Eine Basis für effiziente Konsistenzprüfung

Uwe Herzog *Guido Moerkotte*

Universität Karlsruhe, Fakultät für Informatik
D-76128 Karlsruhe, Germany
email: [herzog|moer]@ira.uka.de
1. Dezember 1994

Zusammenfassung

Im folgenden Beitrag wird eine Übersetzungstechnik für effiziente Konsistenztests in Datenbanken vorgestellt. Während die meisten Arbeiten auf dem Gebiet der Konsistenzprüfung eine Interpretation der vereinfachten Konsistenzbedingungen zur Laufzeit durchführen, wird hier eine Übersetzungstechnik verwendet, welche deklarativ beschriebene Konsistenzbedingungen in eine erweiterte relationale Algebra übersetzt.

Da für die Übersetzung der Konsistenzbedingungen in eine algebraische Darstellung zahlreiche Möglichkeiten existieren, ist eine Optimierung, beispielsweise durch Ausnutzung statistischen Wissens, möglich.

Ein weiterer Vorteil des Ansatzes ergibt sich aus den Erweiterungen der Algebra. Typische algebraische Operatoren haben zwei Eingabemengen und nur eine Ausgabemenge. Werden dagegen mehr als eine Ausgabemenge und mehr als zwei Eingabemengen zugelassen, verbessern sich die Kommunikations- und Ausdrucksfähigkeiten der Operatoren. Die Verwendung dieser als Bypass-Technik bezeichneten Methode ermöglicht einerseits schnelle Konsistenztests zur Laufzeit und bildet andererseits die Basis für eine weitere Effizienzsteigerung durch bessere grobkörnige Parallelisierung der resultierenden algebraischen Ausdrücke.

Anhand eines Kostenmodells und mittels Messungen auf einer Hauptspeicherdatenbank werden die Vorteile der Bypass-Technik gezeigt.

Inhaltsverzeichnis

1 Einleitung **4**

1.1 Motivation 4

1.2 Problemstellung 4

1.3 Lösungsansatz 5

1.4 Gliederung 7

2 Grundlagen **7**

2.1 Formeln 7

2.2 Bereichsbeschränkte Formeln 8

2.3 Datenbasis 9

3 Überblick über das Gesamtsystem zur Konsistenzprüfung **11**

4 Übersetzung in Algebra **13**

5 Basisregeln zur Übersetzung von Konsistenzbedingungen **15**

5.1 Operatoren der relationalen Algebra 15

5.2 Verallgemeinerung von Operatoren der relationalen Algebra 15

5.3 Übersetzungsregeln (Basisregeln) 18

6 Spezialregeln zur Übersetzung von Konsistenzbedingungen **22**

6.1 Die Bypass-Technik 22

6.2 Operatoren der erweiterten relationalen Algebra 23

6.3 Übersetzungsregeln (Spezialregeln) 30

6.3.1 Regeln für existenzquantifizierte Formeln 30

6.3.2 Regeln für allquantifizierte Formeln 32

6.4 Vermeidung von Divisionen 35

7 Auswertung der Bypass-Technik **36**

7.1 Beispiele 36

7.2 Auswertung mittels Kostenmodell 38

7.2.1 Kostenmodell 38

7.2.2 Kostenbetrachtung anhand von Beispielen 39
7.2.3 Allgemeinere Kostenabschätzungen 41
7.3 Auswertung durch Messung am System 42

8 Zusammenfassung **45**

1 Einleitung

1.1 Motivation

Zentraler Bestandteil heutiger Informationssysteme sind meist umfangreiche Datenbasen. Es wird erwartet, daß Datenbasen stets ein korrektes und aktuelles Abbild des modellierten Ausschnittes der realen Welt sind. Um dieser Forderung nachzukommen, werden Bedingungen festgelegt, die gültige Zustände einer Datenbasis beschreiben. Folgt die Datenbasis diesen Bedingungen, so wird sie *konsistent* genannt. Wenn diese Bedingungen im Datenbanksystem abgelegt sind, besteht für das Datenbankverwaltungssystem die Möglichkeit, anhand der vom Anwender *deklarativ* spezifizierten Konsistenzbedingungen eine Übereinstimmung der Datenbasis mit der realen Welt bezüglich der Bedingungen zu garantieren. Wird dies auch realisiert, liegt der Vorteil der Konsistenzsicherung durch das Datenbanksystem gegenüber der Konsistenzsicherung in jeder Anwendung in der höheren Sicherheit der Daten, da die Datenbasis eine globale Ressource ist. Weitere wesentliche Vorteile ergeben sich durch die Vereinfachung und bessere Wartbarkeit der Anwendungen sowie durch eine größere Flexibilität bei der Änderung von Konsistenzbedingungen.

Hier werden Anwendungen betrachtet, bei denen der klassische Transaktionsbegriff zugrundegelegt wird, der eine Transaktion als eine konsistenzerhaltende Einheit betrachtet, bei der ein konsistenter Datenbasiszustand wieder in einen konsistenten Datenbasiszustand überführt wird.

1.2 Problemstellung

Um konsistente Datenbasen zu gewährleisten, muß nach jeder Transaktion, die eine Änderung der Datenbasis nach sich zieht, eine Überprüfung der Konsistenzbedingungen durch das Datenbanksystem erfolgen. Die Konsistenzprüfungen in Datenbanken sind sehr aufwendig, da

- (statische) Konsistenzbedingungen globale Aussagen über den Zustand der gesamten Datenbasis sind,
- Datenbasen oft sehr groß sind,
- Konsistenzbedingungen komplexe Formeln und damit aufwendig auszuwerten sind und
- die Menge der Konsistenzbedingungen und damit die Menge der potentiell durchzuführenden Konsistenztests sehr groß ist.

Dies kann schon bei kleinen Datenbasen zu einem Zeitaufwand führen, der ein Mehrfaches der sonstigen Ausführungszeit für die Transaktion beträgt [GA93]. Für praktische Anwendungen ist das unakzeptabel. Wichtig ist daher die *Optimierung* der Konsistenzbedingungen. In bisherigen Arbeiten zur Konsistenzprüfung in Datenbanksystemen ist diesem Gebiet jedoch nahezu keine Beachtung geschenkt worden [GA93].

Die Forschungsarbeiten auf dem Gebiet der Konsistenzprüfung beschäftigen sich bislang fast ausschließlich mit der Vereinfachung der zu testenden Konsistenzbedingungen und der Reduzierung der zu testenden Datenmenge durch Vereinfachungsmethoden sowohl für relationale Datenbanken [Nic82, MH89, GA93], deduktive Datenbanken [LST87, KSS87, BDM88, MK88, Oli91, Küc91, UO92] temporale Datenbanken [Cho92, CN93, Ple93] und objektorientierte Datenbanken [JJ91, JQ92]. Die Vereinfachung der Konsistenzbedingungen erfolgt entweder zur Laufzeit, indem unter Ausnutzung der Daten aus der Transaktion vereinfachte Konsistenzbedingungen dynamisch generiert werden oder zur Definitionszeit [BDM88], indem sog. Differenzrelationen als Platzhalter für die Transaktionsdaten in die Konsistenzbedingungen eingefügt werden. Letztere Art der Vereinfachung wird als „compilation method" bezeichnet, sagt aber nichts über die Art der Auswertung der Konsistenzbedingung aus. Meist wird davon ausgegangen, daß ein geeigneter Auswertungsmechanismus vorliegt (Theorembeweiser), oder es wird direkt ein Codegenerator eingesetzt. Die Art der Auswertung ist in der Regel tupelorientiert.

Im Gegensatz zu den Testverfahren durch das Datenbankverwaltungssystem gibt es Methoden, welche die Transaktionen vor der Auswertung bezüglich der Konsistenzbedingungen modifizieren, so daß die Konsistenz der Datenbasis nicht verletzt wird [Sto75, Wal91]. Alternativ dazu ist die Verwendung von Regeln (ECA-Regeln, Trigger, Produktionsregeln) in aktiven Datenbanken [CW90, FPT92] ein Mittel zur Sicherstellung der Konsistenz von Datenbasen. Wenig Aufmerksamkeit wurde bisher der effizienten Auswertung von Konsistenzbedingungen [SV84] und deren Parallelisierung [GFA92] geschenkt.

Obwohl Leistungsaspekte als ein Schlüsselproblem bei der Verwendung von Konsistenzbedingungen in realen Anwendungen angesehen wird, gibt es bislang wenige Arbeiten zu diesem Problemkreis. Das Ziel unserer Arbeiten ist die Generierung von optimalen Konsistenztests für die Konsistenzprüfungskomponente des Datenbankverwaltungssystems aus deklarativ spezifizierten Konsistenzbedingungen, die vom Anwender im Laufe des Datenmodellierungsprozesses festgelegt wurden.

1.3 Lösungsansatz

Dabei sollen vorrangig Lösungen für die letzten drei der oben genannten Ursachen für die Komplexität von Konsistenzprüfungen gefunden werden. Das Lösungskonzept läßt sich folgendermaßen charakterisieren.

- Wegen der Größe der Datenbasis verwenden wir im Gegensatz zu bisher bekannten (tupelorientierten) Ansätzen eine mengenorientierte Verarbeitung.
- Zur Definitionszeit werden die deklarativ beschriebenen Konsistenzbedingungen in eine prozedurale Form übersetzt. Die Zielsprache ist wegen des zugrundegelegten relationalen Datenmodells die relationale Algebra. Die relationenalgebraischen Ausdrücke können durch die vorhandenen effizienten Auswertungsalgorithmen der Operatoren des DBMS ausgewertet werden (Übersetzeransatz).
- Aus der Spezifik von Konsistenzbedingungen werden neue Operatoren in die Algebra eingeführt, um noch effizienter auswertbare Ausdrücke zu erhalten; dabei wird

die *Bypass-Technik* [KMPS94] verwendet. Die vorhandenen Operatorimplementierungen können leicht an die Anforderungen der neuen Operatoren der erweiterten relationalen Algebra angepaßt werden.

- Analog der Anfrageoptimierung in Datenbanken [JK84] werden die Konsistenzbedingungen in erweiterter relationaler Algebra durch Einbeziehung statistischen Wissens optimiert. Die aufwendige Optimierung findet zur Definitionszeit der Konsistenzbedingungen statt.

- Der Test aller gleichzeitig von einer Transaktion betroffenen Konsistenzbedingungen kann durch Parallelisierung beschleunigt werden (Intra-Constraint-Parallelismus, Inter-Constraint-Parallelismus).

Die Überführung der Konsistenzbedingungen in eine Algebra als Zwischensprache und deren Verwaltung durch das Datenbanksystem bringt im Gegensatz zu Methoden, welche die Konsistenzprüfungskomponente als Aufsatz auf das Datenbanksystem betrachten [Wüt91, Kar94], zwei wesentliche Vorteile. Zum einen entfällt die Schnittstelle zwischen Konsistenzprüfungs- und Datenbanksystem, und zum anderen können Informationen und Komponenten des Datenbanksystems wie statistische Daten, Zugriffspfade oder Optimierungsmethoden ausgenutzt werden.

Die Übersetzung von deskriptiven Anfragen in eine Algebra ist seit der Einführung der relationalen Datenbanken [Cod70] eine Methode, die als Basis für Optimierungen bei der Anfrageauswertung genutzt wurde. Dabei wurden sowohl Transformationen für den Tupelkalkül [CG85] als auch für den Domänenkalkül [Mai83, AA93] in die relationale Algebra angegeben. Letztere Verfahren dienten jedoch als Beweis der Äquivalenz der Ausdrucksmächtigkeit beider Sprachen und sind nicht unter Effizienzgesichtpunkten entstanden. Insbesondere die Arbeiten von Dayal [Day83, Day87] und Bry [Bry89] stellen einen Ansatz zur optimierten Behandlung von Quantoren und Disjunktionen in Anfragen vor. Diese Arbeiten dienen als Ausgangspunkt für den hier vorgestellten Ansatz.

Obwohl Konsistenzbedingungen als Anfragen an die Datenbasis aufgefaßt und damit die vorhandenen Techniken zur Anfrageoptimierung und -auswertung genutzt werden können, bestehen zu Datenbankanfragen einige Unterschiede. Konsistenzbedingungen sind als statisch anzusehen und können deshalb immer zeitlich unabhängig von der Ausführung optimiert werden. [1] Wegen der häufigen Ausführung kommt der Optimierung von Konsistenzbedingungen zur Übersetzungszeit eine besondere Bedeutung zu. Konsistenzbedingungen sind in der Regel komplexere Formeln als Anfragen [LCW93]. Es treten Schachtelungen von Quantoren, Disjunktionen und häufig Negationen auf, und die Anzahl der Prädikate ist meist sehr groß.

Die Übersetzung in die erweiterte Algebra wird in Form von Übersetzungsregeln angegeben. Der Indeterminismus bei der Übersetzung erlaubt und erfordert eine nachfolgende Optimierung. Dabei können aus der Anfrageoptimierung bekannte Optimierungstechniken genutzt werden [Gra93]. Die erweiterte relationale Algebra wurde auch im Hinblick

[1] Es wird angenommen, daß die Informationen, die den Entscheidungen des Optimierers zugrundeliegt, weitestgehend stabil ist. Alternativ können Konzepte der dynamischen Optimierung zum Tragen kommen, die zur Laufzeit eine Auswahl aus einer Menge voroptimierter Konsistenztests treffen.

auf die Parallelisierung der Konsistenzprüfung entworfen. Dadurch soll ein zusätzlicher Effizienzgewinn resultieren.

Anwendungen für schnelle Konsistenztests

Neben klassischen Anwendungen von Konsistenzprüfungen zur Sicherstellung konsistenter Datenbasen in betrieblichen Informationssystemen (z.B. Rechnerintegrierte Entwicklung und Produktion, Bankwesen, Platzbuchungs- und Reservierungssysteme und Versicherungswirtschaft) gibt es neuere Anwendungsbereiche, die effiziente Konsistenztests erforderlich machen. Die automatische Generierung von Testdatenbasen zur Validierung konzeptueller Schemata im Datenbankentwurf ist eine solche Anwendung [NML93]. Auch bei der konsistenzbasierten Planung, welche auf einem Reparatursystem zur Behandlung von Konsistenzverletzungen in deduktiven Datenbanken [ML91] basiert, bilden effiziente Konsistenztests die Basis für schnelle Planungsverfahren [MM93]. Das hier vorgestellte Konzept zur effizienten Konsistenzprüfung wird für die flexible Verwaltung und Sicherung von Schemainformationen im objektorientierten Datenbanksystem GOM verwendet [MZ93].

1.4 Gliederung

Zunächst werden im folgenden Abschnitt 2 grundlegende Definitionen gegeben. Nach einem Überblick über das Gesamtsystem zur Konsistenzprüfung in Abschnitt 3 werden im vierten Abschnitt grundlegende Aspekte der regelbasierten Übersetzung von Konsistenzbedingungen in eine Algebra vorgestellt. Dabei werden im Abschnitt 5 die Basisregeln und im Abschnitt 6 Erweiterungen relationenalgebraischer Operatoren und Spezialregeln eingeführt. Anschließend erfolgt im Abschnitt 7 eine Bewertung der vorgestellten Übersetzungstechnik mittels eines Kostenmodells und durch Messungen auf einer Hauptspeicherdatenbank.

2 Grundlagen

In diesem Abschnitt wird die Notation von Konsistenzbedingungen vorgestellt. Es werden Konsistenzbedingungen betrachtet, die sich auf einen Datenbasiszustand beziehen. Diese werden als statische oder Zustandsbedingungen bezeichnet. Statische Konsistenzbedingungen sind geschlossene prädikatenlogische Formeln erster Ordnung, denen die (deduktive) Datenbasis genügen muß.

2.1 Formeln

Es werden die folgenden Symbolmengen benötigt: eine Menge V von Variablensymbolen, eine Menge C von Konstantensymbolen und eine Menge P von Prädikatsymbolen.

Variable werden durch $x, y, z, \ldots$ (möglicherweise indiziert), Konstante durch $a, b, c, \ldots$ und Prädikatsymbole durch $p, q, r, s, \ldots$ beschrieben. Prädikatsymbolen ist eine Stelligkeit n zugeordnet. Für ein Prädikatsymbol p mit der Stelligkeit n und den Konstanten $c_1, \ldots, c_n$ ist $p(c_1, \ldots, c_n)$ ein *Fakt*. Ein *Term* ist entweder eine Variable oder eine Konstante. Funktionssymbole in Prädikaten sind nicht zugelassen, um eine endliche Auswertung zu ermöglichen. Wenn p ein Prädikatsymbol der Stelligkeit n mit Termen $t_1, \ldots, t_n$ ist, dann ist $p(t_1, \ldots, t_n)$ eine *Atom* (eine atomare Formel). Ein *Literal* ist entweder ein Atom (l) oder seine Negation ($\neg l$). Ein *positives Literal* ist ein Atom. Ein *negatives Literal* ist die Negation eines Atoms.

Eine *Regel* hat die Form $l_1, \ldots, l_n \implies l_{n+1}$ für positive Literale l_i. Alle in einer Regel vorkommenden Variablen sind allquantifiziert.

Formeln werden wie üblich definiert: Jedes Literal ist eine Formel. Sind f_1 und f_2 Formeln, so sind auch $f_1 \wedge f_2$, $f_1 \vee f_2$, $f_1 \implies f_2$, und $\neg f_1$ Formeln. Ist x ein Variablensymbol, so sind auch $\forall x f_1$ und $\exists x f_1$ Formeln.

Wenn l ein positives Literal ist, tritt es positiv in l und negativ in $\neg l$ auf. Wenn es positiv (negativ) in f_1 auftritt, so tritt es auch positiv (negativ) in $f_1 \wedge f_2$, $f_1 \vee f_2$, und $f_2 \implies f_1$ auf. Wenn l positiv (negativ) in f_1 auftritt, so tritt es negativ (positiv) in $\neg f_1$ und $f_1 \implies f_2$ auf. Die Menge der freien Variablen einer Formel f wird mit $free(f)$ bezeichnet. Abkürzend für eine Formel der Form $\forall x_1 \ldots \forall x_n\ f'$ (analog $\exists x_1 \ldots \exists x_n\ f'$) schreiben wir auch $\forall x_1, \ldots, x_n\ f'$ ($\exists x_1, \ldots, x_n f'$). Wenn f' quantifiziert ist, so ist der erste Quantor $\exists$ ($\forall$) und die Formel wird als existenzquantifiziert (allquantifiziert) bezeichnet. Für eine Formel f wird mit *cnf(f)* (*dnf(f)*) deren pränexe konjunktive (disjunktive) Normalform beschrieben. Dies erleichtert die Beschreibung der Übersetzungsregeln in die Algebra ([Bry89] verwendet dagegen die sog. Miniscope-Form). Durch $\sim f$ wird die Formel beschrieben, welche aus $\neg f$ resultiert, wobei die Negation so weit wie möglich nach innen gezogen wird. Zu beachten ist, daß dabei die Quantoren invertiert werden, d.h. aus $\forall$ wird $\exists$ und umgekehrt.

2.2 Bereichsbeschränkte Formeln

Um die Auswertung einer Konsistenzbedingung nur aufgrund der Kenntnis der Datenbasis zu erlauben (und nicht die zugrundeliegende Domäne betrachten zu müssen), wird die syntaktisch zulässige Formelmenge eingeschränkt. Dabei wird die Definition der Bereichsbeschränktheit wie in [Nic82] verwendet.

Definition 2.1 (Bereichsbeschränktheit) *Eine geschlossene Formel f heißt bereichsbeschränkt gdw. es eine zu f äquivalente Formel g in Pränexer Konjunktiver Normalform gibt, für die gilt:*

- *Jede allquantifizierte Variable kommt in jeder Disjunktion, in der sie überhaupt vorkommt, auch in einem negativen Literal vor, und*
- *wenn eine existenzquantifizierte Variable x in einer Disjunktion negativ auftritt, dann gibt es eine Disjunktion von positiven Literalen, von denen jedes x enthält.*

Die jeweils geforderten Literale heißen Restriktionsliterale für die jeweilige Variable.

2.3 Datenbasis

Eine (deduktive) Datenbasis DB besteht aus einer Menge von Fakten DB^a, einer Menge von Regeln DB^d, und einer Menge von Konsistenzbedingungen DB^c, wobei die Regeln bereichsbeschränkt und Konsistenzbedingungen geschlossene bereichsbeschränkte Formeln sind.

