DB-Caching

Henning Klaßen

23. Juli 2004

Gliederung

Was ist DB-Caching?

DB-Caching-Verfahren

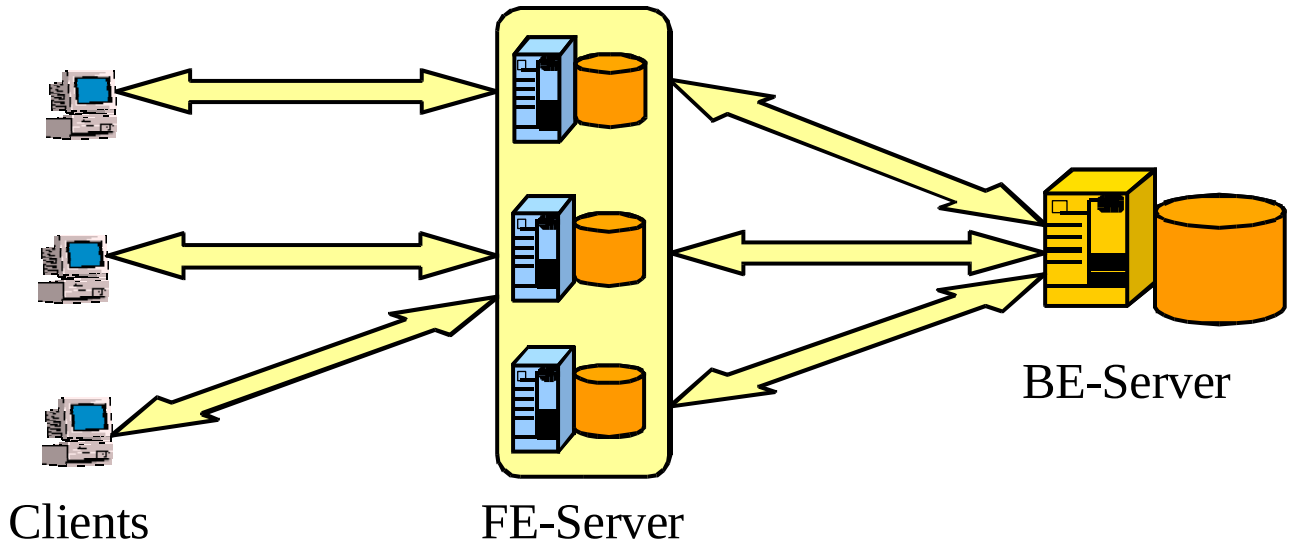
- Materialisierte Sichten
- Full-Table-Caching
- DBProxy
- DBCache

Fazit

Wozu dient DB-Caching?

- Internet: immer mehr dynamische Inhalte
  - Aus Datenbank auf zentralem Back-End-Server (BE-Server)
- Speicherung abgefragter Daten in mehreren Front-End-Servern (FE-Servern)
  - $\Rightarrow$ Verteilung der Anfragelast
  - $\Rightarrow$ Entlastung des BE-Servers
- Anfragebehandlung notwendig
  - Besitzt FE-Server Teilmenge von Daten?
    $\Rightarrow$ Anfrage im Cache auswerten

Beispielarchitektur



Vorteile

- Verteilung der Last auf mehrere FE-Server
- Informationen mit lokaler Bedeutung lokal verfügbar

Was ist beim DB-Caching zu beachten?

Konsistenz

Der FE-Server muss auf eine Anfrage die gleichen Ergebnisse liefern wie auch der BE-Server. Dazu müssen Cache und BE konsistent gehalten werden.

Vollständigkeit

Alle Tupel, die das Anfrageprädikat erfüllen, müssen sich im Cache befinden. Es dürfen z. B. keine Spalten fehlen.

Transparenz

Der Programmierer einer Anwendung soll nichts von der Caching-Logik wissen müssen, sondern wie gewohnt die Anfragen weiter an den BE-Server stellen.

