

Effiziente Erkennung von Leer-Resultat-Abfragen

Matthias Schäfer
Martin-Luther-Universität Halle-Wittenberg

27. März 2007

- ▶ Datenmengen werden größer,
- ▶ leer-Resultat-Abfragen benötigen längere Zeit,
- ▶ leer-Resultat-Abfragen blockieren andere Abfragen.

- ▶ Datenmengen werden größer,
- ▶ leer-Resultat-Abfragen benötigen längere Zeit,
- ▶ leer-Resultat-Abfragen blockieren andere Abfragen.
- ▶ Gefahr von Abfragen mit leerem Resultat wächst,
- ▶ Nutzer möchte nicht auf leere Abfragen warten,
- ▶ leere Abfragen haben i.d.R. keinen Informationsgehalt.

- ▶ IBMs Customer-Relationship-Management-DB lieferte in einem Testzeitraum 18% leere Resultate,
- ▶ in anderen Testszenarien sind es 38%.

- ▶ IBMs Customer-Relationship-Management-DB lieferte in einem Testzeitraum 18% leere Resultate,
- ▶ in anderen Testszenarien sind es 38%.
- ▶ 62% von den 18% sind Wiederholungen anderer Abfragen,
- ▶ i.d.R. sind die übrigen Abfragen zueinander ähnlich.

Trivialer Ansatz:

- ▶ Einige Operatoren verändern die „Leerheit“ einer Datensatzgruppe nicht, z.B. Projektion,
- ▶ wenn solche Operationen die letzten im Auswertungsplan sind, kann abgebrochen werden,

Trivialer Ansatz:

- ▶ Einige Operatoren verändern die „Leerheit“ einer Datensatzgruppe nicht, z.B. Projektion,
- ▶ wenn solche Operationen die letzten im Auswertungsplan sind, kann abgebrochen werden,
- ▶ effektiv, wenn leere Resultate früh im Auswertungsplan entstehen,
- ▶ Problem: die meisten leeren Resultate entstehen durch Verbunde, die eher die letzten Operationen in einem Auswertungsplan darstellen.

Inkonsistente Abfragen:

- ▶ Selektion-Bedingungen werden nach Abfrageoptimierung immer zu falsch ausgewertet,
- ▶ das Resultat ist daher immer leer,

Inkonsistente Abfragen:

- ▶ Selektion-Bedingungen werden nach Abfrageoptimierung immer zu falsch ausgewertet,
- ▶ das Resultat ist daher immer leer,
- ▶ das Leer-Resultat-Problem liegt hier auf der Seite der Nutzereingabe, und ist keine Sache der optimierten Abfrage-Auswertung.

Inkonsistente Abfragen:

- ▶ Selektion-Bedingungen werden nach Abfrageoptimierung immer zu falsch ausgewertet,
- ▶ das Resultat ist daher immer leer,
- ▶ das Leer-Resultat-Problem liegt hier auf der Seite der Nutzereingabe, und ist keine Sache der optimierten Abfrage-Auswertung.

Ausschlussverfahren:

- ▶ Selektion-Bedingungen werden trotz Konsistenz immer zu falsch ausgewertet (ungültige oder nicht verwendete Wertebereiche, müssen gespeichert werden),

Inkonsistente Abfragen:

- ▶ Selektion-Bedingungen werden nach Abfrageoptimierung immer zu falsch ausgewertet,
- ▶ das Resultat ist daher immer leer,
- ▶ das Leer-Resultat-Problem liegt hier auf der Seite der Nutzereingabe, und ist keine Sache der optimierten Abfrage-Auswertung.

Ausschlussverfahren:

- ▶ Selektion-Bedingungen werden trotz Konsistenz immer zu falsch ausgewertet (ungültige oder nicht verwendete Wertebereiche, müssen gespeichert werden),
- ▶ Problem: oft stehen Bedingungen in Relation zu anderen Bedingungen, sodass die Betrachtung einzelner Bedingungen nicht ausreicht.

Definition (Leer-Resultat weiterreichender Operator)

Jeder Operator, der bei Leerheit einer seiner Eingaben eine leere Ausgabe produziert, i.e. Scan, Selektion, Projektion, innerer Verbund, Sortieren, Duplikat-Entfernung.