Sei l ein positives Literal, dann ist:
$M(DB) := \{l | DB^a \cup DB^d \models l\}$ die Menge der aus der Datenbasis mittels Fakten und Regeln herleitbaren Literale und
$C(DB) := M(DB) \cup \{\neg l | DB^a \cup DB^d \not\models l\}$ die Vervollständigung der Datenbasis. "$C(DB) \models$" kann verkürzt geschrieben werden als "$DB \models$".

Definition 2.2 (Konsistenz) *Eine Datenbasis DB ist* konsistent*, genau dann wenn gilt: $DB \models c$ für alle $c \in DB^c$.*

Eine *Substitution* σ_X ist eine Abbildung einer Menge von Variablen X auf eine Menge von Konstanten. [2] Die Anwendung einer Substitution auf eine Formel ersetzt jedes Vorkommen einer freien Variablen $x \in X$ durch σ_X. Für eine Formel f ist die Anwendung von σ_X auf f beschrieben durch $f\sigma_X$. Alle Variablen, die nicht in X vorkommen, bleiben unberührt. Die leere Substitution ($\sigma_\emptyset$) wird durch ϵ bezeichnet.

Um zu betonen, daß f eine Formel mit freien Variablen $X := \{x_1, \ldots, x_n\}$ ist, kann $f(x_1, \ldots, x_n)$ geschrieben werden. Dies ist natürlich für Literale nicht notwendig. Der Ausdruck $f[x_1, \ldots, x_n]$ ist definiert durch $\{\sigma_X | DB \models f\sigma_X\}$. Wenn $X = \dot{X} \dot{\cup} \bar{X}$ ($\dot{\cup}$ bezeichnet die disjunktive Vereinigung) dann gilt $f[\dot{X}] := \{\sigma_{\dot{X}} | \exists \sigma_{\bar{X}}\ DB \models f\sigma_{\dot{X}}\sigma_{\bar{X}}\}$. Für ein Literal $l = p(x_1, \ldots, x_n)$ kann $p[x_1, \ldots, x_n]$ geschrieben werden anstatt $p(x_1, \ldots, x_n)[x_1, \ldots, x_n]$.

Berechenbare Prädikate

In Formeln sind Vergleichsprädikate als Sonderfall berechenbarer Prädikate zugelassen. Da sie nicht explizit in der Datenbasis vorhanden sein müssen, werden sie auch als Built-in-Prädikate bezeichnet. Zur Verfügung stehen alle zweistelligen Vergleichsprädikate $=, \neq, <, >, \leq, \geq$ mit einer offensichtlichen Semantik. Wegen der Endlichkeit der Auswertung wird gefordert, daß die Argumente der Prädikate beim Aufruf gebunden sind.

Äquivalenzen

Die Konsistenzbedingungen sind geschlossene, bereichsbeschränkte Formeln und liegen in pränexer Normalform vor, wobei Negationen soweit wie möglich nach innen gezogen wurden. Falls möglich, werden noch zwei logische Umformungen angewandt [Llo87]:

[2] Eine Substitution kann als Tupel einer Relation und eine Menge von Substitutionen als Relation angesehen werden.

Verschieben von Quantoren Allquantoren werden über die Konjunktionen gezogen, Existenzquantoren über Disjunktionen. Dies ist keine Miniscope-Form, da die Quantoren nur bis vor die Konjunktionen gezogen werden, nicht bis vor die Literale. Hierbei wird die pränexe Normalform aufgegeben.

$\exists\, x_1, \ldots, x_n f_1 \vee \ldots \vee f_n \implies \exists\, x_1, \ldots, x_n f_1 \vee \ldots \vee \exists\, x_1, \ldots, x_n f_n$

$\forall\, x_1, \ldots, x_n f_1 \wedge \ldots \wedge f_n \implies \forall\, x_1, \ldots, x_n f_1 \wedge \ldots \wedge \forall\, x_1, \ldots, x_n f_n$

Eliminierung überflüssiger Quantifizierungen Nutzlose Quantifizierungen (Quantoren, die Variablen quantifizieren, welche nicht in der Teilformel auftreten) werden gestrichen.

$\exists\, x_1, \ldots, x_n f \implies f$, falls kein x_i in f vorkommt.

$\forall\, x_1, \ldots, x_n f \implies f$, falls kein x_i in f vorkommt.

$\exists\, x_1, \ldots, x_n f \implies \exists\, x_{i_1} \ldots x_{i_p} f$, falls kein x_i verschieden von $x_{i_1} \ldots x_{i_p}$ in f vorkommt.

$\forall\, x_1, \ldots, x_n f \implies \forall\, x_{i_1} \ldots x_{i_p} f$, falls kein x_i verschieden von $x_{i_1} \ldots x_{i_p}$ in f vorkommt.

3 Überblick über das Gesamtsystem zur Konsistenzprüfung

In der folgenden Abbildung ist das Gesamtsystem zur Konsistenzprüfung dargestellt. Im wesentlichen können dabei zwei Phasen unterschieden werden. Die erste Phase umfaßt alle Verfahren, die einmalig zur Definitionszeit der Konsistenzbedingung durchgeführt werden. Die zweite Phase beinhaltet alle Verfahren, die während der Ausführung der Konsistenzprüfung zum Einsatz kommen. Folglich werden auch die zum Einsatz kommenden Optimierungsverfahren in diese zwei Phasen unterteilt.

Eine wesentliche Zielstellung des Ansatzes ist es, möglichst einen Großteil der aufwendigen Optimierungen (Suchverfahren) in der ersten Phase durchzuführen.

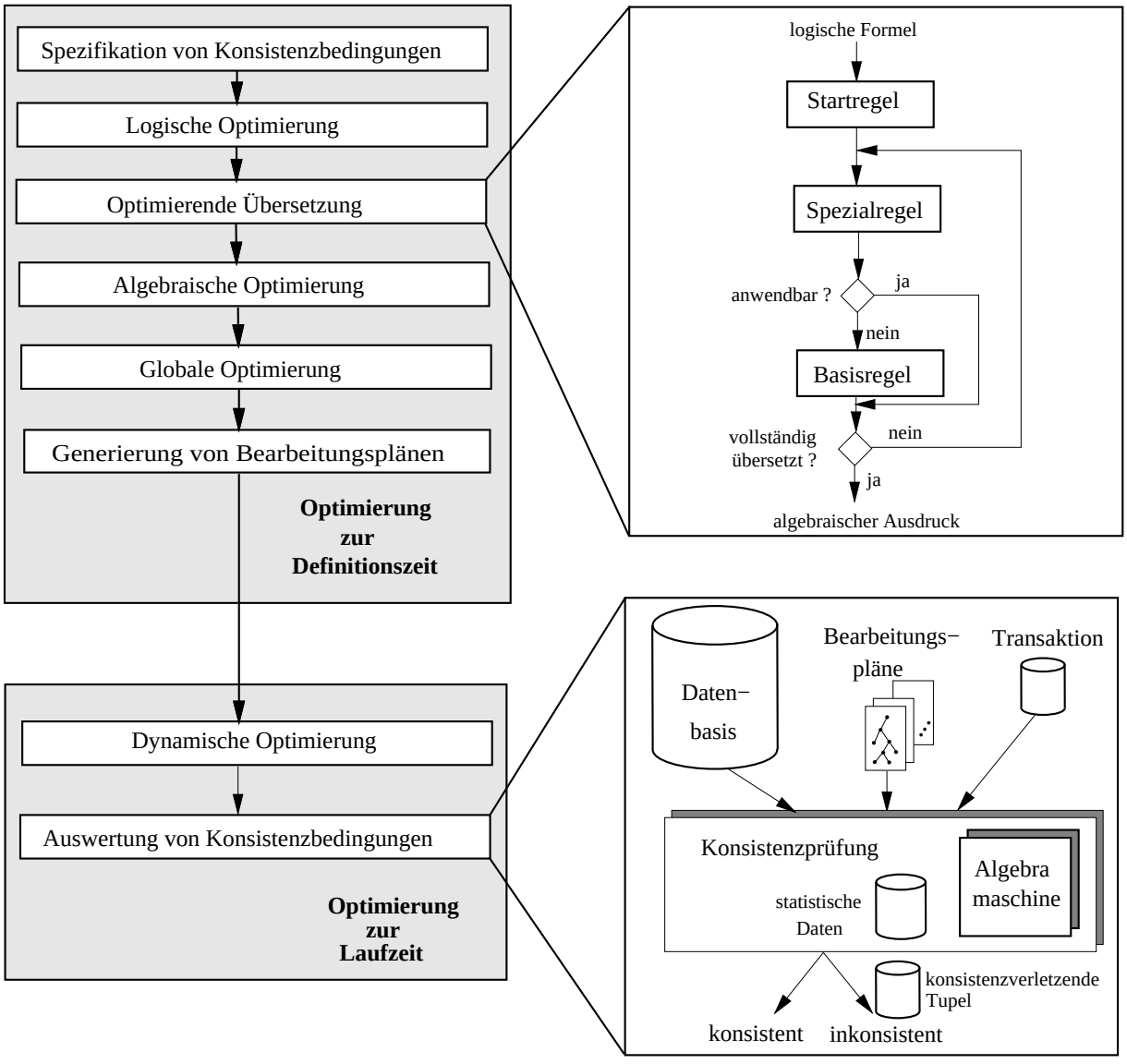


Abbildung 1: Gesamtsystem zur Konsistenzprüfung (Prinziplösung)

Im folgenden wird eine kurze Erläuterung der Teilphasen bei der Konsistenzprüfung gegeben. Im Bericht wird dann die Teilphase „Optimierende Übersetzung“ ausführlich erklärt.

Spezifikation von Konsistenzbedingungen Konsistenzbedingungen werden als bereichsbeschränkte, geschlossene, prädikatenlogische Formeln beschrieben. Als Prädikate sind neben Basisprädikaten (Relationen) auch abgeleitete Prädikate (Deduktionsregeln) und berechenbare Prädikate (Vergleichsoperatoren) zugelassen. Es wird von einer widerspruchs- und redundanzfreien Menge von Konsistenzbedingungen ausgegangen.

Optimierung auf logischer Ebene Unter der Annahme einer konsistenten Datenbasis zu Beginn der Transaktion werden vereinfachte Konsistenzbedingungen generiert [Nic82]. Dabei wird angenommen, daß die durch die Transaktion veränderten Tupel in sog. Differenzrelationen gespeichert sind, auf die während des Konsistenztests zurückgegriffen werden kann. Somit kann die Generierung und Optimierung vereinfachter Konsistenzbedingungen ohne die Kenntnis der Transaktion ausgeführt werden und ist so zur Definitionszeit möglich.

Optimierende Übersetzung Die deklarativ beschriebenen Konsistenzbedingungen werden in Ausdrücke der erweiterten relationalen Algebra übersetzt. Für spezielle Muster von Konsistenzbedingungen werden neue Operatoren in die Algebra eingeführt.

Optimierung auf algebraischer Ebene Analog zur Anfrageoptimierung wird für die algebraischen Ausdrücke die optimale Operatorreihenfolge bestimmt. Insbesondere spielt hier die Optimierung der Join-Reihenfolge eine große Rolle.

Globale Optimierung mehrerer Konsistenzbedingungen Da von einer Transaktion in der Regel mehrere Konsistenzbedingungen betroffen sind und gleichzeitig getestet werden müssen, wird eine gemeinsame Optimierung durch das Herausziehen von gemeinsamen Teilausdrücken durchgeführt.

Generierung von Bearbeitungsplänen Aus den optimierten Konsistenzbedingungen werden unter Ausnutzung der physischen Strukturen der Datenbank und der Operatorimplementierungen Bearbeitungspläne generiert.

Auswertung der Konsistenzbedingungen Für die Auswertung verschiedener Klassen von Konsistenzbedingungen werden spezielle Auswertungsstrategien eingesetzt. Von Interesse für die Auswertungsstrategie sind die Nutzeranforderungen, beispielsweise ob das Ergebnis der Konsistenzprüfung weiterverwendet werden soll. Entsprechend den aktuellen Informationen aus der Transaktion werden aus der Menge der voroptimierten Konsistenztests die optimalen selektiert und in der optimalen Reihenfolge ausgewertet. Für die aktuelle Überprüfung werden Ergebnisse aus vorangegangenen Konsistenztests ausgenutzt. Für die Auswertung von Konsistenzbedingungen mit abgeleiteten Prädikaten ist eine effiziente Bestimmung der Änderung der Datenbasis und eine effiziente Regelauswertungsstrategie erforderlich.

4 Übersetzung in Algebra

Im Rahmen dieses Berichts soll die optimierende Übersetzung der deklarativ spezifizierten Konsistenzbedingungen in eine prozedurale Form beschrieben werden. Die Übersetzung $\mathcal{E}$ überführt eine beliebige bereichsbeschränkte prädikatenlogische Formel in eine Folge von Operatoren der (erweiterten) relationalen Algebra. [3] Die Übersetzung erfolgt regelbasiert und ist indeterministisch, was Freiraum für Optimierungen läßt.

Im folgenden werden drei Aspekte der Konsistenzprüfung betrachtet, die Einfluß auf den Übersetzungsprozeß haben.

Auswertung: Konsistenzbedingungen werden wie Anfragen bzgl. einer Datenbasis ausgewertet. Dabei kann die Konsistenzbedingung entweder erfüllt oder verletzt sein. Diese binäre Entscheidung wird anhand des Ergebnisses der Auswertung getroffen. Das Ergebnis ist eine Menge von Tupeln. Um eine angemessene Reaktion auf eine mögliche Konsistenzverletzung zu ermöglichen, werden in der Ergebnisrelation die konistenzverletzenden Tupel gespeichert.
Das Ergebnis der Auswertung der Konsistenzbedingung wird in Abhängigkeit von der Quantifizierung der Formel unterschiedlich interpretiert:

Definition 4.1 (Auswertung Konsistenzbedingung) *Sei f eine Konsistenzbedingung in Prädikatenlogik 1. Stufe, $\mathcal{E}$ die Übersetzung von f in einen Ausdruck der erweiterten relationalen Algebra und* ***eval*** *die Auswertung dieses algebraischen Ausdruckes bzgl. einer gegebenen Datenbasis DB, deren Ergebnis eine Menge von Tupeln ist. Dann gilt:*

- *f ist eine $\forall$-quantifizierte Formel:* $DB \models f \Leftrightarrow eval(\mathcal{E}(\sim f), DB) = \emptyset$
- *f ist eine $\exists$-quantifizierte Formel:* $DB \models f \Leftrightarrow eval(\mathcal{E}(f), DB) \neq \emptyset$

Der Vorteil dieser Art der Auswertung ist die geringere Datenmenge, die als Ergebnis der Auswertung (eval) geliefert wird. Bei allquantifizierten Konsistenzbedingungen werden folglich nicht alle Tupel als Ergebnis zurückgegeben, welche die Konsistenzbedingung erfüllen, sondern diejenigen, welche sie verletzen (im Normalfall keines oder nur sehr wenige Tupel). Bei existenzquantifizierten Formeln besteht das Ergebnis aus allen Tupeln, welche die Konsistenzbedingung erfüllen, was in der Regel nur wenige sind. Ein weiterer Vorteil dieser Vorgehensweise ist die Tatsache, daß nun für die weiteren Verarbeitungsschritte in beiden Fällen existenzquantifizierte Formeln der Ausgangspunkt sind.

[3] Da die Operatoren der erweiterten relationalen Algebra nicht auf eine feste Anzahl von Eingängen und nur einen Ausgang beschränkt sind, ist es keine Algebra im strengen Sinne.

Ausgabemenge: Für vereinfachte Konsistenzbedingungen mit Differenzrelationen bzw. Δ-Relationen (Phase: Logische Optimierung) sind die Variablen der Δ-Relation die Ausgabevariablen der Konsistenzbedingung. Ansonsten werden die ersten n existenzquantifizierten Variablen als (freie) Ausgabevariablen benutzt. Damit ist die Voraussetzung für eine angemessene Reaktion auf das Ergebnis des Konsistenztests geschaffen. Es werden die algebraischen Ausdrücke vollständig ausgewertet, d.h. bei der ersten Konsistenzverletzung bzw. Konsistenzerfüllung wird nicht abgebrochen. Diese Strategie bildet die Basis für die Reaktionen auf Konsistenzverletzungen.

Ablauf der Übersetzung: Für die Übersetzung werden zwei Regelmengen angegeben. Die Menge der *Spezialregeln* erlaubt die direkte Übersetzung von häufig auftretenden Mustern von Konsistenzbedingungen in eine für die Auswertung effiziente Form. Die *Basisregeln* orientieren sich an den Verfahren der Anfrageübersetzung in relationalen Datenbanken.

Der prinzipielle Ablauf der Übersetzung ist in Abbildung 2 dargestellt. Die Übersetzung beginnt mit der Startregel. Falls die Bedingungen einer Spezialregel zutreffen, wird diese auf die Teilformel angewandt, ansonsten wird eine Basisregel verwendet. Die Transformation endet, wenn alle Teilformeln übersetzt sind. Wir betrachten bei den Spezialregeln Formeln als Einheit von Präfix und Matrix und übersetzen sie gleichzeitig im Gegensatz zu den Basisregeln, bei denen die Verarbeitung von Präfix und Matrix der Formel separat erfolgt.

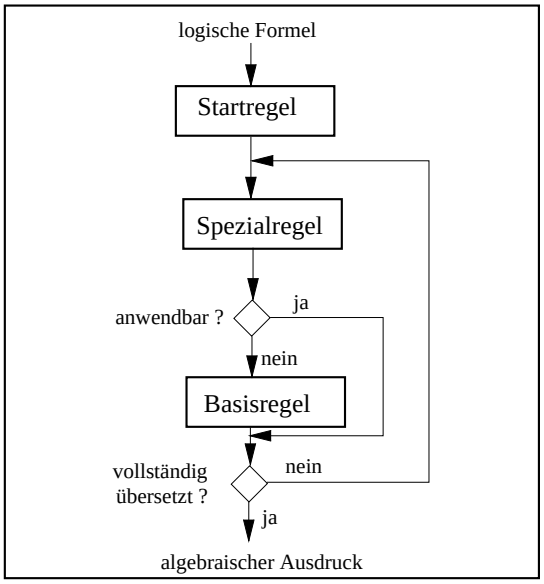


Abbildung 2: Übersetzungsprozeß

5 Basisregeln zur Übersetzung von Konsistenzbedingungen

5.1 Operatoren der relationalen Algebra

Die Übersetzung von Konsistenzbedingungen als logische Formeln in die relationale Algebra orientiert sich an den Arbeiten zur Abbildung von Anfragen im relationalen Domänenkalkül auf die relationale Algebra zum Beweis der gleichen Ausdrucksmächtigkeit beider Sprachen [Mai83, Ull88, AA93]. Speziell die Transformation von Anfragen mit Quantoren und Disjunktionen wird von Bry [Bry89] anhand einiger Beispiele behandelt, wobei keine vollständige Transformation angegeben wird.

Im Unterschied zu bisherigen Ansätzen wird die Tatsache, daß die zu übersetzenden Formeln bereichsbeschränkt sind, bei der Übersetzung ausgenutzt.

Die erweiterte relationale Algebra (XRA) basiert auf der relationalen Algebra [Mai83] mit den in Tabelle 1 dargestellten Operatoren.

Operator	Bezeichnung
π_A	Projektion auf die Attributmenge A
σ_p	Selektion mit Prädikat p
$\times$	Kartesisches Produkt
$\cup$	Vereinigung
$\cap$	Durchschnitt
$\Leftrightarrow$	Differenz
$\bowtie$	Natural Join
$\div$	Division

Tabelle 1: Operatoren der relationalen Algebra

5.2 Verallgemeinerung von Operatoren der relationalen Algebra

Um den speziellen Anforderungen von Konsistenzbedingungen und deren effizienter Auswertung Rechnung zu tragen, werden die bekannten Operatoren der relationalen Algebra erweitert.

Die Vereinigung wird durch eine weniger restriktive Definition ersetzt, welche die Vereinigung von Relationen unterschiedlicher Schemata zuläßt. Diese Definition wird als *Outer-Union* $\uplus$ [Cod79].bezeichnet.