Verfahren zum DB-Caching

- Statische Verfahren: Passen Cache *nicht* anhand der Anfragelast an
  - Materialisierte Sichten
  - Full-Table-Caching
- Dynamische Verfahren: Passen Cache dynamisch anhand der Anfragelast an
  - DBProxy
  - DBCache

Materialisierte Sichten im DB-Caching

- Ausgewählte Inhalte auf FE-DB materialisiert
  - Materialisierung wichtiger Spalten
  - Entscheidung liegt beim DBA
- Überprüfung, ob Anfrage lokal behandelt werden kann
  - Falls möglich: Anfrage an Cache leiten
  - Falls nicht möglich: Anfrage an BE stellen
- Änderungen werden beim BE gesammelt und zur FE-DB propagiert

Vor- und Nachteile materialisierter Sichten

Vorteile

- Erfüllen alle Anforderungen an DB-Caching

Nachteile

- Nicht dynamisch

Full-Table-Caching

- Replizierung des Inhaltes auf die FE-Server
  $\Rightarrow$ Extremfall materialisierter Sichten
- Überprüfung ob Anfragen lokal behandelt werden können
- Änderungen werden beim BE gesammelt:
  FE periodisch aktualisiert

Vor- und Nachteile von Full-Table-Caching

Vorteile

- Forderungen nach **Konsistenz**, **Vollständigkeit** und **Transparenz** gewährleistet
- Einfach zu implementieren

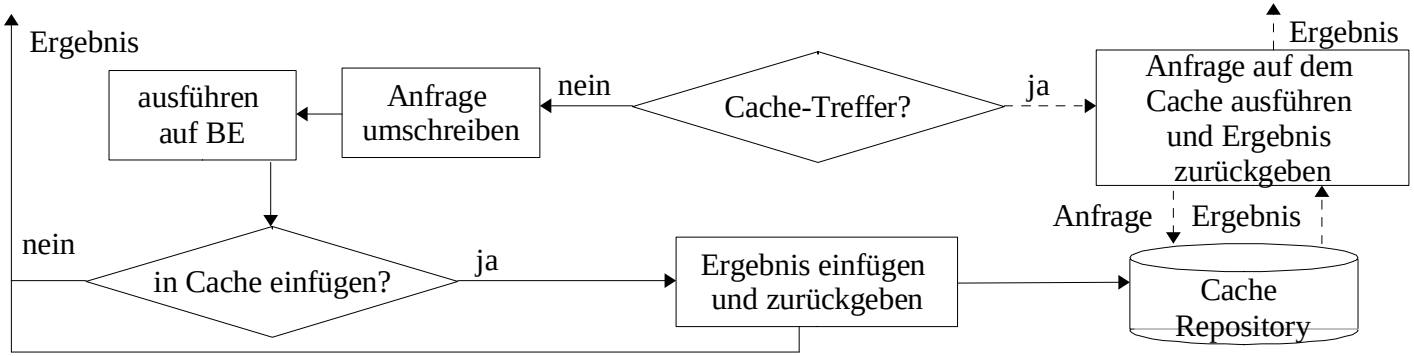
Nachteile

- Leistungsfähige FE-Server nötig
- Geringe Aktualität vs. hohe Netzlast
- Nicht dynamisch

Eigenschaften von DBProxy

- Von IBM entwickelt
- Als JDBC-Treiber implementiert
- Inhalt des Caches wird an Anfragelast angepasst

Abfangen von Anfragen in DBProxy



- Überprüfung, ob Tupel im Cache vorhanden
  - $\Rightarrow$ **Vollständigkeit** gewährleistet
- Wann ist Anfrage Q_B aus dem Cache beantwortbar?
  - Es gibt im Cache eine Anfrage Q_A
  - **where-P(Q_B)** $\Rightarrow$ **where-P(Q_A)**
  - **where-P(Q_B) AND (NOT where-P(Q_A))** nicht erfüllbar
- Ist Anfrage nicht im Cache vorhanden?
  - $\Rightarrow$ Einfügen der Tupel in den Cache
  - $\Rightarrow$ Where-Klausel im Cache speichern
- Anfrage aus dem Cache beantwortbar?
  - Where-Klauseln im Cache ablegen

Beispiel zum Abfangen von Anfragen

- where-P(Q_A) = (Gehalt $>$ 200k)
  where-P(Q_B) = (Gehalt $<$ 150k)
- (Gehalt $<$ 150k) AND (NOT Gehalt $>$ 200k) darf nicht erfüllbar sein
- Q_B nicht im Cache, da Gehalt = 100k mögliche Lösung

Einfügen von Tupeln aus dem BE-Server

- Tupel aus BE-Server werden bei „Cache Miss“ in FE eingefügt
  - ⇒ Anpassen des Caches an Anfragelast
  - ⇒ Gleiches Verhalten wie ein „echter“ Cache
- Erweiterung der Anfragen um Primärschlüssel
  - ⇒ Identifizierung der Tupel im Cache stets eindeutig
- Tabellen im Cache entsprechend SELECT-Klausel der Anfragen erstellen
  - ⇒ Tupel aus einer Tabelle wieder in einer Tabelle speichern