Definition (Leer-Resultat weiterreichender Operator)

Jeder Operator, der bei Leerheit einer seiner Eingaben eine leere Ausgabe produziert, i.e. Scan, Selektion, Projektion, innerer Verbund, Sortieren, Duplikat-Entfernung.

Definition (Leer-Resultat weiterreichende Abfrage)

Jeder Abfrage, deren Auswertungsplan ausschließlich Leer-Resultat weiterreichende Operatoren enthält.

Definition (Leer-Resultat weiterreichender Operator)

Jeder Operator, der bei Leerheit einer seiner Eingaben eine leere Ausgabe produziert, i.e. Scan, Selektion, Projektion, innerer Verbund, Sortieren, Duplikat-Entfernung.

Definition (Leer-Resultat weiterreichende Abfrage)

Jeder Abfrage, deren Auswertungsplan ausschließlich Leer-Resultat weiterreichende Operatoren enthält.

Definition (Abfrageteil)

Jeder Unterbaum eines als Baum dargestellten Auswertungsplans.

Definition (Atomarer Abfrageteil)

Ein geordnetes Paar, bestehend aus einer Menge von Relationensnamen (Entitäten) und einer Selektion-Bedingung (eine Menge von Vergleichen).

Definition (Atomarer Abfrageteil)

Ein geordnetes Paar, bestehend aus einer Menge von Relationensnamen (Entitäten) und einer Selektion-Bedingung (eine Menge von Vergleichen).

Definition (Hülle einer Selektion-Bedingung)

Eine Selektion-Bedingung, die immer wahr ist, wenn eine andere Selektion-Bedingung wahr ist.

Definition (Atomarer Abfrageteil)

Ein geordnetes Paar, bestehend aus einer Menge von Relationensnamen (Entitäten) und einer Selektion-Bedingung (eine Menge von Vergleichen).

Definition (Hülle einer Selektion-Bedingung)

Eine Selektion-Bedingung, die immer wahr ist, wenn eine andere Selektion-Bedingung wahr ist.

Definition (Hülle eines atomaren Abfrageteils)

Ein atomarer Abfrageteil umhüllt einen anderen, wenn die Menge der Relationsnamen des ersten Abfrageteils eine Untermenge der Relationsnamen des zweiten Abfrageteils sind und die Selektion-Bedingung des ersten Abfrageteils eine Hülle für die Selektion-Bedingung des zweiten Abfrageteils darstellt.

Satz (1)

Wenn jeder Teil einer Abfrage Leer-Resultat weiterreichend ist, dann führt ein leeres Resultat einer Unter-Teilabfrage zu einem leeren Resultat der Teilabfrage der nächsthöheren Ebene.

Satz (1)

Wenn jeder Teil einer Abfrage Leer-Resultat weiterreichend ist, dann führt ein leeres Resultat einer Unter-Teilabfrage zu einem leeren Resultat der Teilabfrage der nächsthöheren Ebene.

Beweis.

Laut Definition enthält eine Leer-Resultat weiterreichende Abfrage ausschließlich Leer-Resultat weiterreichende Operatoren.

Satz (1)

Wenn jeder Teil einer Abfrage Leer-Resultat weiterreichend ist, dann führt ein leeres Resultat einer Unter-Teilabfrage zu einem leeren Resultat der Teilabfrage der nächsthöheren Ebene.

Beweis.

Laut Definition enthält eine Leer-Resultat weiterreichende Abfrage ausschließlich Leer-Resultat weiterreichende Operatoren. Sei nun U , das das leere Resultat einer Teilabfrage ist, Eingabe für einen solchen Operator $O(u_1, u_2, \dots, u_n)$,

Satz (1)

Wenn jeder Teil einer Abfrage Leer-Resultat weiterreichend ist, dann führt ein leeres Resultat einer Unter-Teilabfrage zu einem leeren Resultat der Teilabfrage der nächsthöheren Ebene.

Beweis.

