. Zeitpläne für Verkehrsmittel, die unverhältnismäßig viel häufiger abgefragt als aktualisiert werden und sich hervorragend als deduktive Datenbank modellieren lassen, sind ein Beispiel für eine vorteilhafte Anwendung der Materialisierung.

Das Rückwärtsschließen erweist sich zur Materialisierung als völlig ungeeignet, weil es auf einer in diesem Kontext nutzlosen Verwaltung von Anfragen beruht. Das Vorwärtsschließen, das für die eigentliche Logikprogrammierung wegen seines Mangels an Selektivität ineffizient ist, erweist sich für die Materialisierung als vorteilhaft. Sind Fakten und Ableitungsregeln bereichsbeschränkt, so kann das Vorwärtsschließen als wiederholte Anwendung der positiven Unit-Hyperresolution implementiert werden, wobei keine blinde Instantiierung von Regeln erforderlich ist.

Die Faktengenerierung wird so lange fortgesetzt, bis keine neuen Fakten mehr abgeleitet werden können. In der Regel wird davon ausgegangen, daß in einer deduktiven Datenbank die Menge der ableitbaren Fakten endlich ist, was die Terminierung des Verfahrens sicherstellt. Ein Sättigungsverfahren, das der in der theoretischen Behandlung üblichen wiederholten Anwendung des mengenorientierten Folgerungsoperators unmittelbar entspricht, wird als "naives" Iterationsverfahren bezeichnet. Eine inkrementelle Variante dieses Verfahrens, das die systematische Wiederholung von Ableitungen vermeidet, wird "semi-naive", "differentielle" oder "**Δ-Iteration**" genannt [GKB87].

Diese Vorgehensweise läßt sich als Erweiterung eines relationalen Datenbankverwaltungssystems implementieren. Das Datenbankverwaltungssystem bietet Relationen zur Speicherung der abgeleiten Fakten und die (eventuell erweiterte) relationale Algebra zur mengenorientierten Implementierung der Unit-Hyperresolution. Seine Optimierungstechniken sorgen für eine effiziente Materialisierung. Die Möglichkeit, Anfragen durch reines Vorwärtsschließen gekoppelt mit anschließender Selektion der gesuchten Antworten zu implementieren, wird zwar in der Theorie immer wieder als "naive bzw. semi-naive Auswertung" erwähnt, stellt aber (außer in Extremfällen) keine praktikable Auswertungsmethode dar.

4.3 Rückwärtsschließende Anfrageauswertungsverfahren

Das Vorwärtsschließen ermöglicht kaum, die gestellte Anfrage zur Einschränkung der generierten Faktenmenge zu verwenden. Für Datenbankanfragen wie für Anfragen an Logikprogramme ist das Rückwärtsschließen unumgänglich, um den Suchraum einzuschränken.

Weil die Ableitungsregeln deduktiver Datenbanken die gleiche Gestalt haben wie die Klauseln eines Logikprogramms, läßt sich das Rückwärtsschließen für deduktive Datenbanken in gleicher Weise wie für Logikprogramme formalisieren, d.h. als SLD-Resolution. Wenn die SLD-Resolution den Suchraum bestimmt, setzt sie aber keine Suchstrategie voraus. Bei der Auswahl einer für die Anfrageauswertung in deduktiven Datenbanken geeigneten Suchstrategie werden andere Eigenschaften gewünscht als in der eigentlichen Logikprogrammierung. Ein Anfrageauswertungverfahren wird üblicherweise als geeignet für deduktive Datenbanken angesehen, wenn es mengenorientiert ist und in möglichst vielen Fällen terminiert.

