Auswertung von Verbundausdrücken

Komplexität

- Wir betrachten konjunktive Anfragen gegeben als $\pi \bowtie$-Anfragen der relationalen Algebra der Form: $Q = \pi[X](R_1 \bowtie \ldots \bowtie R_n)$.
- Sei $\mathcal{R} = \{R_1, \ldots, R_n\}$ ein Datenbankschema mit einer Instanz $\mathcal{I}$.

Satz

Es ist NP-vollständig zu entscheiden, ob für gegebene $\pi \bowtie$-Anfrage Q über einem Schema $\mathcal{R}$ mit Instanz $\mathcal{I}$ und gegebenem Tupel μ gilt: $\mu \in Q(\mathcal{I})$. Dies gilt sogar für den Fall $\mid Q(\mathcal{I}) \mid \leq 1$.

Beispiel

Betrachte Instanzen zu Schemata $R_i(A_i, A_{i+1})$, $1 \leq i \leq n-1$, und $R_n(A_n, A_1)$ wie folgt:

R_i

A_i	A_{i+1}
0	a
0	b
1	a
1	b
a	0
a	1
b	0
b	1

R_n

A_n	A_1
0	a
0	b
1	a
1	b
a	0
a	1
b	0
b	1

- $\bowtie_{i=1}^{n-1} R_i$ hat exponentiell in n viele Tupel.
- Betrachte $\bowtie_{i=1}^{n} R_i$. Falls n ungerade, so gilt $\bowtie_{i=1}^{n} R_i = \emptyset$; für jedes $j < n$ hat $\bowtie_{i=1}^{j} R_i$ jedoch exponentiell in j viele Tupel.

- Die Berechnung des Verbundes $\bowtie_{i=1}^{n-1} R_i$ ist exponentiell in der Größe der Anfrage und der Eingabe, jedoch polynomiell in der Größe der Anfrage, Eingabe und Ausgabe.
- Die Berechnung von $\bowtie_{i=1}^{n} R_i$ ist exponentiell in der Größe der Anfrage, Eingabe und Ausgabe.

Ziel der Auswertung

Verbundanfragen sollen möglichst so ausgewertet werden, daß keine exponentiell großen Zwischenergebnisse aufgebaut werden, deren Tupel nicht in der Ausgabe erscheinen; d.h., die Anzahl Tupel soll sich während der Auswertung nicht verringern, es sollen keine 'dangling' Tupel auftreten.

Beispiel

Der Fall $n = 2$ ist trivial.
$n = 3$:

A_1	A_2
0	a
0	b
1	a
1	b
a	0
a	1
b	0
b	1

$\bowtie$

A_2	A_3
0	a
0	b
1	a
1	b
a	0
a	1
b	0
b	1

$\bowtie$

A_3	A_1
0	a
0	b
1	a
1	b
a	0
a	1
b	0
b	1

$= \emptyset$

Beispiel

$n = 4$:

A_1	A_2
0	a
0	b
1	a
1	b
a	0
a	1
b	0
b	1

$\bowtie$

A_2	A_3
0	a
0	b
1	a
1	b
a	0
a	1
b	0
b	1

$\bowtie$

A_3	A_4
0	a
0	b
1	a
1	b
a	0
a	1
b	0
b	1

$\bowtie$

A_4	A_1
0	a
0	b
1	a
1	b
a	0
a	1
b	0
b	1

$=$

A_1	A_2	A_3	A_4
0	a	0	a
0	b	0	a
1	a	0	a
1	b	0	a
0	a	0	b
0	b	0	b
1	a	0	b
1	b	0	b
0	a	1	a
0	b	1	a
1	a	1	a
1	b	1	a
0	a	1	b
0	b	1	b
1	a	1	b
1	b	1	b
	...		

Sei $\mathcal{R} = \{R_1, \ldots, R_n\}$ ein Datenbankschema, wobei die Formate X_i der einzelnen Relationsschemata R_i von einander paarweise verschieden sind. Wir interessieren uns für die Berechnung von

$$\bowtie_{i=1}^{n} R_i.$$

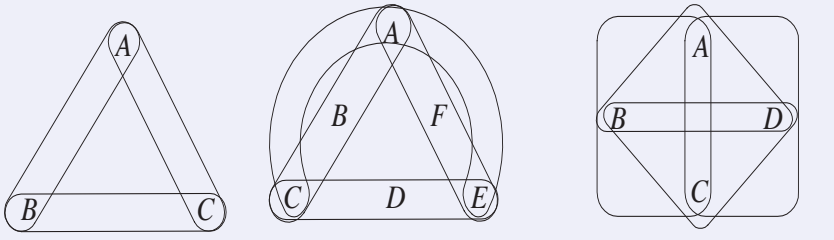
Für den Fall eines azyklischen Verbunds existieren Algorithmen, die polynomiell in der Größe der Anfrage, Eingabe und Ausgabe sind.

