Implementierung von Datenbanksystemen

Braindump der Klausur

19. Februar 2014

Die vorliegende Niederschrift wurde nach bestem Wissen und Gewissen erstellt. Für ihre Richtigkeit und Vollständigkeit kann natürlich trotzdem keinerlei Garantie übernommen werden.

110%-Klausur, 99 Punkte gesamt, 90 Minuten.

1 Wissensfragen — 14 Punkte

1.1 Definition — 1,5 Punkte

Wann ist eine Anwendung datenunabhängig?

1.2 Richtig oder falsch — 7 Punkte

Angeben, ob Behauptungen richtig oder falsch mit Begründung:

- Datenbank ist nur für Ablage von Daten da, nicht zum Zugriff
- TID bleibt gleich, solange der Satz existiert
- Seite im Puffer muss dieselbe Größe haben wie Seite im Betriebssystem
- Mit dem TID-Konzept kommt es zu höchstens einer Indirektion beim Zugriff auf nicht fragmentierte Sätze
- Datenbanksystem kann die Blockgröße frei wählen
- Hashverbund kann nur gemacht werden, wenn ein Hashindex existiert
- Primärorganisation einer Relation ist nach dem Primärschlüssel sortiert

1.3 Schichtenmodell 5,5 Punkte

Gegebene Grafik der Schichten eines Datenbanksystems und ihrer Schnittstellen beschriften (vgl. Vorlesungsfolien 1-14, 7-4 und 10-10; allerdings mit fünf oder sechs Schichten). Zu ergänzen waren Name und Aufgaben jeder Schicht sowie Schnittstellen.

2 Puffer

2.1 Indirekte Seitenzuordnung

Welche Hilfsstrukturen braucht man für indirekte Seitenzuordnung und was wird darin gespeichert?

2.2 Verdrängung

Welche Seite wird bei LFU / LRU / FIFO verdrängt?

2.3 Nachteil

Welchen Nachteil hat LFU bei Direktzugriff?

2.4 Belady-Optimalität

Wie muss eine Seitenersetzungsstrategie sein, damit sie Belady-optimal ist? Was muss man dafür wissen?

3 B*-Baum 8 Punkte

Mehrere Key-Value-Paare in leeren B*-Baum einfügen, Baum nach jeder Strukturänderung neu zeichnen. Nachbarknoten sollten bei Überlauf nicht gemischt werden. $k = 1$ für Blätter und $k^* = 2$ für innere Knoten.

Einzufügende Tupel: (1,i), (2,d), (3,b), (5,e), (6,b), (4,u)

4 Indizes

10 Punkte

4.1 Lineares Hashing

7 Punkte

Lineares Hashing durchführen mit Split immer dann, wenn ein Überlauf auftritt. Mit Überlaufbuckets, Bucketgröße 2, Hashfunktion $f(x) = x \bmod (2 \cdot 2^j)$, Beginn mit zwei Buckets ($j = 0$).

In vier Teilaufgaben waren verschiedene Zustände der Hashmap geben, in die man jeweils einen oder mehrere Werte eintragen sollte. Außerdem war anzugeben, welche Werte für j vor und nach dem Einfügen in Verwendung sind, und der Positionszeiger anzupassen.

Alle benötigten Tabellen waren schon vorgezeichnet, allerdings immer mit mehr Feldern als benötigt.

4.2 Bitmap-Index

3 Punkte

Bitmap-Index nach Attribut „Geschlecht“ angeben für:

PNr	Name	Geschlecht
5	Kurt	m
21	Eloise	w
42	Manni	m
2	Carl	m
30	Albert	m
88	Ulf	m
59	Manuela	w
100	Abelad	m

(Fiktive Tabelle, aber die Größe kommt hin.)

5 Speicherung

5.1 Variable Felder

Speicherung eines solchen Satzes schematisch darstellen bei Feldern variabler Länge mit Pointern:

Personalnummer	Fest
Vorname	Variabel
Nachname	Variabel
Abteilung	Fest

5.2 C-Store

Grundsätzlicher Aufbau von C-Store und Unterschied zur üblichen Speicherung in relationalen Datenbanken.

5.3 Arten der Komprimierung bei C-Store

Tabelle befüllen:

	Wenig verschiedene Werte	Viele verschiedene Werte
Sortiert		
Unsortiert		

5.4 Zusammensetzen

Wie kann man die Tupel in C-Store wieder zusammensetzen?

5.5 Min / Max

Wie werden Min- / Max-Anfragen in C-Store gemacht? Wann und warum ist das effizient?

6 Anfrageoptimierung

6.1 Anfragebaum

Unoptimierten Anfragebaum erstellen für diese Anfrage:

```
SELECT
    ST.x, R.z
FROM
    R, (
        SELECT (S.a, T.e)
        FROM S JOIN T ON S.d = T.e
        WHERE S.b > T.f
    ) as ST
WHERE
    ST.x = R.x AND (R.a = 42 OR R.a = 21);
```

6.2 Optimierung

Zwei Möglichkeiten zur Optimierung des obigen Baums nennen.

6.3 Operatoren

Unterschied logischer Operator zu Planoperator?

6.4 Sort-Merge-Verbund

Wie funktioniert Sort-Merge-Join ohne Index?

6.5 Laufzeitverhalten

Laufzeitverhalten des Sort-Merge-Verbunds ohne Index in O-Notation.

6.6 Grenztrefferrate

Wie wirkt sich die Grenztrefferrate auf die Wahl der Planoperatoren aus? Warum?

7 Recovery 6 Punkte

7.1 Konsistenz 2 Punkte

Unterschied zwischen logischer und physischer Konsistenz?

7.2 Durchführung 4 Punkte

Die drei Schritte bei Recovery angeben. Wie laufen sie ab und welche Operationen finden jeweils statt?

8 Transaktionen

8.1 Atomarität

Warum muss eine Transaktion atomar sein?

8.2 Abhängikeitsgraph

Abhängigkeitsgraph zeichnen zu folgendem Ablauf:

r2(A), r2(B), r3(B), w3(A), r1(A), w1(A), c1, w2(C), c2, r3(C), w3(A), c3

Ist der Ablauf serialisierbar?

8.3 Anfrage mit Transaktion

1. Transaktionsstart
2. Kontonummer über Benutzereingabe abfragen
3. Über Konten scannen und Kontostand bilden
4. Zielkonto und Betrag über Benutzereingabe abfragen
5. Abbruch, falls Kontostand zu gering
6. Geld überweisen
7. Transaktionsende

Was ist das Problem mit dieser Transaktion?

8.4 Problemlösung 1 Punkt

Wie könnte man das Problem von oben lösen?

8.5 Sperren

Sinn und Funktionsweise von Serialisierung mittels Sperren erklären.

9 Programmzugriff

9.1 SQL-Anfrage in Java mit JDBC

Relation Person: (Vorname, Nachname, Geburtsjahr)
Programm schreiben, dass alle Leute auf der Konsole ausgibt, die vor 1970 geboren wurden.

Die genauen Bezeichner der API waren ausdrücklich nicht gefordert; es ging eher darum, die richtigen Konzepte der Schnittstelle zu benutzen. Folgendes Code-Skelett war vorgegeben:

```
public static void main(String[] args){
    try {
        Connection con = // Verbindungsaufbau gegeben

        Statement stat =

        ResultSet resSet =

    }
    catch (Exception e) {
        // Fehlerbehandlung gegeben
    }
}
```

9.2 Prepared statements

Vorteile von Prepared statements gegenüber Abfrage zur Laufzeit nennen.

9.3 Stored procedures

Vorteil von Stored procedures zu Prepared statements?