Laut Definition enthält eine Leer-Resultat weiterreichende Abfrage ausschließlich Leer-Resultat weiterreichende Operatoren. Sei nun U , das das leere Resultat einer Teilabfrage ist, Eingabe für einen solchen Operator $O(u_1, u_2, \dots, u_n)$, dann ist laut Definition sichergestellt, dass die Ausgabe $V = O(\dots, U, \dots)$ wiederum leer ist. □

Satz (2)

Wenn P_1 und P_2 atomare Abfrageteile sind und P_1 P_2 umhüllt, dann folgt aus der Leerheit des Resultats von P_1 die Leerheit des Resultats von P_2 .

Beweis.

Seien $U_1 = \sigma_{S_{C1}}(\prod_{R \in R_{N1}} R)$ und $U_2 = \sigma_{S_{C2}}(\prod_{R \in R_{N2}} R)$ die Ergebnisse der beiden Abfrageteile.

Beweis.

Seien $U_1 = \sigma_{S_{C_1}}(\prod_{R \in R_{N_1}} R)$ und $U_2 = \sigma_{S_{C_2}}(\prod_{R \in R_{N_2}} R)$ die Ergebnisse der beiden Abfrageteile.

$$V = \sigma_{S_{C_1}}(\prod_{R \in R_{N_2}} R)$$

Beweis.

Seien $U_1 = \sigma_{S_{C1}}(\prod_{R \in R_{N1}} R)$ und $U_2 = \sigma_{S_{C2}}(\prod_{R \in R_{N2}} R)$ die Ergebnisse der beiden Abfrageteile.

$$\begin{aligned} V &= \sigma_{S_{C1}}(\prod_{R \in R_{N2}} R) \\ &= \sigma_{S_{C1}}(\prod_{R \in R_{N1}} R \times \prod_{R \in R_{N2} \setminus R_{N1}} R) \end{aligned}$$

Beweis.

Seien $U_1 = \sigma_{S_{C1}}(\prod_{R \in R_{N1}} R)$ und $U_2 = \sigma_{S_{C2}}(\prod_{R \in R_{N2}} R)$ die Ergebnisse der beiden Abfrageteile.

$$\begin{aligned} V &= \sigma_{S_{C1}}(\prod_{R \in R_{N2}} R) \\ &= \sigma_{S_{C1}}(\prod_{R \in R_{N1}} R \times \prod_{R \in R_{N2} \setminus R_{N1}} R) \\ &= \sigma_{S_{C1}}((\sigma_{S_{C1}}(\prod_{R \in R_{N1}} R)) \times \prod_{R \in R_{N2} \setminus R_{N1}} R) \end{aligned}$$

Beweis.

Seien $U_1 = \sigma_{S_{C1}}(\prod_{R \in R_{N1}} R)$ und $U_2 = \sigma_{S_{C2}}(\prod_{R \in R_{N2}} R)$ die Ergebnisse der beiden Abfrageteile.

$$\begin{aligned} V &= \sigma_{S_{C1}}(\prod_{R \in R_{N2}} R) \\ &= \sigma_{S_{C1}}(\prod_{R \in R_{N1}} R \times \prod_{R \in R_{N2} \setminus R_{N1}} R) \\ &= \sigma_{S_{C1}}((\sigma_{S_{C1}}(\prod_{R \in R_{N1}} R)) \times \prod_{R \in R_{N2} \setminus R_{N1}} R) \\ &= \sigma_{S_{C1}}(\emptyset \times \prod_{R \in R_{N2} \setminus R_{N1}} R) \end{aligned}$$

Beweis.

Seien $U_1 = \sigma_{S_{C1}}(\prod_{R \in R_{N1}} R)$ und $U_2 = \sigma_{S_{C2}}(\prod_{R \in R_{N2}} R)$ die Ergebnisse der beiden Abfrageteile.

$$\begin{aligned} V &= \sigma_{S_{C1}}(\prod_{R \in R_{N2}} R) \\ &= \sigma_{S_{C1}}(\prod_{R \in R_{N1}} R \times \prod_{R \in R_{N2} \setminus R_{N1}} R) \\ &= \sigma_{S_{C1}}((\sigma_{S_{C1}}(\prod_{R \in R_{N1}} R)) \times \prod_{R \in R_{N2} \setminus R_{N1}} R) \\ &= \sigma_{S_{C1}}(\emptyset \times \prod_{R \in R_{N2} \setminus R_{N1}} R) = \emptyset \end{aligned}$$