Deduktive Datenbanken, in deren Fakten und Ableitungsregeln keine Negation und keine Funktionssymbole vorkommen, werden Datalog-Datenbanken genannt. (**Datalog** ist ein Kürzel für die Bezeichnung "database prolog", die in den 70er Jahren in der Logikprogrammierung verbreitet war. Datalog-Datenbanken sind eine unmittelbare Erweiterung der relationalen Datenbanken, in denen komplexe Terme ebenfalls nicht zulässig sind.) Das Herbrand-Universum und damit die Menge der ableitbaren Fakten einer solchen Datenbank ist endlich. Suchbäume der SLD-Resolution sind in diesem Fall dennoch nicht immer endlich, weil rekursive Ableitungsregeln zu unendlichen Ästen führen können. Daher sind rückwärtsschließende Anfrageauswertungsverfahren für funktionsfreie Datenbanken besonders eingehend untersucht worden.

Mit unterschiedlichen Formalismen und unter Verwendung verschiedener Datenstrukturen sind für Datalog-Datenbanken Methoden vorgeschlagen worden, denen allen das gleiche Prinzip zugrunde liegt: die **Memoisierung** von Anfragen und Antworten. Anfragen, die bereits einmal gestellt wurden, werden nicht noch einmal ausgewertet, und hergeleitete Antworten werden an alle Inkarnationen der zugehörigen Anfrage weitergegeben. Das Verfahren besteht somit aus einer verschränkten Materialisierung von Anfragen und Antworten, bis weder neue Anfragen noch neuen Antworten generiert werden können [War92]. (Vgl. Abschnitt 2.1 des Beitrags von M. Hess in diesem Heft.)

Eine umfassende Diskussion aller vorgeschlagenen Realisierungen dieses Prinzips würde den Rahmen dieses Überblicks sprengen. Zwei Ansätze seien hier jedoch erwähnt, zum einen weil mit ihnen das Prinzip erstmals eingeführt wurde, zum anderen weil sie das gestellte Problem auf sehr unterschiedliche Weise lösen. Die beiden Ansätze erschienen zunächst so unterschiedlich, daß erst nach einigen Jahren die Übereinstimmung des zugrunde liegenden Prinzips erkannt wurde [Bry90a].

Der eine Ansatz, die OLDT- oder SLDAL-Resolution [TS86,Vie89], ist eine Variante der SLD-Resolution und steht der Logikprogrammierung am nächsten. Der andere Ansatz, als Magic-Set-Methode bekannt [BMSU86,Ull88], steht den relationalen Datenbanken nahe, weil er als Datenstrukturen lediglich Relationen verwendet.

Die **SLDAL-Resolution** verwendet teilweise expandierte Beweisbäume als Datenstruktur. Es genügt im Gegensatz zur reinen Tiefensuche nicht, zu jedem Zeitpunkt nur einen einzigen Ast eines Beweisbaumes zu speichern. Die SLDAL-Resolution beruht also auf dem Ersatz der in der Logikprogrammierung üblichen Keller-Datenstruktur durch eine baumartige Datenstruktur. Terminierung der Auswertung wird bei diesem Ansatz durch eine Kombination von Subsumptionstests für Unteranfragen (admissibility test) und Memoisierung (lemma generation) erreicht, was die Endung AL motiviert.

Die **Magic-Set-Methode** beruht auf einer Transformation des gegebenen Datalog-Programms in ein anderes Datalog-Programm. Eine rein vorwärtsschließende Auswertung des transformierten Programms simuliert auch Schritte einer rückwärtsschließenden Auswertung des ursprünglichen Programms. Die partiellen Beweisbäume werden gewissermassen in einzelne Fakten zerlegt, die in Relationen gespeichert werden können.

Inzwischen ist der Ursprung der Magic-Set-Methode in Vergessenheit geraten. Sie entstand aus einer Formalisierung und Ausarbeitung der Alexander-Methode [RLK86]. Es stellte sich aber später heraus, daß die Supplementary-Magic-Set-Methode, eine Verfeinerung der Magic-Set-Methode, gerade der Alexander-Methode entspricht.