Definition 5.1 (outer-union) *Seien* $R(x_1, \ldots, x_n)$ *und* $S(y_1, \ldots, y_m)$ *zwei Relationen mit den Attributmengen* $\mathcal{A}_R = \{x_1, \ldots, x_n\}$ *und* $\mathcal{A}_S = \{y_1, \ldots, y_m\}$. *Dann liefert der generalisierte Vereinigungsoperator* $R(x_1, \ldots, x_n) \uplus S(y_1, \ldots, y_m)$ *die Menge von Tupeln mit dem Schema* $\mathcal{A}_{res} = \{x_1, \ldots, x_n\} \cup \{y_1, \ldots, y_m\}$ *als Vereinigung der Relationen* $R(x_1, \ldots, x_n)$

und $S(y_1, \ldots, y_m)$, wobei fehlende Attributwerte durch NULL ergänzt werden. In der Arbeit wird weiterhin das Zeichen $\cup$ für den verallgemeinerten Vereinigungsoperator verwendet.

Erweiterte Join-Operatoren

Für die effiziente Auswertung der aus den Konsistenzbedingungen entstandenen algebraischen Ausdrücke sind zwei Dinge bei der Übersetzung zu berücksichtigen.

- In den als prädikatenlogische Formeln spezifizierten Konsistenzbedingungen treten häufig negierte Literale auf. Dafür sind effiziente algebraische Entsprechungen zu finden.
- Die Auswertung der Konsistenzbedingungen zum Nachweis der Konsistenz bzw. Inkonsistenz der Datenbasis sollte mit minimalem Kostenaufwand erfolgen. Insbesondere sollte die Tupelgröße so klein wie möglich gehalten werden.

Die im folgenden beschriebenen Verallgemeinerungen der Join-Operatoren sind die Folge der soeben angestellten Überlegungen. Die beiden Grundbausteine der erweiterten Join-Operatoren sind der bekannte *Natural-Join* ($\bowtie$) und der *Anti-Join* ($\triangleright$) [RGL90] [4] Die Operatoren Kreuzprodukt ($\times$) und Natural Join ($\bowtie$) werden von den verallgemeinerten Join-Operatoren subsumiert.

Der *Anti-Join* als Komplement des Semi-Joins zweier Relationen R und S enthält alle Tupel der Relation R, welche kein entsprechendes Tupel in der Relation S besitzen.

Definition 5.2 (ajoin) *Seien $R(x_1, \ldots, x_n)$ und $S(y_1, \ldots, y_m)$ zwei Relationen mit den Attributmengen $\{x_1, \ldots, x_n\}$ und $\{y_1, \ldots, y_m\}$. Dann ist der Anti-Join von R und S definiert durch:*

$$R(x_1, \ldots, x_n) \triangleright S(y_1, \ldots, y_m) = R(x_1, \ldots, x_n) \Leftrightarrow \pi_{\{x_1, \ldots, x_n\}} (R(x_1, \ldots, x_n) \bowtie S(y_1, \ldots, y_m)).$$

Der *Anti-Join*-Operator wird zur Übersetzung von negierten Literalen in Konjunktionen anstatt der Differenz (bei Attributgleichheit der Literale) oder der Komplementbildung des negierten Literals in anderen Verfahren verwendet, falls alle Variablen des Literals zum Auswertungszeitpunkt gebunden sind.

Beide Operatoren (Natural Join und Anti-Join) werden um eine Projektion erweitert, welche alle Attribute entfernt, die in der weiteren Verarbeitung nicht weiter benötigt werden. Weiterhin wird die Ausführung einer Selektion auf das Ergebnis des Operators erlaubt. Es werden somit implizit die Heuristiken der algebraischen Optimierung angewandt, wonach Projektionen und Selektionen so früh wie möglich ausgeführt werden sollen, um die Zwischenergebnismenge zu verringern.

Der nachfolgend definierte allgemeinere Join-Operator (**gjoin** - generalized join) entspricht einem *Natural Join* mit anschließender Projektion.

[4] Der Anti-Join wird auch als Komplement-Join [Bry89] bezeichnet.

Definition 5.3 (gjoin) *Seien $R(x_1, \ldots, x_n)$ und $S(y_1, \ldots, y_m)$ zwei Relationen mit den Attributmengen $\{x_1, \ldots, x_n\}$ und $\{y_1, \ldots, y_m\}$. Dann ist der verallgemeinerte Join-Operator (gjoin) definiert als:*

$$R(x_1, \ldots, x_n) \bowtie_{X,Y} S(y_1, \ldots, y_m) = \pi_{X,Y}(R(x_1, \ldots, x_n) \bowtie S(y_1, \ldots, y_m))$$

mit $X \subseteq \{x_1, \ldots, x_n\}$ und $Y \subseteq \{y_1, \ldots, y_m\}$.

So wie der gjoin-Operator eine Verallgemeinerung des Natural-Join darstellt, kann auch der Anti-Join verallgemeinert werden. Dieser neue Operator wird als komplementärer Join (**cjoin** - complement join) bezeichnet. Der cjoin-Operator liefert als Ergebnis alle Tupel der Relation R zurück, die kein Gegenstück in der Relation S besitzen; auch hier erweitert um eine Projektion.

Definition 5.4 (cjoin) *Seien $R(x_1, \ldots, x_n)$ und $S(y_1, \ldots, y_m)$ zwei Relationen mit den Attributmengen $\{x_1, \ldots, x_n\}$ und $\{y_1, \ldots, y_m\}$. Dann ist der komplementäre Join-Operator (cjoin) definiert als:*

$$R(x_1, \ldots, x_n) \overline{\bowtie}_{X,Y} S(y_1, \ldots, y_m) = \pi_{X,Y}(R(x_1, \ldots, x_n) \rhd S(y_1, \ldots, y_m))$$

mit $X \subseteq \{x_1, \ldots, x_n\}$ und $Y = \emptyset$.

Die Menge $X \cup Y$ sind jeweils alle Attribute, die im Verlauf der Auswertung des algebraischen Ausdrucks noch gebraucht werden. Wir verwenden bei der Übersetzung von Konjunktionen zur Differenzierung dieser Menge von Attributen zwei Indizes für die Join-Operatoren. Der erste Index gibt die Menge der Variablen an, die bereits gebunden sind und noch benötigt werden, und der zweite Index die Menge der Variablen, die vom aktuellen Join-Operator gebunden und noch benötigt werden. Dies ist erstens bei der Verwendung des Anti-Joins notwendig, um eine sichere Auswertung zu garantieren. Zweitens erlaubt diese Information, alle nicht mehr benötigten Attribute während der Auswertung des Joins durch eine Projektion zu eliminieren.

Analog dem bekannten Semi-Join ist der komplementäre Join zweier Relationen eine Teilmenge des ersten Arguments und damit ebenfalls keine symmetrische Operation. Für die Implementierung des generalisierten Join-Operators als auch die des komplementären Join-Operators können bekannte Join-Algorithmen verwendet werden [Bra84, Sha86, ME92].

Beispiel 5.5 (Erweiterte Join-Operatoren)

Operator	*Formel*	*algebraischer Ausdruck*
gjoin	$\exists x \exists y\ p(x,y) \wedge q(x)$	$p(x,y) \bowtie_{\{x,y\},\{\emptyset\}} q(x)$
cjoin	$\exists x \exists y\ p(x,y) \wedge \neg q(x)$	$p(x,y) \overline{\bowtie}_{\{x,y\},\{\emptyset\}} q(x)$

Die neuen Operatoren der erweiterten relationalen Algebra sind in Tabelle 5.2 aufgeführt. Sie dienen als Basis für die Spezialoperatoren im folgenden Abschnitt.

Operator	Bezeichnung
$\bowtie_{X,Y}$	generalisierter Join
$\overline{\bowtie}_{X,Y}$	komplementärer Join
$\cup$	generalisierte Vereinigung (outer-union)

Tabelle 2: Operatoren der erweiterten relationalen Algebra

5.3 Übersetzungsregeln (Basisregeln)

Die zu übersetzenden Formeln liegen in disjunktiver Normalform vor (siehe Abschnitt 2):

$$g = Q_1 \ldots Q_l(Q_1 \ldots Q_{n_1}(l_1 \wedge \ldots \wedge l_{m_1}) \vee \ldots \vee Q_1 \ldots Q_{n_2}(l_1 \wedge \ldots \wedge l_{m_2}))$$
$$Q_i = \{\exists\, x_i, \forall\, x_i\}$$

Die Übersetzung $\mathcal{E}$ beginnt mit der *einmaligen* Anwendung der Startregel, die in Abhängigkeit vom syntaktischen Aufbau der Formel ausgewählt wird.

Definition 5.6 (Startregel) *Sei g eine bereichsbeschränkte Formel. Dann werden allquantifizierte Formeln zu Beginn der Transformation negiert und existenzquantifizierte Formeln unverändert in die weitere Transformation übernommen.*

- *f ist eine allquantifizierte Formel der Form $\forall\, x_1 \ldots x_n Q f'$.*

  $$\mathcal{E}_{v_{out}}(g) \rightsquigarrow \mathcal{E}_{v_{out}}(\sim g)$$

- *f ist eine existenzquantifizierte Formel der Form $\exists\, x_1 \ldots x_n Q f'$.*

  $$\mathcal{E}_{v_{out}}(g) \rightsquigarrow \mathcal{E}_{v_{out}}(g)$$

mit $v_{out} \subseteq \{x_1, \ldots, x_n\}$.

Die Variablenmenge v_{out} enthält alle Attribute, die das Schema der Ergebnisrelation bestimmen. Dies ist eine Teilmenge der ersten n, vom gleichen Quantor gebundenen Variablen. Die weitere Transformation startet nun in jedem Fall mit existenzquantifizierten Formeln.

Der Grund für die unterschiedliche Behandlung von allquantifizierten und existenzquantifizierten Formeln in der Startregel sind die im vorigen Kapitel angesprochenen Vereinfachungen bei der Auswertung der entstehenden algebraischen Ausdrücke.

Für die Auswertung des aus der Übersetzung $\mathcal{E}$ resultierenden Ausdruckes ist die unterschiedliche Ergebnisinterpretation von allquantifizierten und existenzquantifizierten Formeln zu beachten (siehe Abschnitt 4).

Definition 5.7 (Basisregeln)

Basisfälle

1. *f ist ein Atom der Form $f = p(x_1, \ldots, x_n)$, wobei die x_i $(i = 1, \ldots, n)$ entweder Variablen oder Konstanten sind.*

   $$\mathcal{E}_{v_{out}}(f) \rightsquigarrow p(x_1, \ldots, x_n)$$

   Falls p ein berechenbares Prädikat ist (siehe Abschnitt 2.3), müssen alle Variablen zum Auswertungszeitpunkt gebunden sein. Dies ist möglich, da die Formeln bereichsbeschränkt sind.

2. *$f = (l_1 \wedge \ldots \wedge l_n)$ und die l_i sind positive oder negative Literale.*

   $\mathcal{E}_{v_{out}}(f) \rightsquigarrow p_1\ op_1 \ldots op_{n-1}\ p_n$
   wobei $p_1, \ldots, p_n$ eine Permutation von $l_1, \ldots, l_n$ ist und p_i sich wie folgt aus l_i ergibt:

   $$p_i = \begin{cases} l_i & \text{falls } l_i \text{ ein positives Literal ist} \\ \neg l_i & \text{falls } l_i \text{ ein negatives Literal ist} \end{cases}$$

   - *$i = 1$*
     $p_1 = l_1$, falls l_1 entweder ein positives Literal oder ein negatives Literal, bei dem alle Variablen des Literals gebunden sind, ist.
   - *$i > 1$*
     - *l_i ist ein positives Literal*
       op_{i-1} ist $\bowtie_{\dot{X}_i, \dot{Y}_i}$
       $p_i = l_i$
     - *l_i ist ein negatives Literal und alle Variablen des Literals sind gebunden*
       op_{i-1} ist $\overline{\bowtie}_{\dot{X}_i, \dot{Y}_i}$
       $p_i = \neg l_i$
     - *l_i ist ein negatives Literal und die Variablen des Literals sind nicht vollständig vom bisher entstandenen Teilausdruck gebunden*
       op_{i-1} ist $\bowtie_{\dot{X}_i, \dot{Y}_i}$ $dom_{\bar{v}_l} \overline{\bowtie}_{\dot{X}_i, \dot{Y}_i}$ und $dom_{\bar{v}_l}$ ist der Bereich der Variablen in l
       $p_i = \neg l_i$

   Die Variablenmengen $\dot{X}_i, \dot{Y}_i$ sind die gebundenen bzw. vom aktuellen Join-Operator zu bindenden Variablen.

Induktion über den Aufbau der Formel

3. *$f = f_1 \vee \ldots \vee f_n$ und die f_i sind Konjunktionen.*

   $$\mathcal{E}_{v_{out}}(f) \rightsquigarrow \mathcal{E}_{v_{out}}(f_1) \cup \ldots \cup \mathcal{E}_{v_{out}}(f_n)$$

4. *$f = \exists\, x_1 \ldots x_n f'$ und f' ist eine bereichsbeschränkte Formel.*

$$\mathcal{E}_{v_{out}}(f) \rightsquigarrow \pi_{(schema(f')-(x_1,\ldots,x_n))\cup v_{out}}\mathcal{E}_{v_{out}}(f')$$

Es erfolgt eine Projektion auf die Ausgabevariablen v_{out}. Die Ausgabevariablen wurden eingeführt, da Konsistenzbedingungen geschlossene Formeln sind und damit keine freien Variablen besitzen , die dann die Ergebnismenge bilden könnten. Ansonsten (falls die Menge der Ausgabevariablen leer ist) muß zwischen einer Projektion auf der leeren Menge und einer Projektion auf der nichtleeren Menge (analog der Unterscheidung zwischen der leeren Menge und der leeren Substitution) unterschieden werden.

5. *$f = \forall\, x_1 \ldots x_n f'$ und f' ist eine bereichsbeschränkte Formel.*

   $$\mathcal{E}_{v_{out}}(f) \rightsquigarrow \mathcal{E}_{v_{out}}(f') \div \mathcal{E}(g^{dom_{x_1,\ldots,x_n}})$$

   Die Formel $g^{dom_{x_1,\ldots,x_n}}$ berechnet die Domäne der Variablen $x_1, \ldots, x_n$ aus den Restriktionsliteralen.

Die Berechnung von $\mathcal{E}(g^{dom_{x_1,\ldots,x_n}})$:

Gegeben sei eine bereichsbeschränkte Formel g in pränexer disjunktiver Normalform. Dann ist die Formel zur Berechnung der Domäne:

$g^{dom_{x_1,\ldots,x_n}} = (p_{1,1} \wedge \ldots \wedge p_{1,n}) \ \vee \ldots \vee \ (p_{m,1} \wedge \ldots \wedge p_{m,l})$
wobei $p_{i,j}$ das j-te negative Literal (Restriktionsliteral, siehe Definition 2.1) der i-ten Konjunktion ist.

Der algebraische Ausdruck zur Berechnung der Domäne einer Variablen x ist dann:

$\mathcal{E}(g^{dom_x}) = \pi_x(p_{1,1} \bowtie \ldots \bowtie p_{1,n}) \ \cup \ldots \cup \ \pi_x(p_{m,1} \bowtie \ldots \bowtie p_{m,l})$
wobei $p_{i,j}$ das j-te negative Literal (Restriktionsliteral) der i-ten Konjunktion ist.

Beispiel zur Berechnung von $\mathcal{E}(g^{dom_{x_1,\ldots,x_n}})$:

$$\begin{array}{ll} g & = \forall\, x\ \forall\, y\ p(x,y) \wedge q(x,y) \Longrightarrow r(x) \\ dnf(f) & = \forall\, x\ \forall\, y\ \neg p(x,y) \vee \neg q(x,y) \vee r(x) \\ g^{dom_{x,y}} & = p(x,y) \wedge q(x,y) \\ \mathcal{E}_{\{x,y\}}(g^{dom_{x,y}}) & = p(x,y) \bowtie_{\{x,y\},\{\emptyset\}} q(x,y) \end{array}$$

Beispiel 5.8 *Beispiele zur Übersetzung mittels Basisregeln:*

$$
\begin{array}{lll}
 & f & = \forall\, x\ \forall\, y\ p(x,y) \wedge q(x,y) \Longrightarrow r(x) \\
 & & = \forall\, x\ \forall\, y\ \neg p(x,y) \vee \neg q(x,y) \vee r(x) \\
 & \mathcal{E}_{\{x,y\}}(f) & = \mathcal{E}(\forall\, x\ \forall\, y\ \neg p(x,y) \vee \neg q(x,y) \vee r(x)) \\
1. & & \stackrel{5.6}{=} \mathcal{E}(\exists\, x\ \exists\, y\ p(x,y) \wedge q(x,y) \wedge \neg r(x)) \\
 & & \stackrel{5.7.4}{=} \pi_{x,y}\mathcal{E}(p(x,y) \wedge q(x,y) \wedge \neg r(x)) \\
 & & \stackrel{5.7.2}{=} \pi_{x,y}(p(x,y) \bowtie_{\{x,y\},\{\emptyset\}} q(x,y) \overline{\bowtie}_{\{x,y\},\{\emptyset\}} r(x))
\end{array}
$$

$$
\begin{array}{lll}
 & f & = \forall\, x\ \forall\, y\ p(x,y) \Longrightarrow q(x,y) \wedge r(y) \\
 & & = \forall\, x\ \forall\, y\ \neg p(x,y) \vee \neg(q(x,y) \wedge r(y)) \\
 & \mathcal{E}_{\{x,y\}}(f) & = \mathcal{E}(\forall\, x\ \forall\, y\ \neg p(x,y) \vee \neg(q(x,y) \wedge r(y))) \\
 & & \stackrel{5.6}{=} \mathcal{E}(\exists\, x\ \exists\, y\ (p(x,y) \wedge \neg q(x,y)) \vee (p(x,y) \wedge \neg r(y))) \\
2. & & \stackrel{5.7.4}{=} \pi_{x,y}\mathcal{E}((p(x,y) \wedge \neg q(x,y)) \vee (p(x,y) \wedge \neg r(y)) \\
 & & \stackrel{5.7.3}{=} \pi_{x,y}(\mathcal{E}(p(x,y) \wedge \neg q(x,y)) \cup \mathcal{E}(p(x,y) \wedge \neg r(y))) \\
 & & \stackrel{5.7.2}{=} \pi_{x,y}((p(x,y) \overline{\bowtie}_{\{x,y\},\{\emptyset\}} q(x,y)) \cup \mathcal{E}(p(x,y) \wedge \neg r(y))) \\
 & & \stackrel{5.7.2}{=} \pi_{x,y}((p(x,y) \overline{\bowtie}_{\{x,y\},\{\emptyset\}} q(x,y)) \cup (p(x,y) \overline{\bowtie}_{\{x,y,z\},\{\emptyset\}} r(y)))
\end{array}
$$

$$
\begin{array}{lll}
 & f & = \forall\, x\ \exists\, y\ p(x) \Longrightarrow q(x,y) \\
 & & = \forall\, x\ \exists\, y\ \neg p(x) \vee q(x,y) \\
 & \mathcal{E}_{\{x\}}(f) & = \mathcal{E}(\forall\, x\ \exists\, y\ \neg p(x) \vee q(x,y)) \\
3. & & \stackrel{5.6}{=} \mathcal{E}(\exists\, x\ \forall\, y\ p(x) \wedge \neg q(x,y)) \\
 & & \stackrel{5.7.4}{=} \pi_x\mathcal{E}(\forall\, y\ p(x) \wedge \neg q(x,y)) \\
 & & \stackrel{5.7.5}{=} \pi_x(\mathcal{E}(p(x) \wedge \neg q(x,y)) \div dom_y) \\
 & & \stackrel{5.7.2}{=} \pi_x((p(x) \bowtie_{\{x\},\{y\}} dom_y \overline{\bowtie}_{\{x,y\},\{\emptyset\}} q(x,y)) \div dom_y)
\end{array}
$$

mit $dom_y = \pi_y\ q(x,y)$.

$$
\begin{array}{lll}
 & f & = \forall\ x\ \exists\ y\ p(x) \Longrightarrow\ q_1(x,y) \wedge\ q_2(x,y) \\
 & & = \forall\ x\ \exists\ y\ \neg p(x) \vee\ q_1(x,y) \wedge\ q_2(x,y) \\
 & \mathcal{E}_{\{x\}}(f) & = \mathcal{E}(\forall\ x\ \exists\ y\ \neg p(x) \vee\ q_1(x,y) \wedge\ q_2(x,y)) \\
 & & \stackrel{5.6}{=} \mathcal{E}(\exists\, x\ \forall\, y\ p(x) \wedge \neg(q_1(x,y) \wedge\ q_2(x,y))) \\
4. & & \stackrel{5.7.4}{=} \pi_x\mathcal{E}(\forall\, y\ p(x) \wedge \neg(q_1(x,y) \wedge\ q_2(x,y))) \\
 & & \stackrel{5.7.5}{=} \pi_x(\mathcal{E}(p(x) \wedge \neg(q_1(x,y) \wedge\ q_2(x,y))) \div dom_y) \\
 & & \stackrel{5.7.2}{=} \pi_x(p(x) \bowtie_{\{x\},\{y\}} dom_y \overline{\bowtie}_{\{x,y\},\{\emptyset\}} q_1(x,y)) \cup \\
 & & \quad (p(x) \bowtie_{\{x\},\{y\}} dom_y \overline{\bowtie}_{\{x,y\},\{\emptyset\}} q_2(x,y))) \div dom_y)
\end{array}
$$

mit $dom_y = q_1(x,y) \bowtie_{\{x,y\},\{\emptyset\}} q_2(x,y)$

6 Spezialregeln zur Übersetzung von Konsistenzbedingungen

Die Verwendung von Spezialregeln zur Übersetzung von Konsistenzbedingungen in die Algebra verfolgt zwei Ziele:

- Aus der Spezifik des Aufbaus der logischen Formeln ist die Bildung von neuen Operatoren aus Gründen der effizienteren Auswertung wünschenswert. Die Übersetzung der Formeln in die um neue Operatoren erweiterte relationale Algebra erfolgt in Form von Spezialregeln.
- Für häufig vorkommende Muster von logischen Formeln ist es sinnvoll, Spezialregeln anzugeben, die eine direkte Übersetzung der Konsistenzbedingung ermöglichen. Das erspart nachfolgende (algebraische) Optimierungen und dient der effizienteren Übersetzung.