Verhalten bei mehreren Anfragen

- Anfragen über gleiche Tabelle in einer Tabelle gespeichert
  - $\Rightarrow$ Benachbarte Speicherung von Tupeln
  - $\Rightarrow$ Vermeidung von Redundanzen und Inkonsistenzen
- Bei abweichenden SELECT-Klauseln
  - Hinzufügen zusätzlicher Spalten zur Tabelle
- Falsche NULL-Werte für schon existente Tupel
  - $\Rightarrow$ Schränken Konsistenz nicht ein

Einfügen von Tupeln in DBProxy

Anfrage 1 (A1):
SELECT Gehalt
FROM Personal
WHERE Abt BETWEEN 4 AND 5

Anfrage 2 (A2):
SELECT Name, Gehalt
FROM Personal
WHERE Gehalt $>$ 100k
AND Abt $<$ 5

Personal auf dem BE

Id	Name	Gehalt	Abt
1	Hans Meiser	100k	4
2	Klaus Wald	200k	4
3	Peter Baum	100k	5
4	Manfred Vogel	150k	6
5	Oskar Grün	200k	3

Personal auf dem FE

Id	Name	Gehalt	Durch

Tupel aus A1 in den Cache einfügen

Anfrage 1 (A1):
SELECT Id, Gehalt
FROM Personal
WHERE Abt BETWEEN 4 AND 5

Personal auf dem BE

Id	Name	Gehalt	Abt
1	Hans Meiser	100k	4
2	Klaus Wald	200k	4
3	Peter Baum	100k	5
4	Manfred Vogel	150k	6
5	Oskar Grün	200k	3

Personal auf dem FE

Id	Name	Gehalt	Durch

Tupel aus A1 in den Cache einfügen

Anfrage 1 (A1):

```
SELECT Id, Gehalt
FROM Personal
WHERE Abt BETWEEN 4 AND 5
```

Personal auf dem BE

Id	Name	Gehalt	Abt
1	Hans Meiser	100k	4
2	Klaus Wald	200k	4
3	Peter Baum	100k	5
4	Manfred Vogel	150k	6
5	Oskar Grün	200k	3

Personal auf dem FE

Id	Name	Gehalt	Durch
1	*null*	100k	**A1**
2	*null*	200k	**A1**
3	*null*	100k	**A1**

Tupel aus A2 in den Cache einfügen

Anfrage 2 (A2):
SELECT Id, Name, Gehalt
FROM Personal
WHERE Gehalt $>$ 100k
AND Abt $<$ 5

Personal auf dem BE

Id	Name	Gehalt	Abt
1	Hans Meiser	100k	4
2	Klaus Wald	200k	4
3	Peter Baum	100k	5
4	Manfred Vogel	150k	6
5	Oskar Grün	200k	3

Personal auf dem FE

Id	Name	Gehalt	Durch
1	*null*	100k	**A1**
2	*null*	200k	**A1**
3	*null*	100k	**A1**

Tupel aus A2 in den Cache einfügen

Anfrage 2 (A2):
SELECT Id, Name, Gehalt
FROM Personal
WHERE Gehalt $>$ 100k
AND Abt $<$ 5

Personal auf dem BE

Id	Name	Gehalt	Abt
1	Hans Meiser	100k	4
2	Klaus Wald	200k	4
3	Peter Baum	100k	5
4	Manfred Vogel	150k	6
5	Oskar Grün	200k	3

Personal auf dem FE

Id	Name	Gehalt	Durch
1	*null*	100k	**A1**
2	Klaus Wald	200k	**A1, A2**
3	*null*	100k	**A1**
5	Oskar Grün	200k	**A2**

Konsistent-Haltung des Caches

- Cache-Tabellen melden sich für entsprechende Update-, Delete- und Insert-Operationen (UDIs) beim BE-Server an
- UDIs vom BE-Server gesammelt
  - Weiterleiten an angemeldete Cache-Tabellen
- Zustand der Cache-Tabellen entspricht Zustand der BE-Tabellen vor d Zeiteinheiten.