Die Magic-Set-Methode hat gegenüber der SLDAL-Resolution den Vorteil, daß der vorwärtsschließende Auswertungsmechanismus nicht verändert werden muß, weshalb die bekannten Optimierungstechniken aus relationalen Datenbanken weiter verwendet werden können. Aus den eingeschränkten Datenstrukturen relationaler Datenbanken ergibt sich jedoch auch der wesentliche Nachteil der Magic-Set-Methode: Im Gegensatz zur SLDAL-Resolution kann der Zusammenhang zwischen den generierten Anfragen und Antworten nicht mehr durch die Datenstruktur repräsentiert werden, sondern muß durch möglicherweise teure Join-Operationen wiederhergestellt werden. Außerdem kann die aus Effizienzgründen unumgängliche Endrekursions-Optimierung in die SLDAL-Resolution wesentlich leichter eingebaut werden als in die Magic-Set-Methode.

Weitere terminierende Anfrageauswertungsmethoden sind für Datalog-Datenbanken entwickelt worden, in denen ausschließlich lineare Rekursionsmuster vorkommen. Die praktische Begründung für diese Forschung war die Feststellung – oder Überzeugung –, daß die meisten Anwendungen sich mit linearrekursiven Ableitungsregeln spezifizieren lassen. Zweifelsohne sind diese Methoden nicht nur für deduktive Datenbanken von Belang. Es ist zu erwarten, daß einige dieser Methoden von anderen Bereichen der Logikprogrammierung übernommen werden.

Weitere Methoden wurden und werden noch entwickelt, um Verfahren wie die SLDAL-Resolution und die Magic-Set-Methode um die Behandlung der Negation und von Aggregaten zu erweitern. Auch Heuristiken und Strategien für diese Verfahren sind untersucht worden. Das Rückwärtsschließen in deduktiven Datenbanken bleibt ein aktives Forschungsfeld, auch wenn es wegen der oben erwähnten Besonderheit der Magic-Set-Methode oft als Vorwärtsschließen bezeichnet wird.

4.4 Faktenweise Anfrageauswertung

Eine mengenweise Anfrageauswertung liefert dem Anwendungsprogramm zu einem Zeitpunkt eine Menge von Antworten, welches dieses typischerweise nur faktenweise bearbeiten kann. Dieses Phänomen wird als "impedance mismatch" bezeichnet. Ein weiteres Problem der mengenweisen Auswertung ist, daß ganze Mengen von Zwischenergebnissen im (Sekundär-)Speicher abgelegt und im nächsten Verarbeitungsschritt wieder aufgenommen werden müssen.

Ähnlich wie in relationalen Datenbanken die mengenorientiert dargestellten Ausdrücke der relationalen Algebra tatsächlich vielfach tupelweise ausgewertet werden, können auch Anfragen an deduktive Datenbanken weitgehend faktenweise ausgewertet werden.

Einige deduktive Datenbanksysteme, insbesondere die Systeme Megalog (heute im Prolog-System Eclipse integriert) und Butterfly, belegen, daß durch eine faktenweise Anfrageauswertung eine beachtliche Effizienz erreicht werden kann. Bisher liegt jedoch kein systematischer Vergleich von mengen- und faktenweiser Auswertung vor. Eine Übersicht über diverse Verfahren findet man in [CGT90]. Das Lehrbuch [CGH94] stellt ebenfalls einen SLD-basierten, faktenorientierten Ansatz vor.

5 Integritätserhaltung

In Datenbanken wird zwischen statischen und dynamischen Integritätsbedingungen unterschieden. Statische Integritätsbedingungen können als Ja/Nein-Anfragen angesehen werden, die nach jeder Aktualisierung der Datenbank positiv beantwortet werden müssen. Dynamische Integritätsbedingungen sind Aussagen über die zeitliche Entwicklung einer Datenbank, meist über die beiden Zustände vor und nach einer Änderung. Die überwiegende Mehrzahl der bisher zu diesem Thema publizierten Arbeiten widmet sich statischen Integritätsbedingungen.

Wie gewöhnliche Anfragen können statische Integritätsbedingungen nicht behandelt werden. Zum einen soll eine Aktualisierung erst dann durchgeführt werden, wenn feststeht, daß sie zu keiner Verletzung von statischen Integritätsbedingungen führt. Das heißt, die Integritätsbedingungen sollen gegenüber einer Simulation der aktualisierten Datenbank ausgewertet werden, bevor die Datenbank tatsächlich geändert wird.