Im folgenden soll auf das erste Ziel näher eingegangen und die Einführung der neuen Operatoren motiviert werden. Ausgangspunkt der Überlegungen ist die Beobachtung, daß in Konsistenzbedingungen, die als prädikatenlogische Formeln spezifiziert sind, häufig Disjunktionen auftreten und diese möglichst effizient ausgewertet werden sollen.

6.1 Die Bypass-Technik

In Konsistenzbedingungen treten häufig Teilformeln der Form $p(x) \wedge (r(x) \vee s(x))$ auf. Ein Nachteil bei der Auswertung der konjunktiven Normalform (cnf) ist die Bildung der unnötig großen Zwischenrelation $r(x) \vee s(x)$. Bei der disjunktiven Normalform (dnf) $(p(x) \wedge r(x)) \vee (p(x) \wedge s(x))$ ist das zweimalige Lesen der Basisrelation $p(x)$ und eine u.U. notwendige Duplikatelimination von Nachteil. Eine weitere Beobachtung ist, daß Tupel aus $p(x)$, die einmal von $r(x)$ als wahr identifiziert worden sind, nicht von weiteren Teilformeln (hier: $s(x)$) der Disjunktion geprüft werden müssen.

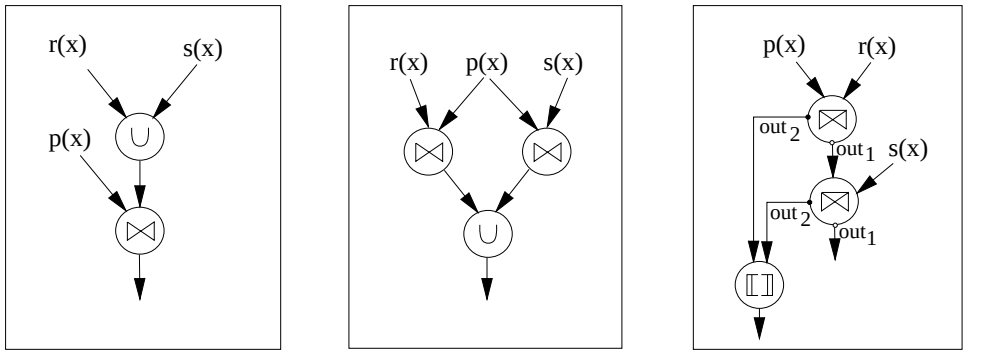


Abbildung 3: Motivation Bypass-Technik

Zur Vermeidung der aufgezeigten Nachteile wird die Idee der Bypass-Auswertung [KMPS94] verwendet. Ausgehend von der konjunktiven Normalform $p(x) \wedge (r(x) \vee s(x))$ werden alle Tupel, die sich durch die erste Operation $p(x) \wedge r(x)$ für das Ergebnis qualifizieren, an der zweiten Operation vorbeigeleitet, und nur die Tupel, die sich nicht qualifizieren konnten, werden mit $s(x)$ getestet.

Das motiviert einen Operator, der sich qualifizierende und sich nicht qualifizierende Tupel unterscheiden kann (ein generalisierter Join mit zwei Ausgabemengen) und die sich qualifizierenden Tupel an der weiteren Verarbeitung „vorbeileitet" (*Bypass-Technik* oder *Zweistromtechnik*). Am Ende des Teilausdruckes sollen dann alle sich qualifizierenden Tupel als Ergebnis erscheinen. Dies wird von einem weiteren Operator erledigt, der alle sich qualifizierenden Tupel aufnimmt und als *Vereinigungsklammer* bezeichnet wird.

6.2 Operatoren der erweiterten relationalen Algebra

Die neuen Operatoren besitzen keine feste Anzahl von Eingabeparametern und mehr als einen Ausgabeparameter. Damit ist es keine Algebra im strengen Sinne. Die Operatorfolgen sind folglich auch keine Bäume, sondern gerichtete, azyklische Graphen (DAG - directed acyclic graph). Die verbesserten Kommunikationsmöglichkeiten der Operatoren bilden die Basis für schnelle Konsistenztests.

Definition 6.1 (GJOIN) *Seien $R(x_1, \ldots, x_n)$ und $S(y_1, \ldots, y_m)$ zwei Relationen mit den Attributmengen $\{x_1, \ldots, x_n\}$ und $\{y_1, \ldots, y_m\}$. Dann wird der generalisierte Join-Operator (**GJOIN**) mit zwei Ausgabemengen folgendermaßen definiert.*

$$R(x_1, \ldots, x_n) \bowtie\!\!\!\!\!\!\bowtie_{X,Y} S(y_1, \ldots, y_m) = \begin{cases} R(x_1, \ldots, x_n) \bowtie_{X,Y} S(y_1, \ldots, y_m) & \text{1. Ausgabemenge} \\ R(x_1, \ldots, x_n) \overline{\bowtie}_{X,\emptyset} S(y_1, \ldots, y_m) & \text{2. Ausgabemenge} \end{cases}$$

mit $X \subseteq \{x_1, \ldots, x_n\}$ und $Y \subseteq \{y_1, \ldots, y_m\}$.

Die erste Ausgabemenge enthält alle Tupel, welche das Joinprädikat p (welches auf Gleichheit der Werte aller gleichbezeichneten Attribute testet) erfüllen. Die zweite Ausgabemenge enthält alle Tupel, die das Joinprädikat p nicht erfüllen.

Es ist manchmal notwendig, im Sinne einer linearen Notation die Ausgabemengen zu vertauschen. Dazu wird ein zum **GJOIN** komplementärer Operator definiert. Die Definition des komplementären Joins (**CJOIN**) ist analog der des generalisierten Joins, nur entspricht die erste Ausgabemenge des komplementären Joins der zweiten Ausgabemenge des generalisierten Joins und umgekehrt.

Definition 6.2 (CJOIN) *Seien $R(x_1, \ldots, x_n)$ und $S(y_1, \ldots, y_m)$ zwei Relationen mit den Attributmengen $\{x_1, \ldots, x_n\}$ und $\{y_1, \ldots, y_m\}$. Dann wird der komplementäre Join-Operator (**CJOIN**) mit zwei Ausgabemengen folgendermaßen definiert.*

$$R(x_1, \ldots, x_n) \overline{\bowtie\!\!\!\!\!\!\bowtie}_{X,Y} S(y_1, \ldots, y_m) = \begin{cases} R(x_1, \ldots, x_n) \overline{\bowtie}_{X,\emptyset} S(y_1, \ldots, y_m) & \text{1. Ausgabemenge} \\ R(x_1, \ldots, x_n) \bowtie_{X,Y} S(y_1, \ldots, y_m) & \text{2. Ausgabemenge} \end{cases}$$

mit $X \subseteq \{x_1, \ldots, x_n\}$ und $Y \subseteq \{y_1, \ldots, y_m\}$.

Zur Darstellung werden die Ausgänge der generalisierten Join-Operatoren mit zwei Ausgängen besonders gekennzeichnet: verallgemeinerter Join (gjoin •) und komplementärer Join (cjoin ○).

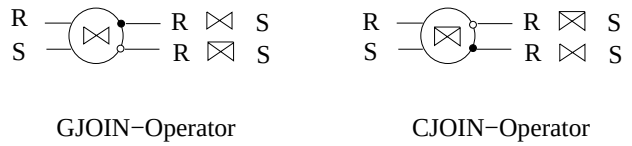


Abbildung 4: Verallgemeinerte Join-Operatoren mit 2 Ausgängen

Der wesentliche *Vorteil* dieser Operatoren ist ihre Eigenschaft, die zweite Ausgabemenge ohne wesentlichen Mehraufwand gegenüber einer einfachen Join-Operation berechnen zu können, da sie als Nebeneffekt bei der Berechnung der ersten Ausgabemenge entsteht. Das erfordert nur eine geringfügige Modifikation der Implementierung des Join-Operators.

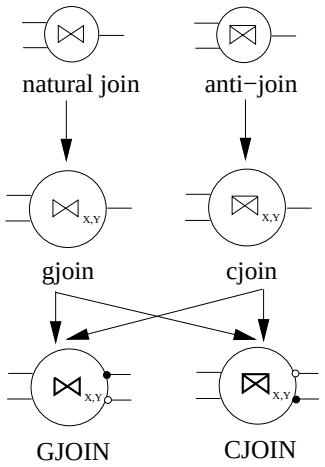


Abbildung 5: Verallgemeinerungshierarchie der Join-Operatoren

Die im folgenden definierten Vereinigungsklammern legen nicht eine Abarbeitungsreihenfolge im Sinne einer echten Klammerung fest, sondern grenzen den Bereich der algebraischen Ausdruckes ein, in welchem die zweiten Ausgabemengen der Join-Operatoren zur Ergebnisbildung verwendet werden.

Definition 6.3 (Vereinigungsklammer)

Seien S ein algebraischer Ausdruck und R_i Relationen. Dann sind die Vereinigungsklammern $[\![\ldots]\!]$ wie folgt definiert:

$$
\begin{aligned}
S\,[\![\dot{\bowtie} R_1 \dot{\bowtie} R_2 \dot{\bowtie} \ldots \dot{\bowtie} R_n]\!] \quad = \quad & S \,\overline{\dot{\bowtie}}\, R_1 \\
& \cup\ S \ \dot{\bowtie}\ R_1 \,\overline{\dot{\bowtie}}\, R_2 \\
& \cup \ldots \\
& \cup\ S \ \dot{\bowtie}\ R_1 \ \dot{\bowtie}\ R_2 \dot{\bowtie}\ \ldots \overline{\dot{\bowtie}}\, R_n
\end{aligned}
$$

mit $\dot{\bowtie} = \{\bowtie, \boxtimes\}$ und $\overline{\dot{\bowtie}}$ als komplementärer Operator zu $\dot{\bowtie}$.

Dieser Operator sammelt alle zweiten Ausgabemengen der Join-Operatoren innerhalb der Vereinigungsklammern $[\![\ldots]\!]$ und leitet sie an den folgenden Operator weiter.

Nun stehen Operatoren zur Verfügung, um die Idee der Bypass-Technik in der erweiterten relationalen Algebra darstellen zu können. Als erstes Beispiel wird eine einfache Konsistenzbedingung gewählt, welche die zur Motivation der Bypass-Technik verwendete Teilformel in der Prämisse enthält.

Beispiel 6.4 (Anwendung der Bypass-Technik (1)) *Gegeben sei ein typisches Muster für eine Konsistenzbedingung:* $c_1 = \forall\, x\ p(x) \wedge (r(x) \vee s(x)) \Longrightarrow\ t(x)$

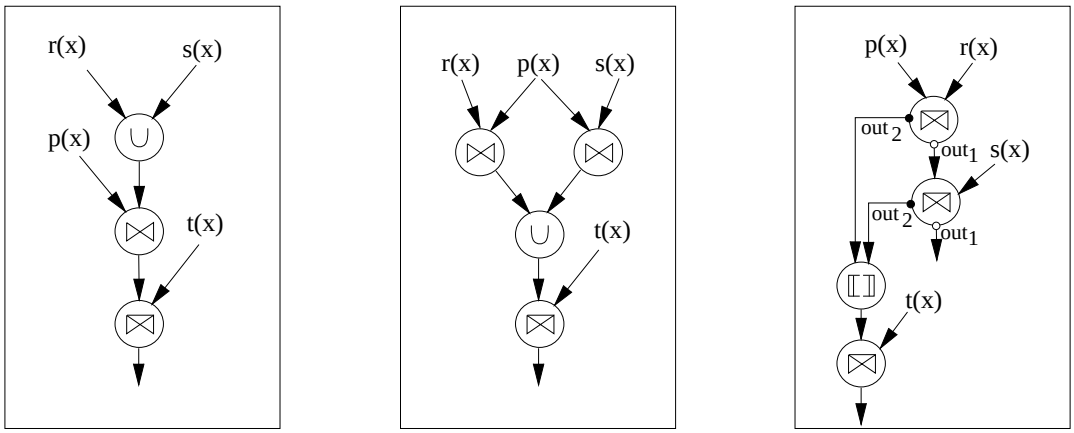
$$\begin{aligned}\mathcal{E}_{v_{out}}(c_1) &= \mathcal{E}_{\{x\}}(\sim c_1)\\ &= \mathcal{E}_{\{x\}}(\exists\, x\ p(x) \wedge (r(x) \vee s(x)) \wedge \neg t(x)\\ &= \mathcal{E}_{\{x\}}(\exists\, x\ (p(x) \wedge \neg t(x) \wedge r(x)) \vee (p(x) \wedge \neg t(x) \wedge s(x)))\end{aligned}$$

Mit den Übersetzungsregeln (Basisregeln und Spezialregeln) können nun alternative algebraische Ausdrücke gebildet werden.

1. $p(x) \bowtie_{\{x\},\{\emptyset\}} (r(x) \cup s(x)) \overline{\bowtie}_{\{x\},\{\emptyset\}} t(x))$
   aus der konjunktiven Normalform (cnf) mit den Basisregeln.
   Nachteilig ist die Bildung der Zwischenrelation $(r(x) \cup s(x))$*, wenn viele Tupel darin nichts zum Ergebnis der Prämisse beitragen.*

2. $(p(x) \bowtie_{\{x\},\{\emptyset\}} r(x)) \cup (p(x) \bowtie_{\{x\},\{\emptyset\}} s(x)) \overline{\bowtie}_{\{x\},\{\emptyset\}} t(x)$
   aus der disjunktiven Normalform (dnf) mit den Basisregeln.
   Nachteilig sind das zweimalige Lesen der Basisrelation $p(x)$ *und die entstehenden Duplikate vor der Vereinigung.*

3. $p(x) [\![\overline{\bowtie}_{\{x\},\{\emptyset\}} r(x) \overline{\bowtie}_{\{x\},\{\emptyset\}} s(x)]\!] \overline{\bowtie}_{\{x\},\{\emptyset\}} t(x)$
   mit der Bypass-Technik als Spezialregel.
   Die genannten Nachteile werden vermieden.



Ausdruck 1 (nach cnf) Ausdruck 2 (nach dnf) Ausdruck 3 (Bypass−Technik)

Abbildung 6: Alternative algebraische Ausdrücke Beispiel 1

In Anwendungen genügen Konsistenzbedingungen oft dem folgenden Muster:
$c = \forall\, x_1, \ldots, x_n\ p(x_1, \ldots, x_n) \implies q_1(x_1) \wedge \ldots \wedge q_n(x_n)$

Beispiel 6.5 (Anwendung der Bypass-Technik (2)) *Gegeben sei ein typisches Muster für eine Konsistenzbedingung:* $c_2 = \forall\, x\ y\ z\ p(x,y,z) \implies q_1(x) \wedge q_2(y) \wedge q_3(z)$

$$\begin{aligned}
\mathcal{E}_{v_{out}}(c_2) &= \mathcal{E}_{\{x,y,z\}}(\sim c_2) \\
&= \mathcal{E}_{\{x,y,z\}}(\exists\, x\ y\ z \quad p(x,y,z) \wedge \neg(q_1(x) \wedge q_2(y) \wedge q_3(z)) \\
&= \mathcal{E}_{\{x,y,z\}}(\exists\, x\ y\ z \quad (p(x,y,z) \wedge \neg q_1(x)) \vee \\
&\qquad\qquad\qquad\quad (p(x,y,z) \wedge \neg q_2(y)) \vee \\
&\qquad\qquad\qquad\quad (p(x,y,z) \wedge \neg q_3(z)))
\end{aligned}$$

Mit den Übersetzungsregeln (Basisregeln und Spezialregeln) können nun alternative algebraische Ausdrücke gebildet werden.

1. $p(x,y,z) \overline{\bowtie}_{\{x,y,z\},\{\emptyset\}} (q_1(x) \bowtie_{\{x\},\{y\}} q_2(y) \bowtie_{\{x,y\},\{z\}} q_3(z))$
   aus der konjunktiven Normalform (cnf) mit den Basisregeln.
   Die Auswertung ist wegen der Bildung von Kreuzprodukten sehr aufwendig und führt zu einer (möglicherweise sehr großen) Zwischenrelation, die nicht notwendig ist.

2. $(p(x,y,z) \overline{\bowtie}_{\{x,y,z\},\{\emptyset\}} q_1(x)) \cup (p(x,y,z) \overline{\bowtie}_{\{x,y,z\},\{\emptyset\}} q_2(y)) \cup (p(x,y,z) \overline{\bowtie}_{\{x,y,z\},\{\emptyset\}} q_3(z))$
   aus der disjunktiven Normalform (dnf) mit den Basisregeln.
   Nachteilig ist hier das mehrmalige Lesen der Basisrelation $(p(x,y,z))$.