  $\Rightarrow$ **Konsistenz** eingehalten

Vor- und Nachteile von DBProxy

Vorteile

- Dynamisch
- Nur jeweils benötigte Tupel im Cache

Nachteile

- Kein Pre-Caching
- Anfragen entweder aus Cache oder aus BE beantwortet
  - $\Rightarrow$ Alles-oder-nichts-Prinzip

Eigenschaften von DBCache

- Von IBM entwickelt
- Erweiterung der DB2
  - Nutzung materialisierter Sichten
  - Erweiterung um Cache-Tabellen, Cache Constraints
- Inhalt des Caches an Anfragelast angepasst

Cache-Tabellen

- Gleichen im Schema den Originaltabellen
- Beziehen sich auf „Spitznamen“ der Originaltabelle
- Zwei Arten von Cache-Tabellen:
  - Deklarative Cache-Tabellen: initial gefüllt
  - Dynamische Cache-Tabellen: dynamisch gefüllt

Cache Constraints

- Gewährleisten **Vollständigkeit**
- Cache Keys:
  - Dienen der Befüllung der Cache-Tabellen und als Einstiegspunkt für Anfragen
  - Bereichsvollständigkeit muss erfüllt sein
- Referential Cache Constraints:
  - Beziehung zwischen zwei Spalten, zwischen denen ein Join möglich sein soll
  - Modelliert eine Art Vater–Sohn-Beziehung
  - Für alle Werte der Vater-Spalte muss die Sohn-Tabelle alle Zeilen mit dem gleichen Wert in der Sohn-Spalte besitzen.

Beispiel für Cache Constraints

Personal auf BE

Id	Name	Abt	Gehalt
1	Klaus Schneider	2	200k
2	Peter Müller	2	250k
3	Hans Schmied	2	200k
4	Werner Bauer	3	200k
5	Michael Seiler	1	225k
6	Robert Weber	5	75k

Abteilung auf BE

Id	Bezeichnung
1	Marketing
2	Verwaltung
3	Vertrieb
4	Fertigung
5	Lager

Cachen der Person mit der Id 1 (Klaus Schneider).
Cache Key auf Spalte Id und Gehalt.
Referential Cache Constraint von Personal.Abt auf Abteilung.Id

Personal auf FE

Id	Name	Abt	Gehalt

Abteilung auf FE

Id	Bezeichnung
1	Marketing

Beispiel für Cache Constraints

Personal auf BE

Id	Name	Abt	Gehalt
1	Klaus Schneider	2	200k
2	Peter Müller	2	250k
3	Hans Schmied	2	200k
4	Werner Bauer	3	200k
5	Michael Seiler	1	225k
6	Robert Weber	5	75k

Abteilung auf BE

Id	Bezeichnung
1	Marketing
2	Verwaltung
3	Vertrieb
4	Fertigung
5	Lager

Cachen der Person mit der Id 1 (Klaus Schneider).
Cache Key auf Spalte Id und Gehalt.
Referential Cache Constraint von Personal.Abt auf Abteilung.Id

Personal auf FE

Id	Name	Abt	Gehalt
1	Klaus Schneider	2	200k

Abteilung auf FE

Id	Bezeichnung
1	Marketing

Beispiel für Cache Constraints

Personal auf BE

Id	Name	Abt	Gehalt
1	Klaus Schneider	2	200k
2	Peter Müller	2	250k
3	Hans Schmied	2	200k
4	Werner Bauer	3	200k
5	Michael Seiler	1	225k
6	Robert Weber	5	75k

Abteilung auf BE

Id	Bezeichnung
1	Marketing
2	Verwaltung
3	Vertrieb
4	Fertigung
5	Lager

Cachen der Person mit der Id 1 (Klaus Schneider).
Cache Key auf Spalte Id und Gehalt.
Referential Cache Constraint von Personal.Abt auf Abteilung.Id

Personal auf FE

Id	Name	Abt	Gehalt
1	Klaus Schneider	2	200k
3	Hans Schmied	2	200k
4	Werner Bauer	3	200k

Abteilung auf FE

Id	Bezeichnung
1	Marketing

Beispiel für Cache Constraints

Personal auf BE

Id	Name	Abt	Gehalt
1	Klaus Schneider	2	200k
2	Peter Müller	2	250k
3	Hans Schmied	2	200k
4	Werner Bauer	3	200k
5	Michael Seiler	1	225k
6	Robert Weber	5	75k

Abteilung auf BE

Id	Bezeichnung
1	Marketing
2	Verwaltung
3	Vertrieb
4	Fertigung
5	Lager

Cachen der Person mit der Id 1 (Klaus Schneider).
Cache Key auf Spalte Id und Gehalt.
Referential Cache Constraint von Personal.Abt auf Abteilung.Id

Personal auf FE

Id	Name	Abt	Gehalt
1	Klaus Schneider	2	200k
3	Hans Schmied	2	200k
4	Werner Bauer	3	200k