Weil $S_{C1} \subseteq S_{C2}$ umhüllt, gilt $U_2 \subseteq V$. Daher ist $U_2 = \emptyset$. □

- ▶ Verfahren ist ein A-posteriori-Algorithmus,
- ▶ basiert auf Auswertungsplänen von Abfragen,
- ▶ DBMS wägt ab, ob die Abfrage unter Zuhilfenahme der Methode, länger benötigt als ohne,
- ▶ beschränkt auf Auswertungspläne, die ausschließlich Leer-Resultate weiterreichende Abfragen enthalten.

- ▶ DBMS hält Liste atomarer Abfrageteile bereit, die als Ergebnis ein leeres Resultat besitzen,
- ▶ vor der Auswertung werden die Informationen der Liste verwendet, um zu entscheiden, ob die Abfrage ein leeres Resultat liefert,
- ▶ nach der Auswertung werden bei leeren Resultaten zusätzliche Informationen in die Liste für spätere Abfragen gespeichert.

- ▶ Aus der Abfrage denjenigen Abfrageteil suchen, der als unterster als erstes ein leeres Resultat liefert,
- ▶ Teilbaum umformen (i.e. auf die wesentlichen Informationen reduzieren),
- ▶ Selektion-Bedingungen umformen (i.e. Form vereinheitlichen),
- ▶ resultierende atomare Abfrageteile speichern.

Baumtransformation:

- ▶ Alle Operationen entfernen, die keinen Einfluss auf die Leerheit einer Abfrage haben, also Projektion, Hashwert-Berechnung, Sortieren und Duplikat-Entfernung.
- ▶ sämtliche physischen Verbunde, also Hash-Verbunde, Merge-Sort-Verbunde und Nested-Loop-Verbunde, werden als eine Art Operation, den logischen Verbund, betrachtet,
- ▶ Index-Scans werden als hintereinander ausgeführte Table-Scans plus Selektionen betrachtet.

Bedingungen-Transformation:

- ▶ Sämtliche Bedingungen einzeln in DNF umschreiben,
- ▶ Negationen werden durch komplementäre Operationen ersetzt,
- ▶ Intervalle werden durch einen Term dargestellt,
- ▶ alle entstandenen Terme werden mit Und verknüpft, der entstehende Term wird wiederum in DNF umgeformt,
- ▶ sämtliche Kombinationen aus den am Teilbaum beteiligten Relationen und den Termen der DNF der Bedingungen bilden atomaren Teilabfragen.

Satz (3)

Folgende Aussagen sind äquivalent:

- 1. Die Ausgabe des ursprünglichen Teilbaums ist leer.*
- 2. Die Ausgabe des transformierten Teilbaums ist leer.*
- 3. Die Ausgabe jedes einzelnen atomaren Abfrageteils ist leer.*

Satz (3)

Folgende Aussagen sind äquivalent:

- 1. Die Ausgabe des ursprünglichen Teilbaums ist leer.*
- 2. Die Ausgabe des transformierten Teilbaums ist leer.*
- 3. Die Ausgabe jedes einzelnen atomaren Abfrageteils ist leer.*

Beweis.

(1.) \leftrightarrow (2.) trivial durch Leerheit-äquivalente Umformungen.

(2.) \leftrightarrow (3.) gilt, weil die Bedingungen der atomaren Abfrageteile per Oder verknüpft sind. Der gesamte Term ist genau dann wahr, wenn mindestens ein Teilterm wahr ist. □

Speichern der atomaren Abfrageteile:

- ▶ Liste von atomaren Abfrageteilen,
- ▶ jeder Eintrag besteht aus der Menge der Relationsnamen und einer verketteten Liste der Bedingungsterme,
- ▶ sämtliche Einträge, die von dem neuen umhüllt werden, werden entfernt.

- ▶ Grundlage ist der Auswertungsplan,
- ▶ atomare Abfrageteile werden wie oben ermittelt,
- ▶ wenn für jeden dieser Abfrageteile einer in der Liste vorhanden ist, der ihn umhüllt, ist das Ergebnis der gesamten Abfrage leer (nach Satz 3).