Zum anderen soll der Effizienz halber ausgenützt werden, daß die statischen Integritätsbedingungen vor der Aktualisierung erfüllt sind. Nur solche (Instanzen von) Integritätsbedingungen, die durch die Änderungen möglicherweise verletzt werden, sollen überprüft werden. Man spricht dann von einer inkrementellen Integritätsprüfung. Wegen der großen Faktenmenge, die aus einer deduktiven Datenbank ableitbar ist, erweisen sich nichtinkrementelle Integritätsprüfungsverfahren als ineffizient.

Schließlich muß der Fall widersprüchlicher Integritätsbedingungen behandelt werden. Sowohl Integritätsbedingungen als auch Ableitungsregeln können als logische Axiome angesehen werden und sind somit potentiell inhärent widersprüchlich bzw. unerfüllbar. In einer deduktiven Datenbank mit widersprüchlichen Integritätsbedingungen wird eine Integritätsprüfung nach einer versuchten Änderung stets fehlschlagen, d.h. es existieren in diesem Fall überhaupt keine konsistenten Datenbankzustände. Die Widerspruchsfreiheit oder Erfüllbarkeit der Integritätsbedingungen selbst kann nicht durch eine Integritätsprüfungsmethode festgestellt werden. Integritäts- und Erfüllbarkeitsprüfung sind zwei komplementäre Aspekte der Wissensakquisition in Datenbanken.

5.1 Integritätsprüfung

Ableitungsregeln sind ein besonders geeignetes Mittel zur Simulation einer aktualisierten Datenbank. Dadurch besteht ein enger Zusammenhang zwischen Integritätsprüfungen und deduktiven Datenbanken. Dies mag der Grund dafür sein, daß Integritätsüberpüfungsmethoden, die auch für nichtdeduktive Datenbanken von großer praktischer Relevanz sind, fast ausschließlich im Gebiet der deduktiven Datenbanken untersucht worden sind.

Ein kurzes Logikprogramm – ein Metaprogramm – reicht aus, um die geänderte deduktive Datenbank zu simulieren. Die in deduktiven Datenbanken üblichen Auswertungs- und Optimierungsmethoden können auf dieses angewandt werden, um die Effizienz der Integritätsprüfung zu gewährleisten. Während die theoretische Formalisierung dieses auf Metaprogrammierung beruhenden Ansatzes noch interessante Fragen offen läßt, hat sich der Ansatz in der Praxis bewährt.

Durch partielle Auswertung des Metaprogramms bezüglich der statischen Integritätsbedingungen und eines Änderungsmusters werden spezialisierte Integritätsbedingungen gewonnen, die als von der Fakten- bzw. Regelmenge unabhängige minimale Zulässigkeitsbedingungen für die Änderung angesehen werden können. In dieser Weise lassen sich häufig vorkommende Änderungsmuster "vorkompilieren". Die so erzeugten Bedingungen sind typischerweise dynamische Integritätsbedingungen. Eine Übersicht zu dieser Thematik bietet [BMM91].

5.2 Erfüllbarkeitsprüfung

Die Überprüfung von Integritätsbedingungen auf **Erfüllbarkeit** ist sehr wenig untersucht worden. Sie ist jedoch insbesondere im Zusammenhang mit der Logikprogrammierung erwähnenswert, weil sie Anlaß zur Entwicklung des in Prolog implementierten Theorembeweisers SATCHMO war [MB88].

Die Erfüllbarkeit einer Klauselmenge läßt sich am besten dadurch überprüfen, daß Herbrand-Modelle für die Klauselmenge systematisch gesucht werden. Basierend auf dieser Feststellung wurde ein neuartiges, tableaux-ähnliches (jedoch wesentlich effizienteres) Modellgenerierungsverfahren entwickelt, das einige Techniken der Logikprogrammierung und deduktiver Datenbanken verwendet. Das Verfahren ließ sich in einem effizienten und äußerst kurzen Prolog-Programm implementieren.