3. $p(x,y,z)[\![\bowtie_{\{x,y,z\},\{\emptyset\}} q_1(x) \bowtie_{\{x,y,z\},\{\emptyset\}} q_2(y) \bowtie_{\{x,y,z\},\{\emptyset\}} q_3(z)]\!]$
   mit der Bypass-Technik als Spezialregel. Die Auswertung liefert genau die Tupel, welche entweder nicht in q_1 *oder nicht in* q_2 *oder nicht in* q_3 *sind. Dabei werden Tupel, die nicht in* q_1 *sind, also schon als Ergebnistupel feststehen, im Laufe der weiteren Verarbeitung nicht mehr behandelt.*

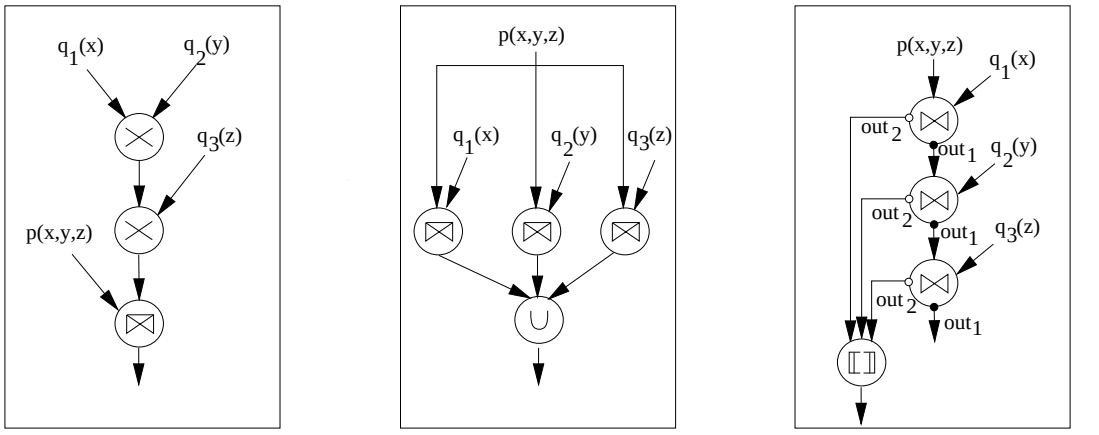


Abbildung 7: Alternative algebraische Ausdrücke Beispiel 2

Beispiel 6.6 *Weitere typische Muster von Konsistenzbedingungen und die dazugehörigen algebraischen Ausdrücke:*

- $\forall\, x\ y\ z\ \ p(x,y,z) \implies\ \ q_1(x,y) \wedge\ \ q_2(y,z) \wedge\ \ q_3(z)$
  $p(x,y,z)\ [\![\ \bowtie_{\{x,y,z\},\{\emptyset\}} q_1(x,y)\ \bowtie_{\{x,y,z\},\{\emptyset\}} q_2(y,z)\ \bowtie_{\{x,y,z\},\{\emptyset\}} q_3(z)\ ]\!]$
- $\forall\, x\ y\ z\ \ p(x,y,z) \implies\ \ q_1(x,y) \vee\ \ (q_2(y) \wedge\ \ q_3(z))$
  $p(x,y,z)\ \overline{\bowtie}_{\{x,y,z\},\{\emptyset\}} q_1(x,y)\ [\![\ \bowtie_{\{z,x,y\},\{\emptyset\}} q_2(y)\ \bowtie_{\{z,x,y\},\{\emptyset\}} q_3(z)\ ]\!]$
- $\forall\, x\ y\ z\ \ p(x,y,z) \implies\ \ q_1(x,y) \wedge\ \ (q_2(y) \vee\ \ q_3(z))$
  $(p(x,y,z)\ \overline{\bowtie}_{\{x,y,z\},\{\emptyset\}} q_1(x,y)) \cup$
  $(p(x,y,z)\ \overline{\bowtie}_{\{x,y,z\},\{\emptyset\}} q_3(z)\ \overline{\bowtie}_{\{x,y,z\},\{\emptyset\}} q_2(y))$
- $\forall\, x\ p(x) \wedge (r(x) \vee s(x)) \implies\ t(x)$
  $p(x)[\![\overline{\bowtie}_{\{x\},\{\emptyset\}} r(x) \overline{\bowtie}_{\{x\},\{\emptyset\}} s(x)]\!]\ \overline{\bowtie}_{\{x\},\{\emptyset\}} t(x)$

Ein Sonderfall der Vereinigungsklammern sind die Filterklammern, die bei der Übersetzung von Teilformeln auftreten, deren Literale allquantifizierte Variablen enthalten.

Das Ergebnis dieses Operators ist die Vereinigung aller zweiten Ausgabemengen ohne die erste Ausgabemenge des letzten Operators innerhalb der Filterklammer $[\![\ldots]\!]^-$, jeweils projiziert auf eine Variablenmenge Z. Die Variablenmenge Z besteht aus den Variablen, die am Beginn dieser speziellen Vereinigungsklammern schon gebunden sind.

Definition 6.7 (Filterklammern)

Seien S ein algebraischer Ausdruck und R_i Relationen. Dann ist die Filterklammer $[\![\ldots]\!]^-$ wie folgt definiert:

$$\begin{aligned} S\,[\![\dot{\bowtie} R_1 \dot{\bowtie} R_2 \dot{\bowtie} \ldots \dot{\bowtie} R_n]\!]^- \quad = \quad & \pi_Z(S \overline{\dot{\bowtie}} R_1 \\ & \cup\ S\ \dot{\bowtie}\ R_1 \overline{\dot{\bowtie}} R_2 \\ & \cup \ldots \\ & \cup\ S\ \dot{\bowtie}\ R_1\ \dot{\bowtie}\ R_2 \dot{\bowtie}\ \ldots \overline{\dot{\bowtie}} R_n) \\ & \Leftrightarrow \pi_Z(S\ \dot{\bowtie} R_1 \dot{\bowtie} R_2 \dot{\bowtie} \ldots \dot{\bowtie} R_n) \end{aligned}$$

mit $\dot{\bowtie} = \{\bowtie, \overline{\bowtie}\}$ und $\overline{\dot{\bowtie}}$ als komplementärer Operator zu $\dot{\bowtie}$.

Beispiel 6.8 (Anwendung der Bypass-Technik (3)) *Gegeben sei die folgende Konsistenzbedingung:*
$c_3 = \forall\ x\ \exists\ y\ p(x) \implies\ \ q_1(x,y) \wedge\ \ q_2(x,y)$

$$\begin{aligned} \mathcal{E}_{v_{out}}(c_3) &= \mathcal{E}_{\{x\}}(\sim c_3) \\ &= \mathcal{E}_{\{x\}}(\exists x \forall y\ p(x) \wedge \neg(q_1(x,y) \wedge q_2(x,y))) \\ &= \mathcal{E}_{\{x\}}(\exists x \forall y\ (p(x) \wedge \neg q_1(x,y)) \vee (p(x) \wedge \neg q_2(x,y))) \end{aligned}$$

Entsprechend den Übersetzungsregeln können folgende alternative algebraische Ausdrücke generiert werden.

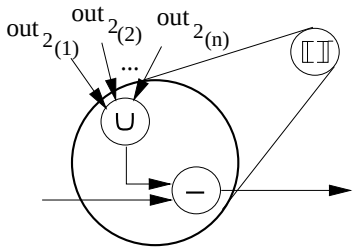


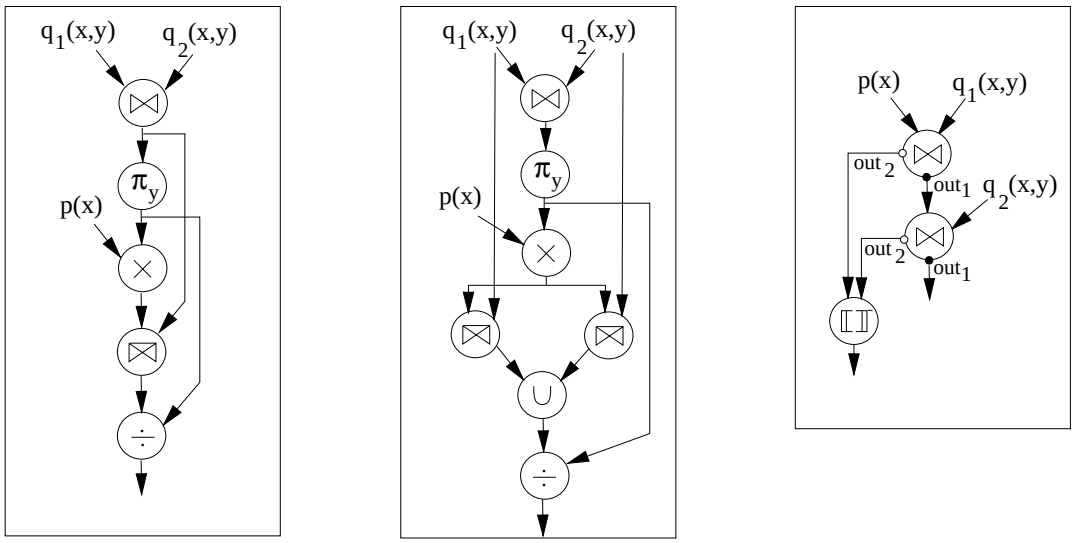
Abbildung 8: Filterklammer als komplexer Operator der XRA

1. (a) $(p(x) \bowtie_{\{x\},\{y\}} dom(y) \overline{\bowtie}_{\{x,y\},\{\emptyset\}} (q_1(x,y) \bowtie_{\{x,y\},\{\emptyset\}} q_2(x,y))) \div dom(y)$
   mit $dom(y) = \pi_y(q_1(x,y) \bowtie_{\{x,y\},\{\emptyset\}} q_2(x,y))$
   aus der konjunktiven Normalform (cnf) mit den Basisregeln,

   (b) $p(x) \overline{\bowtie}_{\{x\},\{\emptyset\}} (q_1(x,y) \bowtie_{\{x,y\},\{\emptyset\}} q_2(x,y))$
   aus der konjunktiven Normalform (cnf) mit den Spezialregeln,

2. $((p(x) \bowtie_{\{x\},\{y\}} dom(y) \overline{\bowtie}_{\{x,y\},\{\emptyset\}} q_1(x,y)) \cup$
   $(p(x) \bowtie_{\{x\},\{y\}} dom(y) \overline{\bowtie}_{\{x,y\},\{\emptyset\}} q_2(x,y))) \div dom(y)$
   mit $dom(y) = \pi_y(q_1(x,y) \bowtie_{\{x,y\},\{\emptyset\}} q_2(x,y))$
   aus der disjunktiven Normalform (dnf) mit den Basisregeln,

3. $p(x)[\![\bowtie_{\{x\},\{y\}} q_1(x,y) \bowtie_{\{x\},\{\emptyset\}} q_2(x,y)]\!]^-$
   mit der Bypass-Technik als Spezialregel.



Ausdruck 1 (nach cnf) Ausdruck 2 (nach dnf) Ausdruck 3 (Bypass−Technik)

Abbildung 9: Alternative algebraische Ausdrücke Beispiel 3

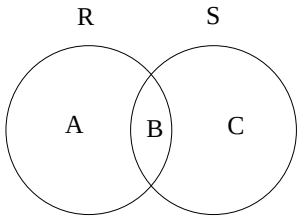
Verallgemeinerte Join-Operatoren mit zwei Ausgängen treten nur innerhalb der Vereinigungsklammern und Filterklammern auf.

Operator	Bezeichnung
$\bowtie$	generalisierter Join (GJOIN) 2 Ausgabemengen
$\overline{\bowtie}$	komplementärer Join (CJOIN) 2 Ausgabemengen
$[\![\ldots]\!]$	Vereinigungsklammer (UNION-BRACKETS)
$[\![\ldots]\!]^-$	Filterklammer (FILTER-BRACKETS)

Tabelle 3: Zusammenfassung der Spezialoperatoren der erweiterten relationalen Algebra (XRA)

Die Vorteile der Bypass-Technik lassen sich am Beispiel der Teilformel $p(x) \wedge (r(x) \vee s(x))$ wie folgt zusammenfassen.

- Die Basisrelation (hier: $p(x)$) wird im Gegensatz zur Auswertung der disjunktiven Normalform (dnf) $(p(x) \wedge r(x)) \vee (p(x) \wedge s(x))$ nur einmal gelesen. Weiterhin ist auch keine Duplikatelimination erforderlich.
- Es werden keine unnötig großen Zwischenrelationen $r(x) \vee s(x)$ wie bei der Auswertung der konjunktiven Normalform (cnf) gebildet.
- Tupel aus $p(x)$, die einmal von $r(x)$ als wahr identifiziert worden sind, müssen nicht von weiteren Teilformeln (hier: $s(x)$) der Disjunktion geprüft werden.
- Die Arbeitsweise ist weiterhin mengenorientiert.



Ausgabe	Gleichheit aller Attribute	Gleichheit einiger Attribute
A	Differenz	cjoin
B	Durchschnitt	gjoin
C	Differenz	cjoin
A,B		GJOIN
C,B		GJOIN
A,B,C	Vereinigung	

Tabelle 4: Zusammenhang von sog. „one-to-one-match“ Operatoren der erweiterten relationalen Algebra (XRA). Diesen Operatoren ist die Eigenschaft gemeinsam, daß ein Eingabetupel in Abhängigkeit von einem anderen Eingabetupel in der Ausgabemenge erscheint.

6.3 Übersetzungsregeln (Spezialregeln)

Die Übersetzung $\mathcal{E}$ beginnt mit der *einmaligen* Anwendung der Startregel (Definition 5.6), die in Abhängigkeit vom syntaktischen Aufbau der Formel ausgewählt wird.

Zu Beginn der Anwendung der im folgenden Abschnitt definierten *Spezialregeln* ist die Einführung einer Menge von Variablen notwendig, die während der Übersetzung diejenigen Variablen enthält, die von den bisher generierten Operatoren der Algebra gebunden wird. Das ist für die sichere Auswertung negativer Literale erforderlich. Da diese Variablenmenge v_{in} auch über die gerade von einer Regel bearbeitete Teilformel hinaus Wirkung hat (für alle nachfolgenden Teilformeln mit negativen Literalen), muß sie während der Transformation mitgeführt werden.

Definition 6.9 (Initialregel)

$$\mathcal{E}_{v_{out}}(f) \rightsquigarrow \mathcal{E}_{v_{in},v_{out}}(f),\ mit\ v_{in} = \emptyset$$

Der Übersetzungsvorgang terminiert, wenn alle Teilformeln f von g vollständig übersetzt sind.

Definition 6.10 (Terminierungsregel)

$$\mathcal{E}_{v_{in},v_{out}}(\Lambda) \rightsquigarrow \Lambda \quad wobei\ \Lambda\ für\ das\ leere\ Wort\ steht.$$

6.3.1 Regeln für existenzquantifizierte Formeln

Wir beschreiben nun die Behandlung existenzquantifizierter Formeln. Die erste Übersetzungsregel („Schüttelregel“) erlaubt die Anordnung der Teilformeln in vielen Möglichkeiten und kann somit zahlreiche Alternativen generieren.

Definition 6.11 *Sei f eine Formel*

$$dnf(f) = \exists x_1, \ldots, x_n Q f'\ mit\ f' = (l_1^1 \wedge \ldots \wedge l_{n_1}^1) \vee \ldots \vee (l_1^\alpha \wedge \ldots \wedge l_{n_\alpha}^\alpha)$$

für eine Folge von Quantoren Q. Dann können wir zwei allgemeine Übersetzungsregeln für existenzquantifizierte Formeln folgendermaßen angeben:

$$\mathcal{E}_{v_{in},v_{out}}(\exists x_1, \ldots, x_n Q f') \rightsquigarrow$$
$$\mathcal{E}_{v_{in},v_{out}}(\exists x_1, \ldots, x_n Q_i\, c_{1'} \vee \ldots \vee c_{j_1'}) \cup \ldots \cup \mathcal{E}_{v_{in},v_{out}}(\exists x_1, \ldots, x_n Q_i\, c_{j_\beta'} \vee \ldots \vee c_{\alpha'})$$

und die $c_{i'}$ $(1 \leq i \leq \alpha)$ sind Permutationen der Konjunktionen von f' und die Q_i sind disjunkte Quantorenfolgen aus Q, die nur in einer Teilformel vorkommen.

$\mathcal{E}_{v_{in},v_{out}}(\exists x_1,\ldots,x_n f') \rightsquigarrow$
$\mathcal{E}_{v_{in},v_{out}}(\exists x_1,\ldots,x_n\, c_{1'} \vee \ldots \vee c_{j'_1}) \cup \ldots \cup \mathcal{E}_{v_{in},v_{out}}(\exists x_1,\ldots,x_n\, c_{j'_\beta} \vee \ldots \vee c_{\alpha'})$
und die $c_{i'}$ $(1 \leq i \leq \alpha)$ *sind Permutationen der Konjunktionen von* f'.

Definition 6.12 *Sei* f *eine Formel*

$$dnf(f) = \exists x_1,\ldots,x_n Q f' \text{ mit } f' = (l^1_1 \wedge \ldots \wedge l^1_{n_1}) \vee \ldots \vee (l^\alpha_1 \wedge \ldots \wedge l^\alpha_{n_\alpha})$$

für ein Folge von Quantoren Q.

Dann definieren wir die Übersetzung von $\mathcal{E}_{v_{in},v_{out}}(f)$ *entsprechend der Struktur von* f:

1. $\mathcal{E}_{v_{in},v_{out}}(\exists x_1,\ldots,x_n Q f') \rightsquigarrow op_1\ l_{1'}\ \ldots\ op_l\ l_{l'}\ \mathcal{E}_{v'_{in},v'_{out}}(\exists x_{i_1},\ldots,x_{i_m} Q f'')$
   wenn die folgenden Bedingungen gelten:
   (a) *die Literale* $l_{j'}$ $(1 \leq j \leq l)$ *treten in jeder Konjunktion von* f' *auf,*
   (b) $\{x_{i_1},\ldots,x_{i_m}\} = \{x_1,\ldots,x_n\} \setminus \bigcup_{1\leq j\leq l} free(l_{j'})$,
   (c) $op_i = \ltimes_{\dot{X}_i,\dot{Y}_i}$, *wenn* $l_{i'}$ *ein positives Literal ist,*
   (d) $op_i = \bar{\ltimes}_{\dot{X}_i,\dot{Y}_i}$, *wenn* $l_{i'}$ *ein negatives Literal ist und* $free(l_{i'}) \subseteq Z_{bound}(i)$,
   (e) f'' *ergibt sich aus* f' *durch Löschen der* $l_{i'}$ $(1 \leq j \leq l)$ *mit* $free(l_{i'}) \subseteq Z_{bound}(i)$ *und nachfolgender Vereinfachung,*
   (f) $v'_{in} = Z_{bound}(l+1) \cap Z_{needed}(l)$,
   $v'_{out} \subseteq v_{out} \cup \bigcup_{1\leq j\leq l} free(l_{j'})$ *und* $v'_{out} \subseteq v_{in} \cup \{x_1,\ldots,x_n\}$,

   für $Z_{tobind} := \{x_1,\ldots,x_n\} \setminus \{x_{i_1},\ldots,x_{i_m}\}$, *und*
   $Z_{needed}(i) = v'_{out} \cup free(\exists x_{i_1},\ldots,x_{i_m} Q f'') \cup \bigcup_{i<j\leq l} free(l_{j'})$.

2. $\mathcal{E}_{v_{in},v_{out}}(\exists x_1,\ldots,x_n l_1 \wedge \ldots \wedge l_l) \rightsquigarrow op_1\ l_{1'} \cdots op_l\ l_{l'}$
   wenn die folgenden Bedingungen gelten:
   (a) $l_{1'},\ldots,l_{l'}$ *ist eine Permutation von* $l_1,\ldots,l_l$
   (b) $op_i = \ltimes_{\dot{X}_i,\dot{Y}_i}$, *wenn* $l_{i'}$ *ein positives Literal ist,*
   (c) $op_i = \bar{\ltimes}_{\dot{X}_i,\dot{Y}_i}$, *wenn* $l_{i'}$ *ein negatives Literal ist und* $free(l_{i'}) \subseteq Z_{bound}(i)$,

   für $Z_{tobind} = \{x_1,\ldots,x_n\}$, *und* $Z_{needed} = v_{out} \cup \bigcup_{i<j\leq l} free(l_{j'})$.

3. $\mathcal{E}_{v_{in},v_{out}}(\exists x_1,\ldots,x_n l_1 \vee \ldots \vee l_l) \rightsquigarrow [\![op_1\ l_{1'} \cdots op_l\ l_{l'}]\!]$
   wenn die folgenden Bedingungen gelten:
   (a) $l_{1'},\ldots,l_{l'}$ *ist eine Permutation von* $l_1,\ldots,l_l$
   (b) $op_i = \bar{\ltimes}_{\dot{X}_i,\dot{Y}_i}$, *wenn* $l_{i'}$ *ein positives Literal ist und* $v_{out} \subseteq v_{in} \cup \bigcap_{1\leq j\leq l} free(l_{j'})$,
   (c) $op_i = \ltimes_{\dot{X}_i,\dot{Y}_i}$, *wenn* $l_{i'}$ *ein negatives Literal ist und* $free(l_{i'}) \subseteq v_{in}$ *und* $v_{out} \subseteq v_{in}$,

   für $Z_{tobind} = \{x_1,\ldots,x_n\}$, *und* $Z_{needed} = v_{out}$.

mit

$Z_{bound}(i) = v_{in} \cup (\bigcup_{1 \leq j' < i} free(l_{j'}) \cap Z_{tobind})$,
$Z_{tobind}(i) = (free(l_i) \setminus Z_{bound}(i)) \cap Z_{tobind}$,
$\dot{X}_i = Z_{bound}(i) \cap Z_{needed}$, *und*
$\dot{Y}_i = Z_{tobind}(i) \cap Z_{needed}$.