Azyklizität kann durch Eigenschaften charakterisiert werden, die jeweils zu einander äquivalent sind.

Eigenschaft 1 basiert auf Hypergraphen

Ein *Hypergraph* ist ein Paar $\mathcal{F} = (V, F)$, wobei V eine Menge von Knoten und F eine Familie von nichtleeren unterschiedlichen Teilmengen von V, den Kanten, bzw. Hyperkanten. Ein Hypergraph heißt *leer*, wenn er die Form $(\emptyset, \emptyset)$ hat.
Der Hypergraph zu $\mathcal{R}$ ist das Paar $(U, \mathcal{R})$, wobei $U = \cup_{R_i \in \mathcal{R}} R_i$.

Beispiel



R1[AB], R2[BC], R3[AC] (a)
S1[ABC], S2[CDE], S3[AFE], S4[ACE] (b)
T1[ABC], T2[BCD], T3[ABD], T4[ACD] (c)

reduzierter Hypergraph

Hypergraph $\mathcal{F}$ enthält kein Kantenpaar f, f', so dass $f \subset f'$. Die *Reduktion* von $\mathcal{F}$ ist ein Paar $(V, F - \{f \in F \mid \exists f' \in F, f \subset f'\})$.

Ohren und Zeugen

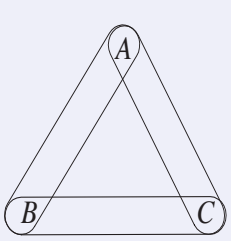
Wir betrachten reduzierte Hypergraphen.

Ein *Ohr* eines Hypergraphen $\mathcal{F} = (V, F)$ ist eine Kante $f \in F$, so dass eine der beiden folgenden Eigenschaften gilt:

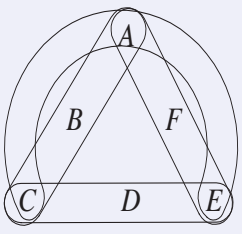
- es existiert eine von f verschiedene Kante $f' \in F$ mit der Eigenschaft:

  $$f \cap (\cup(F - \{f\})) \subseteq f',$$

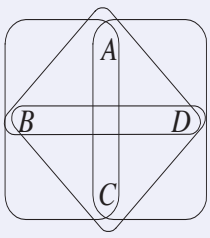
  d.h. alle Attribute in $f \setminus f'$ sind nur Attribute von f.
  f' heißt *Zeuge* dafür, dass f Ohr.
- f ist die einzige Kante in F, oder der Schnitt von f mit allen anderen Kanten in F ist leer.



R1[AB], R2[BC], R3[AC]
(a)



S1[ABC], S2[CDE], S3[AFE], S4[ACE]
(b)



T1[ABC], T2[BCD], T3[ABD], T4[ACD]
(c)

Zu (a), (c) existieren keine Ohren.

In (b) sind $S1, S2, S3$ Ohren mit Zeuge $S4$.

GYO-Algorithmus:[1]

Eingabe: Hypergraph $\mathcal{F} = (V, F)$.
Ausgabe: Hypergraph $\mathcal{F}' = (V', F')$, wobei $V' \subseteq V, F' \subseteq F$.

Do until $\mathcal{F}$ has no ears:

1. Nondeterministically choose an ear f of $\mathcal{F}$.
2. Set $\mathcal{F} := (V', F - \{f\})$, where $V' = \cup(F - \{f\})$.

Eigenschaft 1

Die Ausgabe des GYO-Algorithmus zu $\mathcal{R}$ ist ein leerer Hypergraph.

[1] Graham, Yu, Ozsoyoglu

Eigenschaft 2 basiert auf Verbundbäumen

Ein *Verbund-Baum* zu $\mathcal{R}$ ist ein ungerichteter beschrifteter Baum $T = (\mathcal{R}, E)$, wobei

- Jede Kante (R, R') ist mit der Attributemenge $R \cap R'$ beschriftet.
- Jedes Paar von unterschiedlichen Knoten R, R' hat die folgende Eigenschaft. Wenn $A \in R \cap R'$, dann enthält jede Kante des Pfades von R nach R' Attribut A als Teil ihrer Beschriftung.