Der Erfolg des Beweisers SATCHMO, der vielseitig angewandt und weiterentwickelt wurde, hat die ursprüngliche Motivation für seine Entwicklung, die Erfüllbarkeitsprüfung für Integritätsbedingungen in Datenbanken, in den Hintergrund gedrängt. Dies ist bedauerlich, weil SATCHMO und das mit ihm eingeführte Modellgenerierungsverfahren noch in keiner Weise eine völlig ausreichende Lösung zur Erfüllbarkeitsprüfung darstellt. Oft verlangen eben Datenbankanwendungen, daß die statischen Integritätsbedingungen zusammen mit den Ableitungsregeln ein endliches Modell haben, d.h. erfüllbar im Endlichen sind. Erste Ergebnisse zeigen, daß sich SATCHMO zur Überprüfung der Erfüllbarkeit im Endlichen gut erweitern läßt.

6 Datenbankänderungen

Die Spezifikation und Bearbeitung von Änderungen deduktiver Datenbanken wird seit einiger Zeit intensiv untersucht. Datenbankänderungen betreffen selten einzelne Fakten, sondern vielmehr Faktenmengen, die deklarativ spezifiziert werden, was zur Entwicklung von regelbasierten Sprachen zur Spezifikation von Datenbankänderungen geführt hat. Mehrere Ansätze sind vorgeschlagen worden, die hier in deklarative und imperative Ansätze eingeteilt werden. In diesem Zusammenhang sei noch einmal darauf hingewiesen, daß auch statische und dynamische Integritätsbedingungen, die zur Spezifikation von Änderungen beitragen, durch Regeln dargestellt werden können. Der Materialisierungsansatz (s. Abschnitt 4.2) verlangt besondere Änderungsmethoden. Schließlich können Änderungen explizit gespeicherte oder auch aus der Datenbank ableitbare Fakten betreffen. Diese zweite Art von Datenbankänderungen wurde bereits für relationale Datenbanken untersucht, und wird "view update" genannt.

6.1 Deklarative, regelbasierte Ansätze für Datenbankänderungen

Datenbankänderungen, die eine durch eine Anfrage spezifizierte Faktenmenge betreffen, lassen sich anschaulich und natürlich mittels Regeln der Gestalt

Änderung $\leftarrow$ Anfrage

spezifizieren. Änderungen, die mittels einer solchen Regelsprache definiert sind, lassen sich in verschiedener Weise durchführen.

Eine Möglichkeit der Interpretation solcher Regeln besteht darin, daß zunächst alle Regelrümpfe ausgewertet und danach die den abgeleiteten Regelköpfen entsprechenden Änderungen mengenweise in einer Transaktion durchgeführt werden, sofern sie einander nicht widersprechen.

Regelbasierte Ansätze für Datenbankänderungen, die in dieser Weise bearbeitet werden, nennen wir deklarativ, weil die Spezifikation der Änderung keine imperativen Aspekte enthält (während natürlich die Durchführung der Änderung imperativ verstanden werden muß).

Deklarative Ansätze haben den Vorteil, gut mit Integritätsprüfungsmethoden kombinierbar zu sein. Sie haben den Nachteil, zur Beschreibung einiger komplexer Änderungen wenig geeignet zu sein.

6.2 Imperative, regelbasierte Ansätze für Datenbankänderungen

Änderungsregeln können statt mengenweise auch in irgendeiner für geeignet gehaltenen Reihenfolge nacheinander ausgewertet und durchgeführt werden. Dieser Ansatz entspricht dem der regelbasierten Produktionssysteme der künstlichen Intelligenz, die bekanntlich imperative Ansätze sind.

Wie in der künstlichen Intelligenz sind für deduktive sowie nichtdeduktive Datenbanken verschiedene Regelarten untersucht worden, im wesentlichen Variationen der bekannten Event-Condition-Action-Regeln oder ECA-Regeln. Datenbanken, für die solche Regeln spezifiziert werden können, werden **aktive Datenbanken** genannt.