6.3.2 Regeln für allquantifizierte Formeln

Auch hier wird mit der allgemeinen Regel begonnen, die eine Aufsplittung der Formeln in Teilformeln, die dann separat übersetzt werden, erlauben und Raum für Optimierungen hinsichtlich der Anordnung der Teilformeln läßt.

Definition 6.13 *Sei f eine Formel*

$$cnf(f) = \forall\, x_1, \ldots, x_n Q f' \text{ mit } f' = (l_1^1 \vee \ldots \vee l_{n_1}^1) \wedge \ldots \wedge (l_1^\alpha \vee \ldots \vee l_{n_\alpha}^\alpha)$$

für eine Folge von Quantoren Q.

Seien v_{in} und v_{out} Variablenmengen so daß $v_{in} \neq \emptyset$ und $v_{out} \subseteq v_{in}$ gilt. Dann werden die allgemeinen Übersetzungsregeln von $\mathcal{E}$ für allquantifizierte Formeln folgendermaßen definiert:

$\mathcal{E}_{v_{in},v_{out}}(\forall\, x_1, \ldots, x_n Q f') \rightsquigarrow$
$\mathcal{E}_{v_{in},v_{out}}(\forall\, x_1, \ldots, x_n Q_i d_{1'} \wedge \ldots \wedge d_{j'_1}) \cap \ldots \cap \mathcal{E}_{v_{in},v_{out}}(\forall\, x_1, \ldots, x_n Q_i d_{j'_\beta} \wedge \ldots \wedge d_{\alpha'})$
und die $d_{i'}$ $(1 \leq i \leq \alpha)$ sind Permutationen der Disjunktionen von $cnf(f)$ und die Q_i sind disjunkte Quantorenfolgen aus Q, die nur in einer Teilformel vorkommen.

$\mathcal{E}_{v_{in},v_{out}}(\forall\, x_1, \ldots, x_n f') \rightsquigarrow$
$\mathcal{E}_{v_{in},v_{out}}(\forall\, x_1, \ldots, x_n\, d_{1'} \wedge \ldots \wedge d_{j'_1}) \cap \ldots \cap \mathcal{E}_{v_{in},v_{out}}(\forall\, x_1, \ldots, x_n\, d_{j'_\beta} \wedge \ldots \wedge d_{\alpha'})$
und die $d_{i'}$ $(1 \leq i \leq \alpha)$ sind Permutationen der Disjunktionen von $cnf(f)$.

Es folgen die Übersetzungsregeln für spezielle Muster von Teilformeln.

Definition 6.14 *Sei f eine Formel*

$$cnf(f) = \forall\, x_1, \ldots, x_n Q f' \text{ mit } f' = (l_1^1 \vee \ldots \vee l_{n_1}^1) \wedge \ldots \wedge (l_1^\alpha \vee \ldots \vee l_{n_\alpha}^\alpha)$$

für eine Folge von Quantoren Q. Seien v_{in} und v_{out} Variablenmengen, so daß $v_{in} \neq \emptyset$ und $v_{out} \subseteq v_{in}$ gilt. Dann sind die Übersetzungsregeln folgendermaßen definiert:

1. $\mathcal{E}_{v_{in},v_{out}}(\forall\, x_1,\ldots,x_n Qf') \rightsquigarrow [\![op_1\ l_{1'}\ \ldots\ op_l\ l_{l'}]\!]\ \mathcal{E}_{v'_{in},v'_{out}}(\forall\, x_{i_1},\ldots,x_{i_m} Qf'')$
wenn die folgenden Bedingungen gelten:

   (a) *die Literale* $l_{j'}$ $(1 \le j \le l)$ *treten in jeder Disjunktion von* f' *auf,*

   (b) $\{x_{i_1},\ldots,x_{i_m}\} = \{x_1,\ldots,x_n\} \setminus \bigcup_{1\le j\le l} free(l_j)$,

   (c) $op_i = \overset{\cdot}{\bowtie}_{\dot{X}_i,\dot{Y}_i}$, *wenn* $l_{i'}$ *ein positives Literal ist und* $free(l_{i'}) \subseteq Z_{bound}(i)$,

   (d) $op_i = \overset{\cdot}{\ltimes}_{\dot{X}_i,\dot{Y}_i}$, *wenn* $l_{i'}$ *ein negatives Literal ist,*

   (e) f'' *ergibt sich aus* f' *durch Löschen der* $l_{i'}$ $(1 \le j \le l)$ *mit* $free(l_{i'}) \subseteq Z_{bound}(i)$ *und nachfolgender Vereinfachung,*

   (f) $v'_{in} = Z_{bound}(l+1) \cap Z_{needed}(l)$,
   $v'_{out} \subseteq v_{out} \cup \bigcup_{1\le j\le l} free(l_{j'})$ *und* $v'_{out} \subseteq v_{in} \cup \{x_1,\ldots,x_n\}$,

   für $Z_{tobind} := \{x_1,\ldots,x_n\} \setminus \{x_{i_1},\ldots,x_{i_m}\}$, *und*
   $Z_{needed}(i) = v'_{out} \cup free(\forall x_{i_1},\ldots,x_{i_m} Qf'') \cup \bigcup_{i<j\le l+k} free(l_{j'})$.

2. $\mathcal{E}_{v_{in},v_{out}}(\forall\, x_1,\ldots,x_n l_1 \wedge \ldots \wedge l_l) \rightsquigarrow op_1\ l_{1'} \cdots op_l\ l_{l'}$
wenn die folgenden Bedingungen gelten:

   (a) $l_{1'},\ldots,l_{l'}$ *ist eine Permutation von* $1,\ldots,l$.

   (b) $op_i = \overset{\cdot}{\ltimes}_{\dot{X}_i,\dot{Y}_i}$, *wenn* $l_{i'}$ *ein positives Literal ist und* $free(l_{i'}) \subseteq v_{in}$,

   (c) $op_i = \overset{\cdot}{\rtimes}_{\dot{X}_i,\dot{Y}_i}$, *wenn* $l_{i'}$ *ein negatives Literal ist,*

   für $Z_{tobind} = \{x_1,\ldots,x_n\}$, *und* $Z_{needed} = v_{out}$.

3. $\mathcal{E}_{v_{in},v_{out}}(\forall\, x_1,\ldots,x_n l_1 \vee \ldots \vee l_l)$

   3.1 $\mathcal{E}_{v_{in},v_{out}}(\forall\, x_1,\ldots,x_n l_1 \vee \ldots \vee l_l) \rightsquigarrow [\![op_1\ l_{1'} \cdots op_l\ l_{l'}]\!]$
   wenn die folgenden Bedingungen gelten:

      (a) $l_{1'},\ldots,l_{l'}$ *ist eine Permutation von* $1,\ldots,l$

      (b) *für jedes Literal gilt:* $v_{in} \subseteq free(l_{i'})$,

      (c) *jedes* $x \in \{x_1,\ldots,x_n\}$ *tritt nur in einem Literal auf*

      (d) $op_i = \overset{\cdot}{\bowtie}_{\dot{X}_i,\dot{Y}_i}$, *wenn* $l_{i'}$ *ein positives Literal ist und* $free(l_{i'}) \subseteq Z_{bound}(i)$,

      (e) $op_i = \overset{\cdot}{\ltimes}_{\dot{X}_i,\dot{Y}_i}$, *wenn* $l_{i'}$ *ein negatives Literal ist,*

      für $Z_{tobind} = \{x_1,\ldots,x_n\}$, *und* $Z_{needed} = v_{out} \cup \bigcup_{i<j\le l} free(l_{j'})$.

   3.2 $\mathcal{E}_{v_{in},v_{out}}(\forall\, x_1,\ldots,x_n l_1 \vee \ldots \vee l_l) \rightsquigarrow [\![op_1\ l_{1'} \cdots op_l\ l_{l'}]\!]^-$
   wenn die folgenden Bedingungen gelten:

      (a) $l_{1'},\ldots,l_{l'}$ *ist eine Permutation von* $1,\ldots,l$

      (b) *für jedes Literal gilt:* $v_{in} \subseteq free(l_{i'})$,

      (c) $op_i = \overset{\cdot}{\bowtie}_{\dot{X}_i,\dot{Y}_i}$, *wenn* $l_{i'}$ *ein positives Literal ist und* $free(l_{i'}) \subseteq Z_{bound}(i)$,

      (d) $op_i = \overset{\cdot}{\ltimes}_{\dot{X}_i,\dot{Y}_i}$, *wenn* $l_{i'}$ *ein negatives Literal ist,*

      für $Z_{tobind} = \{x_1,\ldots,x_n\}$, *und* $Z_{needed} = v_{out} \cup \bigcup_{i<j\le l} free(l_{j'})$.

   3.3 $\mathcal{E}_{v_{in},v_{out}}(\forall\, x_1,\ldots,x_n l_1 \vee \ldots \vee l_l) \rightsquigarrow \overset{\cdot}{\rtimes}_{\dot{X},\dot{Y}}\ (l_{1'}\ op_2\ \cdots op_l\ l_{l'})$
   wenn die folgenden Bedingungen gelten:

(a) $l_{1'}, \ldots, l_{l'}$ *ist eine Permutation von* $1, \ldots, l$.
(b) *mindestens ein Literal ist negativ* $(l_{1'})$,
(c) *für jedes Literal gilt:* $v_{in} \subseteq free(l_{i'})$,
(d) $op_i = \ltimes_{\dot{X}_i, \dot{Y}_i}$, *wenn* $l_{i'}$ *ein positives Literal ist,*
(e) $op_i = \overline{\ltimes}_{\dot{X}_i, \dot{Y}_i}$, *wenn* $l_{i'}$ *ein negatives Literal ist und* $free(l_{i'}) \subseteq Z_{bound}(i)$,
für $Z_{tobind} = \{x_1, \ldots, x_n\}$, *und* $Z_{needed} = v_{out} \cup \bigcup_{i<j\leq l} free(l_{j'})$.

3.4 $\mathcal{E}_{v_{in}, v_{out}}(\forall\, x_1, \ldots, x_n l_1 \vee \ldots \vee l_k \vee l_{k+1} \vee \ldots \vee l_l) \rightsquigarrow$
$\overline{\ltimes}_{\dot{X}_i, \dot{Y}_i}\ ((l_{1'} \overline{\ltimes}_{\dot{X}_i, \dot{Y}_i} \ldots \overline{\ltimes}_{\dot{X}_i, \dot{Y}_i} l'_k \div l'_{k+1}) \cup \ldots \cup (l_{1'} \overline{\ltimes}_{\dot{X}_i, \dot{Y}_i} \ldots \overline{\ltimes}_{\dot{X}_i, \dot{Y}_i} l'_k \div l'_l))$
wenn die folgenden Bedingungen gelten:
(a) $l_{1'}, \ldots, l_{l'}$ *ist eine Permutation von* $1, \ldots, l$.
(b) $l_{1'} \ldots l_k$ *sind positive Literale und* $v_{in} \subseteq free(l_{i'})$ $(1 \leq i \leq k)$.
(c) $l'_{k+1} \ldots l_l$ *sind negative Literale und* $free(l_{i'}) \subset v_{in}$ $(k+1 \leq i \leq l)$.
für $Z_{tobind} = \{x_1, \ldots, x_n\}$, *und* $Z_{needed} = v_{out}$.

mit

$Z_{bound}(i) = v_{in} \cup (\bigcup_{1\leq j<i} free(l_{j'}) \cap Z_{tobind})$,
$Z_{tobind}(i) = (free(l_i) \setminus Z_{bound}(i)) \cap Z_{tobind}$,
$\dot{X}_i = Z_{bound}(i) \cap Z_{needed}$, *und*
$\dot{Y}_i = Z_{tobind}(i) \cap Z_{needed}$.

Beispiel 6.15 (Beispiele dritte Regel Def. 6.14)

- *Fall 1*
  $c = \forall\, x \exists\, y \exists\, z\ p(x) \implies q_1(x,y) \wedge q_2(x,z)$
  $\mathcal{E}_{v_{out}}(c) = \mathcal{E}_{\{x\}}(\sim c)$
  $= p(x)\mathcal{E}_{\{x\}}(\neg q_1(x,y) \vee \neg q_2(x,z))$
  $= p(x)[\![\ltimes_{\{x\},\{\emptyset\}} q_1(x,y) \ltimes_{\{x\},\{\emptyset\}} q_2(x,z)]\!]$
- *Fall 2*
  $\forall\, x \exists\, y\ p(x) \implies q_1(x,y) \wedge q_2(x,y)$
  $\mathcal{E}_{v_{out}}(c) = \mathcal{E}_{\{x\}}(\sim c)$
  $= p(x)\mathcal{E}_{\{x\}}(\neg q_1(x,y) \vee \neg q_2(x,y))$
  $= p(x)[\![\ltimes_{\{x\},\{y\}} q_1(x,y) \ltimes_{\{x\},\{\emptyset\}} q_2(x,y)]\!]^-$
- *Fall 3*
  $\forall\, x \exists\, y \exists\, z\ p(x) \implies q_1(x,y) \wedge q_2(x,z)$
  $\mathcal{E}_{v_{out}}(c) = \mathcal{E}_{\{x\}}(\sim c)$
  $= p(x)\mathcal{E}_{\{x\}}(\neg q_1(x,y) \vee \neg q_2(x,z))$
  $= p(x)\overline{\ltimes}_{\{x\},\{\emptyset\}}(q_1(x,y) \ltimes_{\{x\},\{\emptyset\}} q_2(x,z))$
- *Fall 4*
  $\forall\, x \forall\, y \exists\, z\ r(x,y) \implies t(y,z) \wedge \neg g(x,y,z)$
  $\mathcal{E}_{v_{out}}(c) = \mathcal{E}_{\{x,y\}}(\sim c)$
  $= r(x,y)\mathcal{E}_{\{x,y\}}(\neg t(y,z) \vee g(x,y,z)$
  $= r(x,y)\overline{\ltimes}_{\{x,y\},\{\emptyset\}}(g(x,y,z) \div t(y,z))$

6.4 Vermeidung von Divisionen

Es kann sinnvoll sein, bei Formeln der Bauart:

$Q_1 ... Q_n \forall \{x_1, \ldots, x_n\} f_1 \wedge \ldots \wedge f_m$ die letzten Allquantoren in die negierten Teilformeln über die Konjunktionen hinweg zu ziehen (siehe Regel 1 im Abschnitt 2 Grundlagen). Dann kann bei negierten Teilformeln jeder Allquantor in einen Existenzquantor umgewandelt werden.

Beispiel 6.16 $c = \forall x \exists y \; p(x) \Longrightarrow q(x,y)$
Die Basis-Transformation würde folgenden Ausdruck liefern:

$$\begin{aligned}
\mathcal{E}_{v_{out}}(c) &= \mathcal{E}_{\{x\}}(\sim c) \\
&= \mathcal{E}_{\{x\}}(\exists x \forall y \; p(x) \wedge \neg q(x,y)) \\
&= (p(x) \bowtie_{\{x\},\{y\}} dom_y \overline{\bowtie}_{\{x,y\},\{\emptyset\}} q(x,y)) \div dom_y
\end{aligned}$$

mit $dom_y = \pi_y \, p(x,y)$.

Durch Hineinziehen des Allquantors in die Matrix der Formel vor das Literal q ergibt sich ein wesentlich einfacher auszuwertender algebraischer Ausdruck.

$$\begin{aligned}
\mathcal{E}_{v_{out}}(c) &= \mathcal{E}_{\{x\}}(\sim c) \\
&= \mathcal{E}_{\{x\}}(\exists x \forall y \; p(x) \wedge \neg q(x,y)) \\
&= \mathcal{E}_{\{x\}}(\exists x \; p(x) \wedge \neg \exists y \; q(x,y)) \\
&= p(x) \overline{\bowtie}_{\{x\},\{\emptyset\}} q(x,y)
\end{aligned}$$

Im ersten Fall sind 3 Operationen auszuführen; ein aufwendige Kreuzproduktbildung, ein komplementärer Join und eine ebenfalls aufwendige Division. Im zweiten Fall ist nur ein komplementärer Join zur Auswertung notwendig.

An zwei weitere Beispiele soll das Hineinziehen von Allquantoren in die Formel und damit die Umgehung der Divisionen verdeutlicht werden.

$$\begin{aligned}
f &= \exists x \; \exists y \forall z \; r(x,y) \wedge \neg(s(x,y,z) \wedge g(x,y,z)) \\
&= \exists x \; \exists y \; r(x,y) \wedge \neg(\exists z (s(x,y,z) \wedge g(x,y,z)) \\
\mathcal{E}_{\{x,y\}}(f) &= r(x,y) \overline{\bowtie}_{\{x,y\},\{\emptyset\}} (s(x,y,z) \bowtie_{\{x,y,z\},\{\emptyset\}} g(x,y,z))
\end{aligned}$$

$$\begin{aligned}
f &= \exists x \; \exists y \forall z \; r(x,y) \wedge \neg(s(x,y,z) \wedge \neg g(x,y,z)) \\
&= \exists x \; \exists y \; r(x,y) \wedge \neg(\exists z (s(x,y,z) \wedge \neg g(x,y,z)) \\
\mathcal{E}_{\{x,y\}}(f) &= r(x,y) \overline{\bowtie}_{\{x,y\},\{\emptyset\}} (s(x,y,z) \overline{\bowtie}_{\{x,y,z\},\{\emptyset\}} g(x,y,z))
\end{aligned}$$

Im folgenden Beispiel kann die Division nicht durch obige Vereinfachung umgangen werden.

$$\begin{aligned}
f &= \exists x \; \exists y \forall z \; r(x,y) \wedge \neg(t(y,z) \wedge \neg g(x,y,z)) \\
&= \exists x \; \exists y \; r(x,y) \wedge \neg(\exists z (t(y,z) \wedge \neg g(x,y,z)) \\
&\quad \text{Anwendung } \textbf{cjoin} \text{ nicht möglich, da } x \text{ nicht gebunden !} \\
&= \exists x \; \exists y \; r(x,y) \wedge \forall z \; \neg t(y,z) \vee g(x,y,z)) \\
&= \exists x \; \exists y \; r(x,y) \wedge \forall z \; t(y,z) \Longrightarrow \; g(x,y,z)) \\
\mathcal{E}_{\{x,y\}}(f) &= r(x,y) \bowtie_{\{x,y\},\{\emptyset\}} (g(x,y,z) \div t(y,z))
\end{aligned}$$

7 Auswertung der Bypass-Technik

Ausgangspunkt der Überlegungen zur Entwicklung der Übersetzungstechnik war die Forderung nach effizienten Auswertungsverfahren für Konsistenzbedingungen. Die Bewertung der entwickelten Verfahren kann analytisch (durch ein Kostenmodell), durch Simulation oder durch Messung am realen System erfolgen. Im folgenden wird eine analytische Auswertung vorgenommen. Die Evaluierung erfolgt durch Messungen an einer Hauptspeicherdatenbank. Zum einen soll gezeigt werden, wann sich der Einsatz der im vorherigen Abschnitt vorgestellten Bypass-Technik lohnt und zum anderen wie groß der Effizienzgewinn durch Anwendung der Bypass-Technik ist.