- ▶ Wurzeloperationen können auch Aggregationsfunktionen sein,
- ▶ bei Vereinigungsoperation und äußerden Verbunden werden die Teilabfragen separat ausgewertet.

- ▶ Verfahren benötigt wenig Speicher,
- ▶ je häufiger eine Abfrage verwendet wird, desto genauer werden die Informationen für die leeren Resultate gespeichert,
- ▶ bei sich ändernden Abfragemustern werden die Informationen für die leeren Resultate sukzessive aktualisiert,
- ▶ Verfahren funktioniert nur in DBen hinreichend optimal, in denen auf Daten lesend zugegriffen werden.

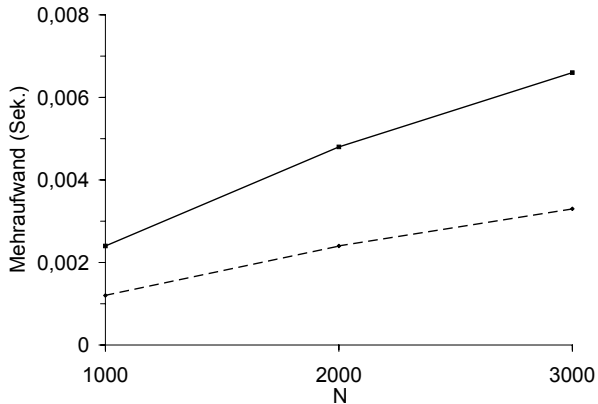
- ▶ Test-Implementierung in PostgreSQL,
- ▶ Test-Schema: Standard TPC-R Benchmark (Verwaltung von Kundenbestellungen),
- ▶ Abfragen wurden so formuliert, dass nur die Wurzel-Operation leere Resultate lieferte.

- ▶ Test-Implementierung in PostgreSQL,
- ▶ Test-Schema: Standard TPC-R Benchmark (Verwaltung von Kundenbestellungen),
- ▶ Abfragen wurden so formuliert, dass nur die Wurzel-Operation leere Resultate lieferte.

Einige benutzte Variablen:

- N Anzahl der atomaren Abfrageteile in der gespeicherten Liste,
- F Kombinationsfaktor, spiegelt die Anzahl von atomaren Abfrageteilen einer Abfrage wider,
- s Größenfaktor der Datenbank.

F und s konstant



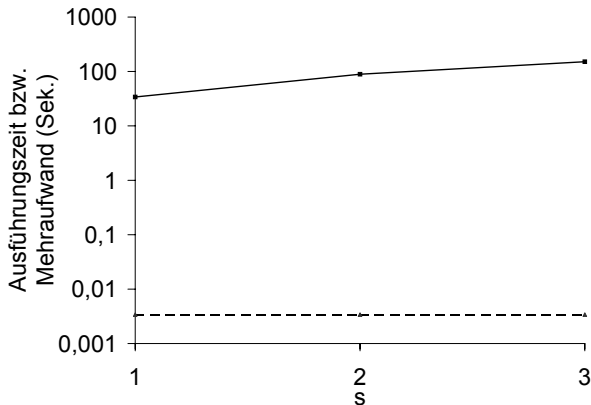
Ergebnisse:

- ▶ Zeit wächst mit N ,
- ▶ erfolglose Tests benötigen doppelten Mehraufwand,

Ergebnisse:

- ▶ Zeit wächst mit N ,
- ▶ erfolglose Tests benötigen doppelten Mehraufwand,
- ▶ Mehraufwand ist etwa linear davon abhängig, über wie viele Attribute eine Selektion durchgeführt wird.

F und N konstant



Ergebnisse:

- ▶ Zeit bleibt mit zunehmendem s nahezu konstant,
- ▶ Mehraufwand ist im Vergleich zur tatsächlichen Ausführung einer Abfrage vernachlässigbar,

Ergebnisse:

- ▶ Zeit bleibt mit zunehmendem s nahezu konstant,
- ▶ Mehraufwand ist im Vergleich zur tatsächlichen Ausführung einer Abfrage vernachlässigbar,
- ▶ Zeiten sind nahezu unabhängig von der Anzahl der abgefragten Attribute.

Fragen