Bei solchen Regelsprachen ist die Konfliktauflösung, d.h. die Bestimmung der Reihenfolge der Durchführung einer Regelinstanz, eine schwierige Frage, die in Datenbanken eine neue Dimension erhält. Die Semantik einer solchen Sprache kann unter Verwendung der formalen Methoden der Logikprogrammierung untersucht werden. Die Forschung zum Thema "Aktive Datenbanken" beschränkt sich heute allerdings nicht mehr nur auf die Verwendung von aktiven Regeln zur Änderungsspezifikation, sondern läßt eine Vielzahl von Ereignistypen zu. Auch mengenorientierte Semantiken werden inzwischen häufig verwendet [WC96].

6.3 Regelbasierte Darstellung von Integritätsbedingungen

Statische Integritätsbedingungen lassen sich natürlich und einfach als besondere Ableitungsregeln darstellen. Sei B eine Integritätsbedingung, so ermöglicht eine Regel der folgenden (oder einer entsprechenden normalisierten) Gestalt

$$\mathit{false} \leftarrow \neg B$$

die Ableitung von *false*, sobald die Integritätsbedingung B falsifiziert wird. Diese Darstellung hat einige Vorteile: Sie ermöglicht eine einheitliche Darstellung von Integritätsbedingungen und Ableitungsregeln und sie erleichtert die Einbeziehung von statischen Integritätsbedingungen in Regelsystemen zur Darstellung von Änderungen.

Der Zustand einer Datenbank nach einer Änderung läßt sich mit Hilfe von modalen Operatoren beschreiben, die durch ein Metaprogramm als Relationssymbole höherer Stufe implementiert werden können (s. Abschnitt 5.1). Dies ermöglicht auch eine einfache Darstellung dynamischer Integritätsbedingungen als Regeln.

6.4 Änderungen einer materialisierten Datenbank

Wird eine Datenbank nach dem Materialisierungsansatz verwaltet, sind nicht nur die explizit gefordeten Änderungen durchzuführen, sondern sie müssen auch über die Ableitungsregeln auf die materialisierten Fakten propagiert werden.

Schwierigkeiten treten insbesondere beim Entfernen von Fakten sowie in Gegenwart von nicht-monotonen Operationen (Negation und Aggregation) auf. Techniken von Truth-Maintenance-Systemen können hier Anwendung finden. Eine einfache Technik besteht darin, für jedes materialisierte Faktum über die Zahl seiner Herleitungen Buch zu führen. Erst wenn diese Zahl auf Null zurückgesetzt wird, kann das Faktum gelöscht werden.

6.5 Sichtenänderungen

Eine **Sichtenänderung** (engl.: view update) ist eine Spezifikation einer intendierten Änderung abgeleiteter Daten. Eine Änderung der explizit gespeicherten Fakten soll ermittelt werden, um ein gegebenes nichtableitbares (bzw. ableitbares) Faktum ableitbar (bzw. nichtableitbar) zu machen.

Das Inferenzprinzip zur Bestimmung solcher Realisierungsmöglichkeiten für Sichtenänderungen wird oft Abduktion genannt [Bry90b]. Die Bestimmung von "view updates" ist der Änderung einer generierten Datenbank ähnlich. Im Gegensatz zu den Änderungen einer generierten Datenbank hat sie jedoch im allgemeinen keine eindeutige Lösung. Welche dieser Möglichkeiten letztlich zu wählen ist, hängt von den tatsächlich vorliegenden Veränderungen in der Realität ab und kann nicht automatisiert werden. Imperative, regelbasierte Änderungssprachen erweisen sich als besonders geeignet für die Spezifikation von Sichtenänderungen.

7 Zukunft der deduktiven Datenbanken

Anfang der 80er Jahre haben sich einige Computer-Hersteller durch Forschungszentren wie MCC in Austin, Texas, oder ECRC in München, sehr aktiv an der Forschung im Bereich der deduktiven Datenbanken beteiligt. Die Gründung beider Zentren war eine Reaktion der amerikanischen und europäischen Industrie auf das japanische "Fifth Generation Computer Systems" Projekt und die Gründung des ICOT. Am ICOT selbst wurden ebenfalls deduktive Datenbanken untersucht.