7.1 Beispiele

Die Vorteile der Bypass-Technik sollen anhand zweier typischer Konsistenzbedingungen gezeigt werden, bei denen Disjunktionen auftreten (siehe Abschnitt 6):

1. $c_1 = \forall\, x\ p(x) \wedge (r(x) \vee s(x)) \Longrightarrow\ t(x)$
   (Disjunktion in der Prämisse)

2. $c_2 = \forall\, x\ y\ z\ p(x,y,z) \Longrightarrow\ q_1(x) \wedge\ q_2(y) \wedge\ q_3(z)$
   (Disjunktion in der Konklusion nach der Negierung)

Beispiel 7.1 (Anwendung der Bypass-Technik (1))

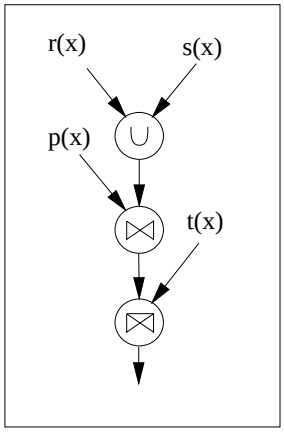
Konsistenzbedingung: $c_1 = \forall\, x\ p(x) \wedge (r(x) \vee s(x)) \Longrightarrow\ t(x)$

Alternative algebraische Ausdrücke:

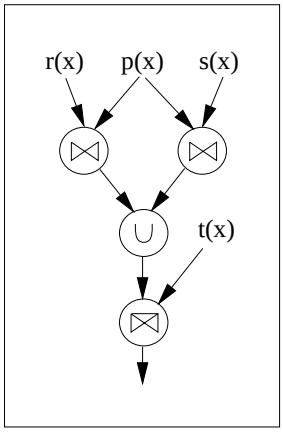
1. $p(x) \bowtie_{\{x\},\{\emptyset\}} (r(x) \cup s(x)) \overline{\bowtie}_{\{x\},\{\emptyset\}} t(x))$
   aus der konjunktiven Normalform (cnf) mit den Basisregeln,

2. $(p(x) \bowtie_{\{x\},\{\emptyset\}} r(x)) \cup (p(x) \bowtie_{\{x\},\{\emptyset\}} s(x)) \overline{\bowtie}_{\{x\},\{\emptyset\}} t(x)$
   aus der disjunktiven Normalform (dnf) mit den Basisregeln,

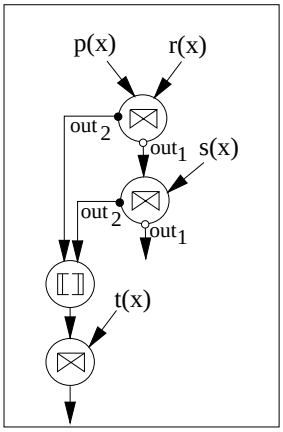
3. $p(x) [\![\overline{\bowtie}_{\{x\},\{\emptyset\}} r(x) \overline{\bowtie}_{\{x\},\{\emptyset\}} s(x)]\!] \overline{\bowtie}_{\{x\},\{\emptyset\}} t(x)$
   mit der Bypass-Technik als Spezialregel.



Ausdruck 1 (nach cnf)



Ausdruck 2 (nach dnf)



Ausdruck 3 (Bypass−Technik)

Abbildung 10: Alternative algebraische Ausdrücke Beispiel 1

Beispiel 7.2 (Anwendung der Bypass-Technik (2))

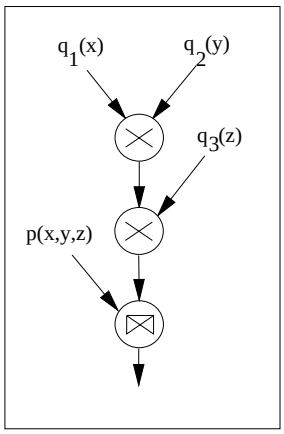
Konsistenzbedingung: $c_2 = \forall\, x\ y\ z\ p(x,y,z) \implies\ q_1(x)\ \wedge\ q_2(y)\ \wedge\ q_3(z)$

Alternative algebraische Ausdrücke:

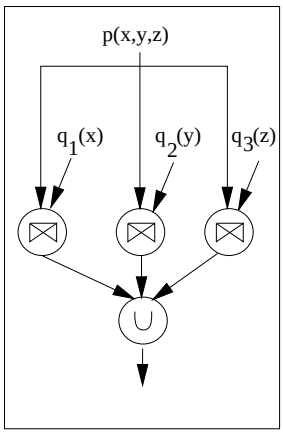
1. $p(x,y,z)\overline{\bowtie}_{\{x,y,z\},\{\emptyset\}}(q_1(x) \bowtie_{\{x\},\{y\}} q_2(y) \bowtie_{\{x,y\},\{z\}} q_3(z))$
   aus der konjunktiven Normalform (cnf) mit den Basisregeln,

2. $(p(x,y,z)\overline{\bowtie}_{\{x,y,z\},\{\emptyset\}}q_1(x)) \cup (p(x,y,z)\overline{\bowtie}_{\{x,y,z\},\{\emptyset\}}q_2(y)) \cup (p(x,y,z)\overline{\bowtie}_{\{x,y,z\},\{\emptyset\}}q_3(z))$
   aus der disjunktiven Normalform (dnf) mit den Basisregeln,

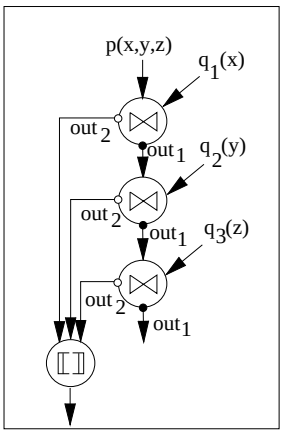
3. $p(x,y,z)[\![\bowtie_{\{x,y,z\},\{\emptyset\}}q_1(x)\bowtie_{\{x,y,z\},\{\emptyset\}}q_2(y)\bowtie_{\{x,y,z\},\{\emptyset\}}q_3(z)]\!]$
   mit der Bypass-Technik als Spezialregel.



Ausdruck 1 (nach cnf)



Ausdruck 2 (nach dnf)



Ausdruck 3 (Bypass−Technik)

Abbildung 11: Alternative algebraische Ausdrücke Beispiel 2

7.2 Auswertung mittels Kostenmodell

Nach einer eher informalen Argumentation im vorigen Abschnitt zur Motivation der Bypass-Technik soll der Vorteil dieser Technik nun mit einem einfachen Kostenmodell quantitativ belegt werden.

7.2.1 Kostenmodell

Das verwendete Kostenmodell basiert auf der Größe der Basis- und Zwischenrelationen und der Berechnungskosten für die Operatoren. Speicherkosten für die Zwischenrelationen werden in diesem Modell nicht berücksichtigt.

Zuerst sollen die Berechnungen der Ergebnisgröße für die verwendeten Operatoren angegeben werden. Dabei spielt die Selektivität von Operatoren [5] neben der Größe der Basisrelationen eine große Rolle. Es wird angenommen, daß dem Optimierer diese statistischen Informationen über die Datenbasis bekannt sind. Die Ergebnisgröße für den verallgemeinerten Join-Operator mit einer Ausgabemenge (gjoin) mit Relationen R und S ist:

$$C^{out}_{R \bowtie S} = |R| \cdot |S| \cdot s_{RS} \tag{1}$$

Für den komplementären Join-Operator (cjoin) berechnet sich die Ergebnisgröße aus:

$$|R| \Leftrightarrow (|R| \cdot |S| \cdot s_{RS}) \tag{2}$$

Für die Vereinigung ergibt sich unter der Annahme disjunkter Relationen:

$$C^{out}_{R \cup S} = |R| + |S| \tag{3}$$

Der verallgemeinerte Join-Operator (GJOIN) erzeugt zwei Ausgabemengen:

$$C^{out}_{R \overline{\bowtie} S} = \left\langle \begin{array}{ll} |R| \cdot |S| \cdot s_{RS} & \text{1. Ausgabemenge} \\ |R| \Leftrightarrow (|R| \cdot |S| \cdot s_{RS}) & \text{2. Ausgabemenge} \end{array} \right. \tag{4}$$

Analog werden die Ergebnisgrößen des komplementären Join-Operators (CJOIN) berechnet, die Ausgabemengen sind dabei vertauscht.

Für die Join-Operatoren wird eine Nested-Loop-Implementierung zugrundegelegt. Damit ergeben sich für die Operatoren verallgemeinerter Join und komplementärer Join mit einer (gjoin, cjoin) und mit zwei (GJOIN, CJOIN) Ausgabemengen folgende Auswertungskosten:

$$C_{R \bowtie S} = |R| \cdot |S| \tag{5}$$

Die Berechnung der zweiten Ausgabemenge des verallgemeinerten bzw. komplementären Join-Operators erfordert keinen zusätzlichen Aufwand. Die Kosten des Vereinigungs-Operators ohne Berücksichtigung der Duplikatelimination sind bestimmt durch:

$$C_{R \cup S} = |R| + |S| \tag{6}$$

[5] Unter der Join-Selektivität s_{RS} wird wie üblich der Anteil der Tupel aus dem Kreuzprodukt von R und S, der das Join-Prädikat erfüllt, verstanden.

Für die Berechnung der Kosten wird zwischen *gegebenen Parametern* wie Kardinalitäten der Basisrelationen und Join-Selektivitäten zwischen Basisrelationen und *abgeleiteten Parametern* wie Join-Selektivitäten von Basisrelationen mit Zwischenrelationen unterschieden. Die Zwischenrelation, welche durch die Auswertung der Prämisse der Konsistenzbedingung entsteht, ist mit $\mathcal{E}(p(x) \wedge (r(x) \vee s(x)))$ bezeichnet. Deren Kardinalität wurde mit einer Konstante vorgegeben, wobei die Attributwerte der Zwischenrelation mit denen aus der Relation t identisch sind und somit die Konsistenz erfüllt ist. Für die Auswertung des 1. algebraischen Ausdrucks aus Beispiel 10 wird die Selektivität zur Berechnung der Ausgabemenge des zweiten Operators benötigt.

$$s_{(r \cup s) \bowtie p} = \frac{|\mathcal{E}(p(x) \wedge (r(x) \vee s(x)))|}{|r(x) \cup s(x)| \cdot |p|} \tag{7}$$

Im dritten algebraischen Ausdruck aus Beispiel 10 wird für die Bestimmung der zweiten Ausgabemenge des zweiten Join-Operators die Selektivität $s_{p \bowtie r \bowtie s}$ benötigt.

$$s_{p \bowtie r \bowtie s} = \frac{|\mathcal{E}(p(x) \wedge (r(x) \vee s(x)))| \Leftrightarrow (|p| \cdot |r| \cdot s_{pr})}{|p| \Leftrightarrow (|p| \cdot |r| \cdot s_{pr}) \cdot |s|} \tag{8}$$

Die nachfolgende Tabelle faßt die Berechnung der Auswertungskosten und Ergebnisgröße für die Operatoren der erweiterten relationalen Algebra zusammen.

Operator	Auswertungskosten C_{RopS}	Ergebnisgröße C^{out}_{RopS}
cartesian product $\times$	$\|R\| \cdot \|S\|$	$\|R\| \cdot \|S\|$
gjoin $\bowtie$	$\|R\| \cdot \|S\|$	$\|R\| \cdot \|S\| \cdot s_{RS}$
cjoin $\overline{\bowtie}$	$\|R\| \cdot \|S\|$	$\|R\| \Leftrightarrow (\|R\| \cdot \|S\| \cdot s_{RS})$
GJOIN $\bowtie$	$\|R\| \cdot \|S\|$	$\|R\| \cdot \|S\| \cdot s_{RS}$ $\|R\| \Leftrightarrow (\|R\| \cdot \|S\| \cdot s_{RS})$
CJOIN $\overline{\bowtie}$	$\|R\| \cdot \|S\|$	$\|R\| \Leftrightarrow (\|R\| \cdot \|S\| \cdot s_{RS})$ $\|R\| \cdot \|S\| \cdot s_{RS}$
union $\cup$	$\|R\| + \|S\|$	$\|R\| + \|S\|$
intersection $\cap$	$\|R\| + \|S\|$	$min(\|R\|, \|S\|)$

Tabelle 5: Auswertungskosten und Ergebnisgröße von XRA-Operatoren

7.2.2 Kostenbetrachtung anhand von Beispielen

Zuerst soll exemplarisch an zwei Beispielen und bestimmten Konstellationen von Parametern gezeigt werden, daß durch die Anwendung der Bypass-Technik ein Effzienzgewinn erreicht werden kann.

Auswertung Beispiel 1

Für die Konsistenzbedingung aus Beispiel 7.1 seien die Kardinalitäten der Basisrelationen mit $|p| = 25, |r| = 10, |s| = 10, |t| = 8$ und Join-Selektivitäten mit $s_{pr} = 0.024$, $s_{ps} = 0.008$ und $s_{(r \cup s)p} = 0.016$ gegeben.

Die Auswertungskosten für die drei algebraischen Ausdrücke aus Abbildung 10 sind: [6]

$$
\begin{aligned}
C_{Ausdruck1} &= C^1_{\cup} + C^2_{\bowtie} + C^3_{\overline{\bowtie}} \\
&= (|r| + |s|) + (|p| \cdot C^{out}_{r \cup s}) + (C^{out}_{p \bowtie (r \cup s)} \cdot |t|) \\
&= (10 + 10) + (25 \cdot 20) + (8 \cdot 8) \\
&= 584 \\
C_{Ausdruck2} &= C^1_{\bowtie} + C^2_{\bowtie} + C^3_{\cup} + C^4_{\overline{\bowtie}} \\
&= (|p| \cdot |r|) + (|p| \cdot |s|) + (C^{out}_{p \bowtie r} + C^{out}_{p \bowtie s}) + (C^{out}_{(p \bowtie r) \cup (p \bowtie s)} \cdot |t|) \\
&= (25 \cdot 10) + (25 \cdot 10) + (6 + 2) + (8 \cdot 8) \\
&= 572 \\
C_{Ausdruck3} &= C^1_{\overline{\bowtie}} + C^2_{\overline{\bowtie}} + C^3_{[]} + C^4_{\overline{\bowtie}} \\
&= (|p| \cdot |r|) + (C^{out^1}_{p \overline{\bowtie} r} \cdot |s|) + (C^{out^2}_{p \overline{\bowtie} r} + C^{out^2}_{p \overline{\bowtie} r \overline{\bowtie} s}) + (C^{out}_{(C^{out^2}_{p \overline{\bowtie} r}) \cup (C^{out^2}_{p \overline{\bowtie} r \overline{\bowtie} s})} \cdot |t|) \\
&= (25 \cdot 10) + (19 \cdot 10) + (6 + 2) + (8 \cdot 8) \\
&= 512
\end{aligned}
$$

Der Effizienzgewinn in diesem Beispiel bei der gewählten Konstellation von Parametern beträgt etwa 10 % gegenüber den mit Basisregeln generierten algebraischen Ausdrücken.

Auswertung Beispiel 2

Für die Konsistenzbedingung aus Beispiel 7.2 seien die Kardinalitäten der Basisrelationen mit $|p| = 100, |q_1| = 10, |q_2| = 10, |q_3| = 10$ und Join-Selektivitäten mit $s_{q_1,q_2} = 1.0$, $s_{q_2,q_3} = 1.0$,$s_{p,q_i} = 0.0$ (i=1,2,3) und $s_{p,q_1 \times q_2 \times q_3} = 0.0$ gegeben.

Die Auswertungskosten für die drei algebraischen Ausdrücke aus Abbildung 11 sind:

$$
\begin{aligned}
C_{Ausdruck1} &= C^1_{\times} + C^2_{\times} + C^3_{\overline{\bowtie}} \\
&= (|q_1| \cdot |q_2|) + (C^{out}_{q_1 \times q_2} \cdot |q_3|) + (|p| \cdot C^{out}_{q_1 \times q_2 \times q_3}) \\
&= (10 \cdot 10) + (100 \cdot 10) + (100 \cdot 100) \\
&= 11100 \\
C_{Ausdruck2} &= C^1_{\overline{\bowtie}} + C^2_{\overline{\bowtie}} + C^3_{\overline{\bowtie}} + C^4_{\cup} \\
&= (|p| \cdot |q_1|) + (|p| \cdot |q_2|) + (|p| \cdot |q_3|) + (C^{out}_{p \overline{\bowtie} q_1} + C^{out}_{p \overline{\bowtie} q_2} + C^{out}_{p \overline{\bowtie} q_3}) \\
&= (100 \cdot 10) + (100 \cdot 10) + (100 \cdot 10) + (0 + 0 + 0) \\
&= 3000 \\
C_{Ausdruck3} &= C^1_{\bowtie} + C^2_{\bowtie} + C^3_{\bowtie} + C^4_{[]} \\
&= (|p| \bowtie |q_1|) + (C^{out^1}_{p \bowtie q_1} \cdot |q_2|) + (C^{out^1}_{p \bowtie q_1 \bowtie q_2} \cdot |q_3|) + (C^{out^2}_{p \bowtie q_1} + C^{out^2}_{p \bowtie q_1 \bowtie q_2} + C^{out^2}_{p \bowtie q_1 \bowtie q_2 \bowtie q_3}) \\
&= (100 \cdot 10) + (100 \cdot 10) + (100 \cdot 10) + (0 + 0 + 0) \\
&= 3000
\end{aligned}
$$

In diesem Beispiel ist bei der angenommenen Parameterkonstellation kein Effizienzgewinn (aber auch kein Verlust) durch die Anwendung der Bypass-Technik im Vergleich zum aus der disjunktiven Normalform entstandenen Anfrageplan entstanden. Sobald aber mindestens ein Tupel die Konsistenzbedingung verletzt, ist die Anwendung der Bypass-Technik von Vorteil, und dieser Vorteil vergrößert sich mit der Anzahl der Tupel, welche die Konsistenzbedingung verletzen.

[6] Zur einfacheren Darstellung werden die Operatoren der jeweiligen Ausdrücke von oben nach unten und von links nach rechts durchnumeriert ($C^1_{\bowtie}$ bezeichnet die Kosten des ersten Operators im Ausdruck).

7.2.3 Allgemeinere Kostenabschätzungen

In einem zweiten Schritt werden nun einige Fälle genauer charakterisiert, bei denen die Anwendung der Bypass-Technik vorteilhaft ist. Es wird dabei das Kostenmodell aus dem vorherigen Abschnitt übernommen und die Konsistenzbedingung aus Beispiel 7.1 verwendet:

$$c_1 = \forall\, x\ p(x)\ \wedge\ (r(x)\ \vee\ s(x))\ \Longrightarrow\ \ t(x)$$

Bei den Kostenbetrachtungen hat sich gezeigt, daß die Anwendung der Bypass-Technik von Vorteil ist, wenn:

- viele Tupel in der Disjunktion auftreten, die nicht zum Ergebnis der Prämisse beitragen,

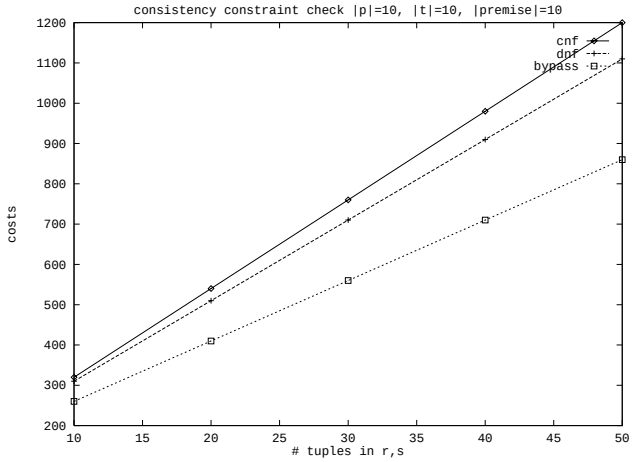


Abbildung 12: Kostenbetrachtung: Disjunktion in Prämisse

Bei dieser Betrachtung wurde die Anzahl der Tupel in den Relationen r und s der Disjunktion in der Prämisse variiert und die Kardinalität der anderen Relationen konstant gehalten. Je mehr die Kardinalität von r und s die Kardinalität der anderen Relationen übersteigt, desto deutlicher wird der Effizienzgewinn bei der Anwendung der Bypass-Technik, da Tupel, die sich schon für das Ergebnis der Prämisse qualifiziert haben, nicht doppelt getestet werden müssen und keine Tupel betrachtet werden, die nichts zum Ergebnis beitragen.

- viele Duplikate in Attributwerten der Relationen in der Disjunktion auftreten.

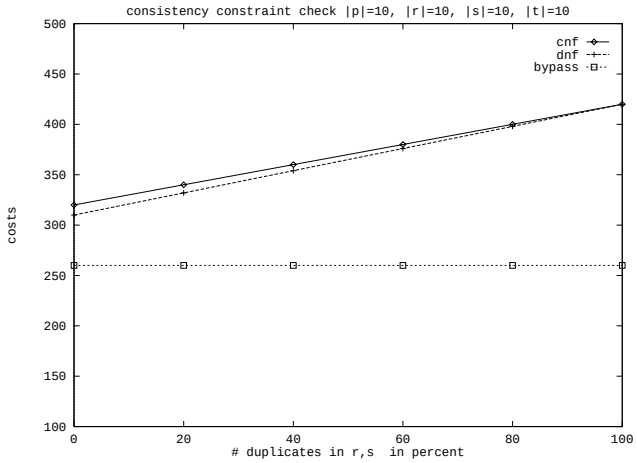


Abbildung 13: Kostenbetrachtung: Duplikate in Disjunktion

In der zweiten Kostenbetrachtung wurde die Anzahl der Duplikate in den Attributwerten der Relationen r und s von 0 % bis 100 % verändert, und es zeigt sich, daß bei Anwendung der Bypass-Technik keine Erhöhung der Kosten durch Duplikate entsteht, hingegen bei der Auswertung der alternativen Auswertungspläne die Kosten steigen.