Idee: haben R, R' gemeinsame Attribute, dann soll ein Verbundbaum eine Auswertungsreihenfolge vorgeben, die dangling tuples vermeidet.

Eigenschaft 2

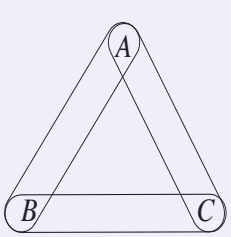
$\mathcal{R}$ hat einen Verbund-Baum.

Beispiel

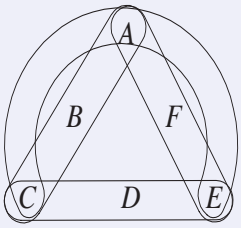
Betrachte die Schemata $R_i(A_i, A_{i+1})$, $1 \leq i \leq n-1$ und $R_n(A_n, A_1)$.

- $\{R_1, \ldots, R_{n-1}\}$ besitzen einen Verbundbaum.
- $\{R_1, \ldots, R_n\}$ besitzen keinen Verbundbaum.

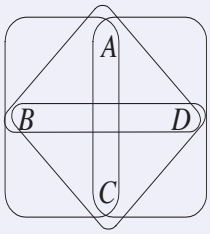
Beispiel



R1[AB], R2[BC], R3[AC]
(a)



S1[ABC], S2[CDE], S3[AFE], S4[ACE]
(b)



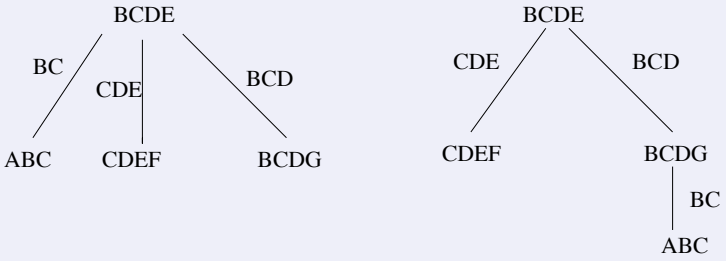
T1[ABC], T2[BCD], T3[ABD], T4[ACD]
(c)

- Zu (a), (c) existiert jeweils kein Verbundbaum.
- (b) hat einen Verbundbaum.
  $\Longrightarrow ((S1 \bowtie S4) \bowtie S2) \bowtie S3$ ist ein geeigneter Verbundausdruck - warum?

Beispiel

Betrachte $\mathcal{R} = \{R_1(ABC), R_2(BCDE), R_3(BCDG), R_4(CDEF)\}$.

- Es existieren zwei Verbundbäume:



- Der GYO-Algorithmus erzeugt einen leeren Hypergraphen. Es können zum Beispiel nacheinander die folgenden Ohren entfernt werden:

$$ABC, CDEF, BCDG, BCDE$$

bzw.

$$ABC, BCDG, CDEF, BCDE$$

Zusammenhang GYO-Algorithmus - Verbundbaum

Angenommen, der GYO-Algorithmus erzeugt einen leeren Hypergraphen.

Wir können dann einen Verbund-Baum $T = (\mathcal{R}, E)$ konstruieren, der genau dann eine Kante (R, R') (R Elter, R' Kind) enthält, wenn vom GYO-Algorithmus ein Ohr R' mit Zeuge R ausgewählt wird.

Eigenschaft 3 basiert auf paarweiser und globaler Konsistenz

Sei $\mathcal{I} = \{I_1, \ldots, I_n\}$ eine Instanz zu $\mathcal{R}$.

- $\mathcal{I}$ heißt *paarweise konsistent*, wenn für jedes Paar $j, k \in [1, n]$ gilt:

$$\pi[R_j](R_j \bowtie R_k)(\mathcal{I}) = I_j.$$

- $\mathcal{I}$ heißt *global konsistent*, wenn für jedes $j \in [1, n]$ gilt:

$$\pi[R_j](\bowtie_{i=1}^{n} R_i)(\mathcal{I}) = I_j.$$

Globale Konsistenz

* schließt *dangling Tupel* aus,
* impliziert paarweise Konsistenz.

Eigenschaft 3

Wenn $\mathcal{I}$ paarweise konsistent, dann ist $\mathcal{I}$ auch global konsistent.

Beispiel

Betrachte Instanzen zu den Schemata $R_i(A_i, A_{i+1})$, $1 \leq i \leq n-1$, und eine Instanz zu $R_n(A_n, A_1)$.