In den Turbulenzen der Wirtschaftskrise zu Beginn der 90er Jahre hat das Interesse der Industrie an deduktiven Datenbanken deutlich nachgelassen. Ein aus der Forschung am ECRC hervorgegangenes industrielles Projekt wird jedoch beim französischen Unternehmen Bull weitergeführt. Dieses Vorhaben wird seit 1992 wissenschaftlich unterstützt durch ein europäisches Kooperationsprojekt (IDEA, ESPRIT-Projekt 6333). Ein Konsortium aus industriellen und akademischen Partnern untersucht im IDEA-Projekt eine deduktive Datenbanksprache, genannt CHIMERA, die objektorientierte und aktive Datenbank-Konzepte miteinbezieht, im Hinblick auf Methodik des Entwurfs, Implementierung und Anwendungen [CM94].

Während die industrielle Forschung und Entwicklung im Bereich der deduktiven Datenbanken etwas ins Stocken geraten ist, setzt sie sich in der akademischen Welt unvermindert fort. Theoretische Themen, wie etwa die Semantik der nichtmonotonen Negation, sowie praktischere Aspekte, wie die Anfrageoptimierung, werden weiter untersucht. Eine Übersicht über aktuelle Projekte zur Systementwicklung bieten [VLDB94,RU94].

Neue Datenbankthemen, wie z.B. Data Mining, Knowledge Discovery und Constraint-Datenbanken, scheinen von Methoden der Logikprogrammierung profitieren zu können. Bei den angegebenen Beispielen sind dies insbesondere die induktive Logikprogrammierung und die Constraint-Logikprogrammierung.

Dies könnte die Fachrichtung "deduktive Datenbanken" mit neuen Fragestellungen beleben.

Die Frage nach der Bedeutung deduktiver Datenbanken für die Datenbankforschung und -entwicklung stellt sich dennoch. Die Meinungen hierüber sind äußerst gespalten. Bei der Durchsetzung neuer Ideen und Techniken für Datenbanken spielen wirtschaftliche Gesichtspunkte eine noch entscheidendere Rolle als etwa bei Programmiersprachen, da die Implementation eines Datenbankverwaltungssystems hohe Anforderungen stellt und einen viel größeren Aufwand bedeutet als bei vielen anderen Software-Systemen.

Auch in der kommerziellen Entwicklung von Datenbankverwaltungssystemen setzen sich neue Ideen aus der Forschung bemerkenswert langsam durch. Die relationalen Datenbanken, die in den letzten Jahren allgemein anerkannt worden sind, galten noch zu Anfang der 80er Jahre unter Datenbankexperten als fragwürdige Technologie, obwohl sie bereits damals weitgehend erforscht waren.

Unserer Meinung nach gibt es keinen Grund daran zu zweifeln, daß sich viele, wenn nicht die meisten Methoden der deduktiven Datenbanken durchsetzen werden, auch wenn wir nicht damit rechnen, daß deduktive Datenbankverwaltungssysteme unter dieser Bezeichnung kommerziell angeboten werden. Im Gegensatz zu relationalen und objektorientierten Datenbanken haben deduktive und aktive Datenbanken nie den Anspruch erhoben, neue vollständige Datenbank-Paradigmen anzubieten, sondern sich stets auf die Untersuchung einer Regelform beschränkt, die mit sonstiger Datenbank-Technologie kombinierbar sein soll.