Bei Verwendung der Bypass-Technik wird der verallgemeinerte Join-Operator verwendet. Der verallgemeinerte Join-Operator ist *nicht kommutativ*, da innerhalb des Operators ein Join *und* ein Anti-Join berechnet werden. Das kann nachteilig für Parameterkonstellationen sein, bei denen die Kardinalität des Ergebnisses vor der Disjunktion sehr viel größer ist als die Kardinalität des Ergebnisses der Disjunktion und die Disjunktion in ihrer Eigenschaft als „Filter“ nahezu keinen Beitrag zum Ergebnis der Prämisse liefert.

7.3 Auswertung durch Messung am System

Nun soll an einem realen System gezeigt werden, wann sich der Einsatz der Bypass-Technik lohnt. Die Messungen wurden auf einer in CLOS implementierten Hauptspeicherdatenbank durchgeführt. Die Join-Operatoren sind als Hash-Joins implementiert. Weiterhin wird bei einer Vereinigung eine automatische Duplikateliminierung durchgeführt.

Auswertung Beispiel 1

Zuerst wurden die Laufzeiten für die Auswertung der Konsistenzbedingung 7.1 gemessen.

Im Ausgangszustand bestand jede Relation aus 1000 Tupeln. Die Tupelwerte wurden zufällig erzeugt. Der Grundbereich jedes Attributes lag zwischen 1 und 1000. Variiert wurde die Anzahl der Tupel in den Relationen der Disjunktion bis zum 5-fachen der Originalgröße der Relationen.

Der Vorteil der Anwendung der Bypass-Technik ist deutlich zu sehen. Die konstanten Laufzeiten bei den aus der disjunktiven Normalform und mittels Bypass-Technik gene-

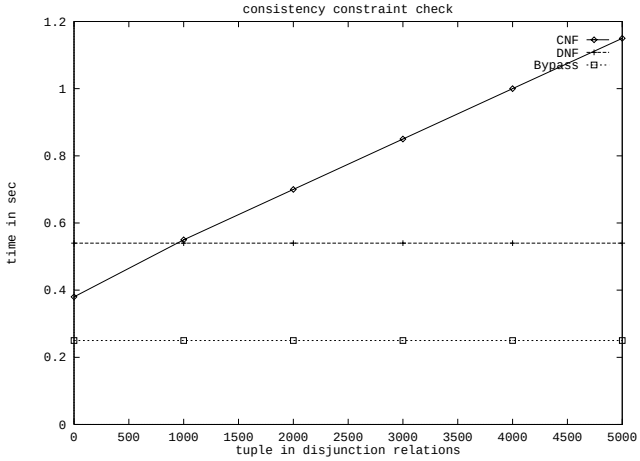


Abbildung 14: Benchmark Beispiel 1

rierten Konsistenztests sind durch die Duplikateliminierung bei der Vereinigungsoperation zu erklären.

Auswertung Beispiel 2

Im der zweiten Meßreihe werden die Laufzeiten für die Auswertung der Konsistenzbedingung 7.2 gezeigt. Die Kurve für den aus der konjunktiven Normalform generierten Konsistenztest ist im Diagramm nicht zu sehen, da schon bei einer Relationengröße von 30 Tupeln mehr als eine Minute für den Test benötigt werden. Das liegt an den kartesischen Produkten im Bearbeitungsplan.

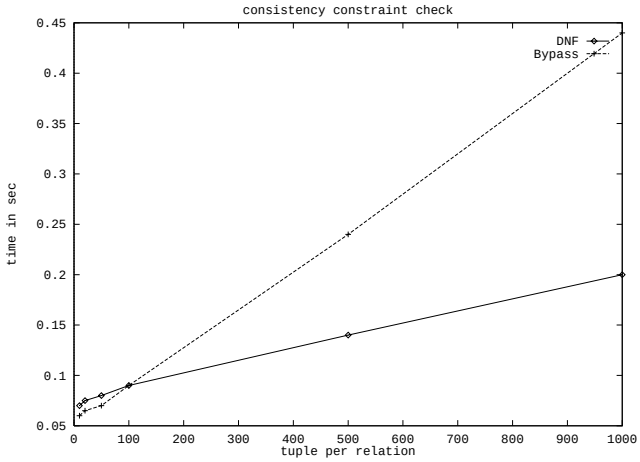


Abbildung 15: Benchmark Beispiel 2

Aus den Diagramm ist ersichtlich, daß die mit der Bypass-Technik generierten Konsistenztests eine höhere Auswertungszeit benötigen, obwohl nach dem Kostenmodell aus Abschnitt 7.2 gegenüber den aus der disjunktiven Normalform generierten Konsistenztests kein höherer Aufwand entsteht. Die Ursache dafür ist darin zu sehen, daß eine Schreiboperation in der vorliegenden Implementierung einen erheblich höheren Aufwand als eine Leseoperation verursacht.

Bisher wurde eine konsistente Datenbasis für den Test angenommen. Nun soll untersucht werden, wie sich die Laufzeitkosten der Konsistenztests verhalten, wenn eine Anzahl von Tupeln der Datenbasis die Konsistenzbedingung verletzen. In den Diagrammen sind die Ergebnisse für den Fall, daß 10%, 20% und 50% der Tupel der Relation p die Konsistenzbedingung verletzen. [7] Die Anzahl der Tupel pro Relation wurde im Bereich von 10 bis 1000 variiert.

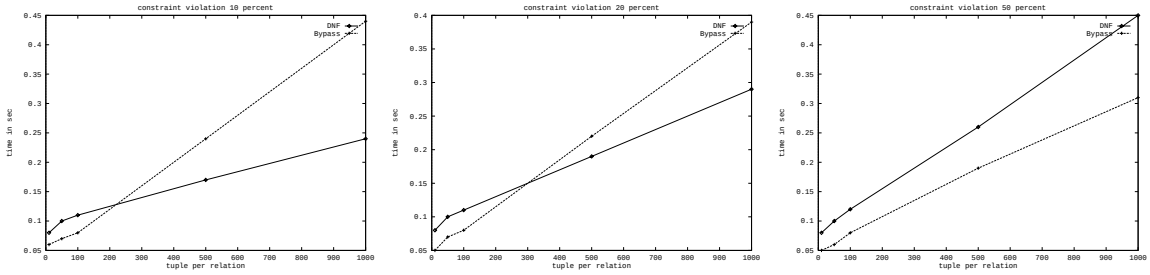


Abbildung 16: Benchmark Konsistenzverletzung

Aus den Kurven ist zu ersehen, daß bis zu 20 % der Tupel, welche die Konsistenzbedingung verletzen, die Bypass-Technik im Nachteil ist; bei 50 % Konsistenzverletzung jedoch die Laufzeit bei Anwendung der Bypass-Technik deutlich geringer ist. Hier zeigt sich, daß mit zunehmender Anzahl der Tupel, welche die Konsistenzbedingung verletzen, der Vorteil des mit der Bypass-Technik generierten Konsistenztests wächst, da nun die Bypass-Technik zum Tragen kommt und der Tupelstrom geteilt und damit getrennt weiterverarbeitet wird. Da in der Regel eine so große Anzahl von Konsistenzverletzungen nicht zu erwarten ist, sollte an dieser Stelle auf die Anwendung der Bypass-Technik verzichtet werden.

Abschließende Bemerkungen zur Auswertung

- Es hat sich sowohl in den Kostenbetrachtungen als auch in den Messungen gezeigt, daß die Anwendung der Bypass-Technik Vorteile bringt, wenn Konsistenzbedingungen ausgewertet werden, die Disjunktionen enthalten. Dabei ist der erreichte Effizienzgewinn abhängig von den konkreten Parameterkonstellationen der Datenbasis, wie z.B. der Häufigkeit von Duplikaten in den Relationen der Disjunktion. Dann kann die Teilung des Tupelstromes durch die Bypass-Technik die Auswertung der Konsistenzbedingung beschleunigen.

- In der vorliegenden Implementierung kann das Erzeugen großer Zwischenrelationen hohe Laufzeitkosten hervorrufen, da zusätzliche Schreiboperationen zur Erzeugung der Zwischenrelation anfallen, die auch kostenintensiver sind als Leseoperationen. Dieser Nachteil kann durch die Implementierung einer Pipeline-Auswertung vermieden werden. Auch die Implementierung einer destruktiven Variante der Join-Operatoren kann diesen Nachteil verringern.

- Das Kostenmodell kann noch verfeinert und erweitert werden. Eine Möglichkeit wäre die Berücksichtigung von Speicherkosten für Zwischenrelationen. Weiterhin kann der

[7] Eine andere Möglichkeit der Darstellung wäre, die Laufzeitkosten über der Anzahl konsistenzverletzender Tupel bei einer konstanten Basisrelationengröße abzutragen.

Zugriff auf Tupel aus den Basisrelationen im Hintergrundspeicher vom Zugriff auf Tupel im Hauptspeicher unterschieden werden, indem Zugriffe auf die Seitenschnittstelle explizit berücksichtigt werden. Wenn dabei von der Pufferverwaltung abgesehen und eine Implementierung der Join-Operatoren als Hash-Joins zugrundegelegt wird, kann pro Zugriff auf die Hashtabelle mit 2 Seitenzugriffen gerechnet werden.

8 Zusammenfassung

Wir haben mit der Übersetzung von deklarativ spezifizierten Konsistenzbedingungen in eine (erweiterte) relationale Algebra die Basis für die Optimierung und damit effiziente Auswertung von Konsistenztests geschaffen. Somit ist die Voraussetzung für eine flexible Handhabung von statischen Konsistenzbedingungen in Datenbanksystemen vorhanden. Dabei wurden bei den betrachteten Klassen von Konsistenzbedingungen keine Einschränkungen hinsichtlich des Datenmodells gemacht.

Die Übersetzung in eine erweiterte relationale Algebra bietet den Vorteil, vorhandene effiziente Algorithmen für Datenbankoperationen zu nutzen. Die verwendete mengenorientierte Auswertung ist für die Verarbeitung großer Datenmengen erforderlich. Die Operationalisierung der Konsistenzbedingungen und die Anwendung der Vereinfachungsmethode [Nic82] zur Definitionszeit bietet die Möglichkeiten weitreichender algebraischer und nichtalgebraischer Optimierungen unter Ausnutzung von statistischem Wissen über die Datenbasis und das Transaktionsprofil. Es wurde gezeigt, daß durch die Einführung neuer Operatoren (verallgemeinerte Join-Operatoren, die Vereinigungsklammer und die Filterklammer) die Effizienz von Konsistenztests verbessert werden kann.

Wir machen keine Annahmen oder Einschränkungen bezüglich der Transaktionen. Auch komplexe Transaktionen, d.h. mehrere Änderungen pro Transaktion, die sich auf verschiedene Prädikate beziehen können, werden zugelassen.

In der aktuellen Arbeit befassen wir uns mit der Optimierung einzelner Konsistenztests als Ausdrücke der erweiterten relationalen Algebra. In Anbetracht der Tatsache, daß von einer Transaktion meist mehr als eine Konsistenzbedingung betroffen ist, von denen wiederum von jeder Konsistenzbedingung mehrere vereinfachte Konsistenzbedingungen betroffen sein können, erscheint eine *globale Optimierung* durch Ausnutzung gemeinsamer Teilausdrücke ein weiterer sinnvoller Optimierungsschritt zu sein.

Da sich die Parallelisierung von relationalen Anfragen in der Vergangenheit als ein erfolgversprechender Weg zur Effizienzsteigerung bei der Anfragebearbeitung gezeigt hat, wird die *Parallelisierung von Konsistenztests* ein weiterer Arbeitsschwerpunkt sein.

Literatur

[AA93] P. Atzeni und V. De Antonellis. *Relational Database Theory.* The Benjamin Cummings Publishing Company, Inc., 1993.

[BDM88] F. Bry, H. Decker und R. Manthey. A uniform approach to constraint satisfaction and constraint satisfiability in deductive databases. In *Proceedings of the Conference Extending Data Base Technology (EDBT'88)*, Venedig, März 1988. LNCS, Springer.

[Bra84] K. Bratbergsengen. Hashing methods and relational algebra operations. In *Proceedings of the 10th Conference on Very Large Data Bases (VLDB)*, Seiten 323–333, Singapore, August 1984.

[Bry89] F. Bry. Towards an efficient evaluation of general queries: Quantifier and disjunction processing revisited. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Seiten 193–204, Portland, Oregon, 1989.

[CG85] S. Ceri und G. Gottlob. Translating SQL into relational algebra: Optimization, semantics, and equivalence of SQL queries. *IEEE Transactions on Software Engineering*, 11(4):324–345, April 1985.

[Cho92] J. Chomicki. History-less checking of dynamic integrity constraints. In F. Golshani (Hrsg.), *8th Int. Conf. on Data Engineering*, Seiten 557–564, Tempe, Arizona, 1992.

[CN93] J. Chomicki und D. Niwinski. On the feasibility of checking temporal integrity constraints. In *Proceedings ACM SIGMOD/SIGACT Conference on Principles of Database Systems (PODS)*, Seiten 202–213, 1993.

[Cod70] E. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.

[Cod79] E.F. Codd. Extending the relational database model to capture more meaning. *ACM Transactions on Database Systems*, 4(4):397–434, Dezember 1979.

[CW90] S. Ceri und J. Widom. Deriving production rules for constraint maintenance. In *Proceedings of the 17th International Conference on Very Large Data Bases*, Seiten 566–577, Brisbane, Australia, 1990.

[Day83] U. Dayal. Processing queries with quantifiers: A horicultural approach. In *Proceedings ACM SIGMOD/SIGACT Conference on Principles of Database Systems (PODS)*, Seiten 125–136, 1983.

[Day87] U. Dayal. Of nests and trees: A unified approach to processing queries that contain nested subqueries, aggregates, and qantifiers. In *Proc. of the 13th VLDB Conference*, Seiten 197–208, 1987.

[FPT92] P. Fraternali, S. Paraboschi und L. Tanca. Automatic rule generation for constraint enforcement in active databases. In U.W. Lipeck und B. Thalheim (Hrsg.), *Proc. 4th Int. Workshop on Foundations of Models and Languages for Data and Objects*, Seiten 153–173, Volkse, Germany, Oktober 1992. Springer, Workshops in Computing.

[GA93] P.W.P.J. Grefen und P.M.G. Apers. Integrity control in relational database systems - An overview. *Data & Knowledge Engineering*, 10(2):187–223, 1993.

[GFA92] P.W.P.J. Grefen, J. Flokstra und P.M.G. Apers. Performance evaluation of integrity control in a parallel main-memory database system. In *Proc. 3nd Int. Conf. on Database and Expert Systems Applications*, Valencia, Spain, 1992.

[Gra93] G. Graefe. Query evaluation techniques for large databases. *ACM Computing Surveys*, 25(2):73–170, Juni 1993.

[JJ91] M. Jeusfeld und M. Jarke. From relational to object-oriented integrity simplification. In *Proc. Second Int. Conf. on Deductive and Object-Oriented databases*, Munich, Germany, 1991.

[JK84] M. Jarke und J. Koch. Query optimization in database systems. *ACM Computing Surveys*, 16(2):111–152, Juni 1984.

[JQ92] H.V. Jagadish und X. Qian. Integrity management in an object-oriented database. In *Proceedings of the 18th VLDB Conference*, Seiten 469–480, Vancouver, British Columbia, Canada, 1992.

[Kar94] D. Karagiannis. *Wissensbasierte Datenbanken*. Oldenbourg-Verlag, 1994.

[KMPS94] A. Kemper, G. Moerkotte, K. Peithner und M. Steinbrunn. Optimizing disjunctive queries with expensive predicates. In M. Winslett (Hrsg.), *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Seiten 336–347, Minneapolis, Minnesota, Mai 1994.

[KSS87] R. Kowalski, F. Sadri und P. Soper. Integrity checking in deductive databases. In P.M. Stocker und W. Kent (Hrsg.), *Proceedings of the 13th International Conference on Very Large Data Bases (VLDB)*, Seiten 61–69, 1987.

[Küc91] V. Küchenhoff. On the efficient computation of the difference between consecutive database states. In C. Delobel, M. Kifer und Y. Masunaga (Hrsg.), *Proceedings DOOD '91*, Seiten 478–502. Springer LNCS 566, 1991.

[LCW93] H. Lu, H.C. Chan und K.K. Wei. A survey on usage of SQL. *SIGMOD RECORD*, 22(4):60–65, Dezember 1993.

[Llo87] J.W. Lloyd. *Foundations of Logic Programming*. Springer, 1987. Chapter 5: Deductive Databases, Section 24: Integrity Constraints.

[LST87] J.W. Lloyd, E.A. Sonenberg und R.W. Topor. Integrity constraint checking in stratified databases. *Journal Logic Programming*, 4:331–343, 1987.

[Mai83] D. Maier. *The Theory of Relational Databases*. Computer Science Press, 1983.

[ME92] P. Mishra und M.H. Eich. Join processing in relational databases. *ACM Computing Surveys*, 24(1):63–113, März 1992.

[MH89] W. McCune und L. Henschen. Maintaining state constraints in relational databases: A proof theoretic basis. *JAM*, 36(1):46–68, Januar 1989.

[MK88] G. Moerkotte und S. Karl. Efficient consistency checking in deductive databases. In *2nd International Conference on Database Theory*, Seiten 118–128, Bruges, Belgium, August 1988.

[ML91] G. Moerkotte und P.C. Lockemann. Reactive consistency control in deductive databases. *ACM Transactions on Database Systems*, 16(4):118–128, Dezember 1991.

[MM93] G. Moerkotte und H. Müller. Exploiting consistency maintenance for planning. In *Fourth International Workshop on the Deductive Approach to Information Systems and Databases*, Barcelona, Spain, September 1993. Universitat Politècnica de Catalunya.

[MZ93] G. Moerkotte und A. Zachmann. Towards more flexible schema management in object bases. In *Proceedings 9th International Conference on Data Engineering ICDE-93*, Vienna, Austria, 1993.

[Nic82] J.M. Nicolas. Logic for improving integrity checking in relational data bases. *Acta Informatica*, 18(3):227–253, 1982.

[NML93] A. Neufeld, G. Moerkotte und P.C. Lockemann. Generating consistent test data: Restricting the search space by a generator formula. *The International Journal on Very Large Data Bases*, 2(2):173–213, 1993.

[Oli91] A. Olive. Integrity constraints checking in deductive databases. In G.M. Lohman, A. Sernades und R. Camps (Hrsg.), *Proceedings of the 17th International Conference on Very Large Data Bases*, Barcelona, September 1991.

[Ple93] D. Plexousakis. Integrity constraint and rule maintenance in temporal deductive knowledge bases. In R. Agrawal, S. Baker und D. Bell (Hrsg.), *Proceedings of the 19th VLDB Conference*, Seiten 146–157, Dublin, Ireland, August 1993.

[RGL90] A. Rosenthal und C. Galindo-Legaria. Query graphs, implementing trees, and freely-reorderable outerjoins. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Seiten 291–299, 1990.

[Sha86] L.D. Shapiro. Join processing in database systems with large main memories. *ACM Transactions on Database Systems*, 11(3):239–264, September 1986.

[Sto75] M. Stonebraker. Implementation of integrity constraints and views by query modification. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Seiten 65–78, 1975.

[SV84] E. Simon und P. Valduriez. Design and implementation of an extendible integrity subsystem. In B. Yormark (Hrsg.), *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Seiten 9–17, Boston, 1984.

[Ull88] J.D. Ullman. *Database and Knowledge-Base Systems*, Band I. Computer Science Press, 1988.

[UO92] T. Urpi und A. Olive. A method for change computation in deductive databases. In L. Yuan (Hrsg.), *Proceedings of the 18th VLDB Conference*, Seiten 225–237, Vancouver, Canada, 1992.

[Wal91] M. Wallace. Compiling integrity checking into update procedures. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1991.

[Wüt91] B. Wüthrich. *Large Deductive Databases with Constraints.* Dissertation, Swiss Federal Institute of Technology Zurich, 1991.