Implementierung von Subqueries im SECONDO Optimierer

Burkart Poneleit

3. Januar 2009

1 Einleitung

Beschreibung SECONDO

2 Problembeschreibung

Der SECONDO Optimierer soll um die Fähigkeit zur Übersetzung von geschachtelten Abfragen erweitert werden. Grundlagen zur Übersetzung liefert [5] und [4] Soweit möglich werden Queries mit Subqueries in äquivalente nicht geschachtelte Abfragen überführt werden. Grundsätzlich lassen sich geschachtelte Abfragen mit der 'nested-iteration' Methode ausführen, d.h. die Subquery wird für jedes Tupel der äußeren Abfrage ausgeführt.

2.1 Typ-A Queries

select $A_1, \ldots, A_n$
from $R_1, \ldots, R_m$
where $P_1, \ldots, P_l$
$A_i\theta$(select AGGR($T_i.B$)
from $T_1, \ldots, T_s$
where $Q_1, \ldots, Q_r$)

select$A_1, \ldots, A_n$
from $R_1, \ldots, R_m$
where $P_1, \ldots, P_l$
$A_i\theta C$

C ist die durch Auswertung von select AGGR($T_i.B$) from $T_1, \ldots, T_s$where $Q_1, \ldots, Q_r$ gewonnene Konstante.

2.2 Algorithm NEST-N-J

select $A_1, \ldots, A_n$
from $R_1, \ldots, R_m$
where $P_1, \ldots, P_l$,
$X\theta$(select $T_i.B$
from $T_1, \ldots, T_s$
where $Q_1, \ldots, Q_r$)

select $A_1, \ldots, A_n$
from $R_1, \ldots, R_m, T_1, \ldots, T_s$
where $P_1, \ldots, P_l, Q_1, \ldots, Q_r$
X $\theta' B$

X $\subset \{A_1, \ldots, A_n\}$
$\theta \in$\{IN,NOT IN,$=, \neq, >, \geq, <, \leq$\}

$$\theta' = \begin{cases} = & \text{falls } \theta = \text{IN} \\ \neq & \text{falls } \theta = \text{NOT IN} \\ \theta & \text{sonst} \end{cases}$$

2.3 Algorithm NEST-JA2

select $A_1, \ldots, A_n$
from $R_1, \ldots, R_m$
where $P_1(R_1), \ldots, P_k(R_1), P_1, \ldots, P_l$,
$R_i.X\theta$(select AGGR($T_j.A$)
from $T_1, \ldots, T_s$
where pred[$R_1.Y, T_1.Z$]$, Q_1, \ldots, Q_r$)

let Temp1 = select $R_1.Y$
from R_1
where $P_1(R_1), \ldots, P_k(R_1)$

let Temp2 = select $T_j.A, T_1.Z$
from $T_1, \ldots, T_s$
where $Q_1, \ldots, Q_r$

let Temp3 = Temp1 feed t1
Temp2 feed
outerjoin[pred[$R_1.Y, T_1.Z$]]
sortby[Z asc]
groupby[Z; AggrResult: group AGGR]
consume

select $A_1, \ldots, A_n$
from $R_1, \ldots, R_m, Temp3$
where $P_1, \ldots, P_l$,
$R_i.X \theta Temp3.AggrResult$,
$R_1.Y = Temp3.Z$

$\theta \in \{=, \neq, >, \geq, <, \leq\}$

2.4 Algorithm NEST-D

select A_1
from R
where $P_1, \ldots, P_k$
(select $B_1, \ldots, B_n$
from T
where $B_2 = A_2, \ldots, B_n = A_n$)
op
(select $C_1, \ldots, C_m$
from U
where $C_2 = A_2, \ldots, C_m = A_m$)

let $Temp1 =$ select $C_2, \ldots, C_m$
from U

let $Temp2 =$ select $B_2, \ldots, B_n$
from T

let $Temp3 = Temp2$ feed sort $Temp2$ feed sort mergediff consume

select A_1
from $R, Temp3$
where $P_1, \ldots, P_k$

$Temp3.C_{m+1} = A_{m+1}, \ldots, Temp3.C_n = A_n$

Literatur

[1] Brantner, M., May, N., and Moerkotte, G. Unnesting scalar SQL queries in the presence of disjunction. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on* (Istanbul,, Apr. 2007), pp. 46–55.

[2] Galindo-Legaria, C., and Rosenthal, A. Outerjoin simplification and reordering for query optimization. *ACM Trans. Database Syst. 22*, 1 (1997), 43–74.

[3] Galindo-Legaria, C. A. Outerjoins as disjunctions. In *SIGMOD '94: Proceedings of the 1994 ACM SIGMOD international conference on Management of data* (New York, NY, USA, 1994), ACM, pp. 348–358.

[4] Ganski, R. A., and Wong, H. K. T. Optimization of nested sql queries revisited. *SIGMOD Rec. 16*, 3 (1987), 23–33.

[5] Kim, W. On optimizing an sql-like nested query. *ACM Trans. Database Syst. 7*, 3 (1982), 443–469.

[6] Rosenthal, A., and Galindo-Legaria, C. Query graphs, implementing trees, and freely-reorderable outerjoins. In *SIGMOD '90: Proceedings of the 1990 ACM SIGMOD international conference on Management of data* (New York, NY, USA, 1990), ACM, pp. 291–299.