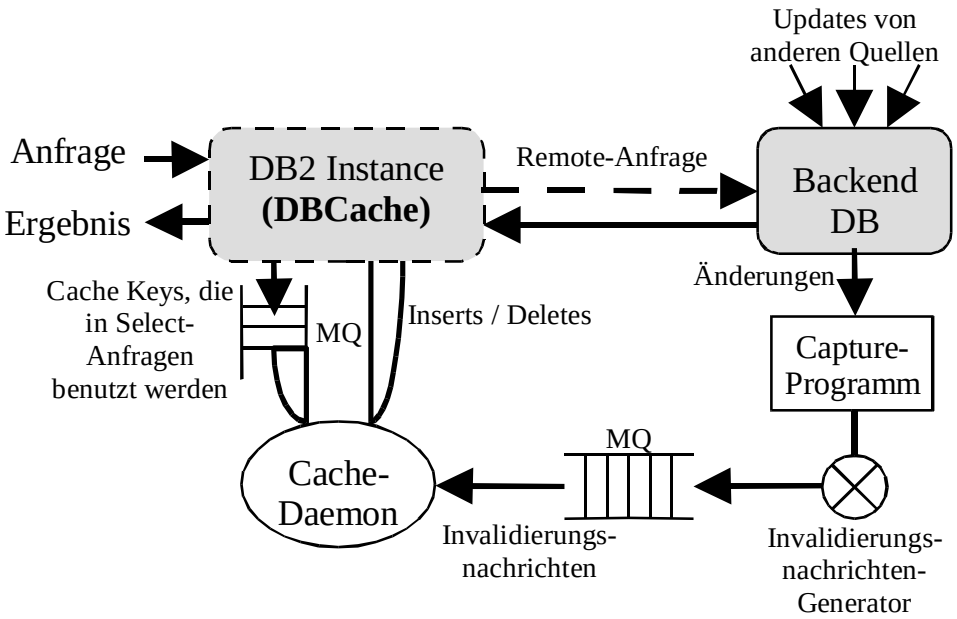
Abteilung auf FE

Id	Bezeichnung
1	Marketing
2	Verwaltung
3	Vertrieb

Anfragen in DBCache

- Deklarative Cache-Tabellen
  - Mechanismus der materialisierten Sichten
- Dynamische Cache-Tabellen
  - Erstellen eines „Janus-Plans“:
    – lokaler Plan für Cache und BE
    – entfernter Plan für BE
    $\Rightarrow$ Gewährleistet Vollständigkeit
  - Falls einige Tupel im Cache: lokaler Plan, sonst entfernter Plan

Wartung des Caches



- UDIs werden gesammelt
- Einfüge- und Aktualisierungs-Operationen werden in Vater–Sohn-Reihenfolge unter Beachtung der Cache Constraint durchgeführt
- Lösch-Operationen werden unter Beachtung der Cache Constraints durchgeführt
- Eine Transaktion

Vor- und Nachteile von DBCache

Vorteile

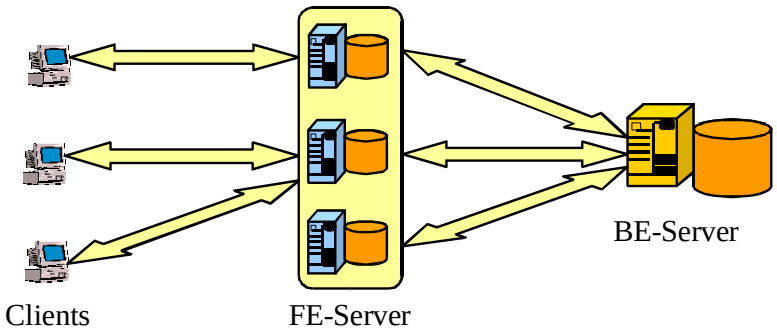
- Dynamisch
- Pre-Caching
- Anfragen können teilweise vom FE-Server und vom BE-Server beantwortet werden
- Strukturierter Cache-Inhalt

Nachteile

- Unter Umständen schlechtes Einlagerungsverhalten
- Bereichsvollständigkeit restriktiv

Fazit

- Statische Verfahren ungeeignet für ein möglichst effektives DB-Caching
- Steigender Stellenwert dynamischer Webseiten
  $\Rightarrow$ Forschung im DB-Caching wichtig
- Verbesserungen bestehender Verfahren zu erwarten
  Konkret: Ersetzung der Bereichsvollständigkeit durch Wertvollständigkeit



Fragen?