Literatur

[BMSU86] F. Bancilhon, D. Maier, Y. Sagiv und J. Ullman: "Magic sets and other strange ways to implement logic programs", in: Proc. 5th ACM Symp. on Principles of Database Systems (PODS), 1986, 1–15

[Bid91] N. Bidoit: "Negation in rule-based database languages: a survey", in: **Theoretical Computer Science**, Vol. 78: , 1991

[Bry90a] F. Bry: "Query evaluation in recursive databases: bottom-up and top-down reconciled", in: **Data and Knowledge Engineering**, Vol. 5, 1990, 289–312

[Bry90b] F. Bry: "Intensional Updates: Abduction via Deduction", in: Proc. 7th ICLP, 1990, 561–575

[BMM91] F. Bry, B. Martens und R. Manthey: "Integrity Verification in Knowledge Bases", in: Proc. 2nd Russian Logic Programming Conf., 1991, LNAI 592, 114–139

[CGT90] S. Ceri, G. Gottlob und L. Tanca: "**Logic Programming and Databases**", Springer, 1990

[CM94] S. Ceri und R. Manthey: "Chimera: A Model and Language for Active DOOD Systems", in: Proc. 2nd Intern. East-West Database Workshop, Klagenfurt, 1994, Springer, 1995

[CGH94] A.B. Cremers, U. Griefahn und R. Hinze: "**Deduktive Datenbanken – Eine Einführung aus der Sicht der Logischen Programmierung**", Vieweg, 1994

[GMN84] H. Gallaire, J. Minker, und J.-M. Nicolas: "Logic and Databases: A Deductive Approach", in: **ACM Computing Surveys**, Vol. 16 (2):153–185, 1984

[GM92] J. Grant und J. Minker: "The Impact of Logic Programming on Databases", in: **CACM**, Vol. 35 (3):66–81, 1992

[GKB87] U. Güntzer, W. Kiessling und R. Bayer: "On the Evaluation of Recursion in (Deductive) Database Systems by Efficient Differential Fixpoint Iteration", in: Proc. Intern. Conf. on Data Engineering, 1987, 120–129

[KL89] M. Kifer und G. Lausen: "F-logic: A Higher-Order Language for Reasoning about Objects, Inheritance and Scheme", in: Proc. ACM-SIGMOD Conf. 1989, 134–146

[MB88] R. Manthey und F. Bry: "SATCHMO: a theorem prover implemented in Prolog", in: Proc. 9th CADE, 1988, LNCS 310, 415–434

[Min88b] J. Minker: "Perspectives in Deductive Databases", in: **Journal of Logic Programming**, Vol. 5 (1):33–60, 1988

[Min88a] J. Minker (ed.): "**Foundations of Deductive Databases and Logic Programming**", Morgan-Kaufmann, 1988

[NR91] J. Naughton und R. Ramakrishnan: "Bottom-Up Evaluation of Logic Programs", in: J.L. Lassez and G. Plotkin (eds.): "**Computational Logic - Essays in Honor of Alan Robinson**", The MIT Press, 1991, 640–700

[RU94] R. Ramakrishnan und J. Ullman: "A survey of research on deductive database systems", in: **Journal of Logic Programming**, Vol. 15, 125–149, 1995

[VLDB94] K. Ramamohanarao und J. Harland (eds.): Special Issue on Prototypes of Deductive Database Systems, in: **The VLDB Journal**, Vol. 3 (2), 1994

[RLK86] J. Rohmer, R. Lescoeur und J.M. Kerisit: "The Alexander Method – A Technique for the Processing of Recursive Axioms in Deductive Database Queries", in: **New Generation Computing**, Vol. 4, 522–528, 1986

[TS86] H. Tamaki und T. Sato: "OLD Resolution with Tabulation", in: Proc. 3rd ICLP, 1986, 84–98, LNCS 225

[Ull88] J. Ullman: "**Principles of Database and Knowledge-Base Systems**", Computer Science Press, Rockville/USA, Vol. 1, 1988, Vol. 2, 1989

[Vie89] L. Vieille: "Database Complete Proof Procedures Based on SLD-Resolution", in: Proc. 4th ICLP, 1987, 74–103

[WC96] J. Widom und S. Ceri (eds.): "**Active Database Systems**", Morgan Kaufmann, 1996

[War92] D.S. Warren: "Memoing for Logic Programs", in: CACM, Vol. 35 (3):93–111, 1992