- Sind Instanzen zu $\{R_1, \ldots, R_{n-1}\}$ paarweise konsistent, dann sind sie auch global konsistent.
- Es existieren Instanzen zu $\{R_1, \ldots, R_n\}$, die paarweise, jedoch nicht global konsistent sind.

  Wähle beispielsweise für $n = 4$ und den bereits betrachteten Relationen R_1, R_2, R_3 gerade

$$R_4 = \begin{array}{|cc|} \hline A_4 & A_1 \\ \hline 0 & 0 \\ 1 & 1 \\ a & a \\ b & b \\ \hline \end{array}$$

Eigenschaft 4 beruht auf einem vollen Reduzierer

Seien R, S Relationsschemata. Der *Semiverbund* von R bzgl. S ist:

$$R \ltimes S = \pi[R](R \bowtie S).$$

Mittels eines Semiverbundes werden beim entsprechenden Verbund auftretende dangling Tupel entfernt.
Es gilt:

$$R \bowtie S = (R \ltimes S) \bowtie S = (S \ltimes R) \bowtie R.$$

Definition

- Ein *Semiverbund-Programm* Π zu $\mathcal{R}$ ist eine Folge von Anweisungen:

$$\begin{aligned} R_{i_1} &:= R_{i_1} \ltimes R_{j_1}; \\ R_{i_2} &:= R_{i_2} \ltimes R_{j_2}; \\ &\vdots \\ R_{i_p} &:= R_{i_p} \ltimes R_{j_p}; \end{aligned}$$

- Ein Semiverbund-Programm zu $\mathcal{R}$ heißt *voller Reduzierer*, wenn das Programm zu jeder Instanz $\mathcal{I}$ von $\mathcal{R}$ eine global konsistente Instanz $\mathcal{I}'$ berechnet.

Eigenschaft 4:

$\mathcal{R}$ hat einen vollen Reduzierer.

Beispiel

Betrachte die Schemata $R_i(A_i, A_{i+1})$, $1 \leq i \leq n-1$, und $R_n(A_n, A_1)$.

- $\{R_1, \ldots, R_{n-1}\}$ besitzen einen vollen Reduzierer. Für $n = 3$ ergibt sich

$$\Pi = \begin{array}{l} R_2 := R_2 \ltimes R_1; \\ R_1 := R_1 \ltimes R_2; \end{array}$$

- $\{R_1, \ldots, R_n\}$ besitzen keinen vollen Reduzierer. Für $n = 3$ benötigen wir zur Eliminierung potentieller dangling Tupel z.B. den nicht zulässigen Semiverbund

$$R_1 := R_1 \ltimes (R_2 \bowtie R_3)$$

Beispiel

Sei $\mathbf{R} = \{R_1(ABC), R_2(BCDE), R_3(BCDG), R_4(CDEF)\}$. Betrachte die folgende Instanz $\mathcal{I}$:

R_1

A	B	C
0	3	2
0	1	2
3	1	2
1	1	3

R_2

B	C	D	E
3	2	1	0
1	2	3	0
1	3	1	0

R_3

B	C	D	G
3	2	1	4
1	2	3	2
1	3	1	0
1	3	1	1

R_4

C	D	E	F
2	1	1	4
2	3	0	1
3	1	0	2
3	1	0	3

$\mathcal{I}$ ist weder global konsistent, noch paarweise konsistent.
Ein voller Reduzierer ist wie folgt:

$$\begin{array}{l} R_2 := R_2 \ltimes R_1; \\ R_2 := R_2 \ltimes R_4; \\ R_2 := R_2 \ltimes R_3; \\ R_3 := R_3 \ltimes R_2; \\ R_4 := R_4 \ltimes R_2; \\ R_1 := R_1 \ltimes R_2; \end{array}$$

Satz

Die Eigenschaften (1) bis (4) sind äquivalent.

Wir beschränken uns beim Beweis auf (1) $\Rightarrow$ (4).

Beweis

(1) $\Rightarrow$ (4): Wir definieren induktiv einen Algorithmus, der einen vollen Reduzierer bildet. Wir betrachten hierzu die Schritte des GYO-Algorithmus rückwärts.

- Für den Fall, dass $\mathcal{R}$ aus genau einem Schema R besteht, ist der volle Reduzierer ein Programm ohne Anweisungen.
- Angenommen der GYO-Algorithmus produziert zu dem Hypergraph $\mathcal{F} = (U, \mathcal{R})$ einen leeren Hypergraph als Ausgabe und $\mathcal{R}$ enthält mehr als ein Relationsschema.
- Sei S ein Ohr von $\mathcal{F}$ und sei T Zeuge hierfür. Sei weiter $\mathcal{G} = (U', \mathcal{R} - S)$, wobei $U' = \cup(\mathcal{R} - S)$.
- Induktionsannahme, $\mathcal{G}$ hat einen vollen Reduzierer Π'. Wir zeigen, dass das folgende Semiverbund-Programm Π ein voller Reduzierer für $\mathcal{F}$ ist:

$$\Pi: \qquad T := T \ltimes S;\ \Pi';\ S := S \ltimes T;$$

Beweis fortgesetzt

$$\Pi: \qquad T := T \ltimes S;\ \Pi';\ S := S \ltimes T;$$

- Angenommen es existiert ein dangling Tupel in einer der Relationen **nach** Ausführung von Π.
- S kann kein solches Tupel enthalten, da einmal jedes Tupel in S mit irgendeinem Tupel in T in einer Verbundbeziehung steht. Desweiteren sind alle gemeinsamen Attribute von S und irgendeiner Relation in $\mathcal{G}$ auch bereits in T enthalten und somit durch Π' bereits berücksichtigt.
- Aufgrund der ersten Anweisung kann es auch kein dangling Tupel in einer von S verschiedenen Relation (die bereits in Π' berücksichtigt ist) geben.

Beispiel

Betrachte $R_1 \bowtie R_2 \bowtie R_3$, wobei $R_1 = A_1\ A_2$, $R_2 = A_2\ A_3$ und $R_3 = A_3\ A_4$.

- $\mathcal{R} = \{R_2\}$. $\Pi_1 = \emptyset$.
- $\mathcal{R} = \{R_2, R_3\}$.

$$\Pi_2 = \begin{array}{l} R_2 := R_2 \ltimes R_3; \\ \Pi_1; \\ R_3 := R_3 \ltimes R_2; \end{array}$$

- $\mathcal{R} = \{R_1, R_2, R_3\}$.

$$\Pi_3 = \begin{array}{l} R_2 := R_2 \ltimes R_1; \\ \Pi_2; \\ R_1 := R_1 \ltimes R_2; \end{array} \quad = \quad \begin{array}{l} R_2 := R_2 \ltimes R_1; \\ R_2 := R_2 \ltimes R_3; \\ R_3 := R_3 \ltimes R_2; \\ R_1 := R_1 \ltimes R_2; \end{array}$$

Korollar

Sei ein Datenbankschema $\mathcal{R}$ azyklisch.

Für jede Instanz $\mathcal{I}$ zu $\mathcal{R}$ kann $\bowtie \mathcal{I}$ in polynomieller Zeit in der Größe von $\mathcal{R}$, der Eingabe und der Ausgabe berechnet werden.

Beweis

Der folgende Algorithmus kann verwendet werden.

- Berechne einen vollen Reduzierer und reduziere jede Relation.
- Bilde einen Verbundbaum.
- Betrachte jeden von der Wurzel verschiedenen Knoten, nachdem seine Kinder betrachtet wurden. Sei R ein solcher Knoten und sei S Elternknoten von R. Berechne

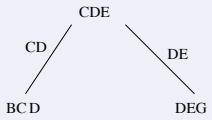
$$S := (S \bowtie R).$$

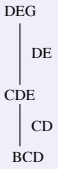
In jedem Berechnungsschritt werden maximal so viele Tupel erzeugt, wie in der Ausgabe vorhanden sind.

Beispiel

Betrachte den Ausdruck $BCD \bowtie CDE \bowtie DEG$.

- Der zugehörige Hypergraph ist azyklisch, da der GYO-Algorithmus einen leeren Hypergraph erzeugen kann. Protokolliere die Schritte des GYO-Algorithmus. Gib alle möglichen Schrittfolgen an, die zu einem leeren Hypergraphen führen.
- Bilde die zugehörigen vollen Reduzierer.
- Es existieren die Verbundbäume:





- Betrachte die paarweise konsistenten Instanzen:

B	C	D
b_1	c_1	d_1
b_1	c_2	d_1

D	E	G
d_1	e_1	g_1
d_1	e_2	g_1

C	D	E
c_1	d_1	e_1
c_2	d_1	e_2

- Leite aus den Verbundbäumen die resultierenden vollen Reduzierer her und wende sie auf obige Instanz an.
- Welche Ausdrücke können aus den Verbundbäumen abgeleitet werden? Werte die Ausdrücke aus und vergleiche mit $(BCD \bowtie DEG) \bowtie CDE$.