TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Alfons Kemper, Ph.D.

Übung zur Vorlesung *Einführung in die Informatik 2 für Ingenieure (MSE)*

Alexander van Renen (renen@in.tum.de)
http://db.in.tum.de/teaching/ss17/ei2/

Lösungen zu Blatt 7

Aufgabe 1: Hashing in Java

Warum sollte man in Java, wenn man `equals()` überschreibt, auch `hashCode()` überschreiben?

Lösung

Überschreibt man nur eine der beiden Methoden kommt es zu unerwarteten Ergebnissen. Ein Beispiel sind Hashtabellen, die davon ausgehen, dass wenn zwei Objekte gleich sind (im Sinne von `equals()`), sie auch den gleichen Hashwert haben (`hashCode()`). Sonst kann es passieren, dass zwei „gleiche“ Objekte an verschiedene Stellen in der Hashtabelle landen.

```
1  import java.util.HashMap;
2
3  class Student {
4    String name;
5    Integer matrNr;
6
7    Student(String name, Integer matrNr) {
8      this.name = name;
9      this.matrNr = matrNr;
10   }
11
12   public int hashCode() {
13     final int prime = 31;            // Berechne den hashCode
           basierend
14     int result = 1;                  // auf den gleichen Attributen!
15     int result = 31 * result + name.hashCode();
16     return 31 * result + matrNr.hashCode();
17     return result;
18   }
19
20   public boolean equals(Object obj) {
21     if (obj == this)                 // Teste auf Referenzgleichheit
22       return true;
23     if (!(obj instanceof Student))   // Teste Klasse
24       return false;
25     Student other = (Student)obj;    // Teste Attribute
26     return name.equals(other.name) && matrNr.equals(other.matrNr);
27   }
28 }
```

```
29
30 class Equality {
31   public static void main(String[] args) {
32     HashMap<Student, Integer> map = new HashMap<Student, Integer>();
33     Student uliOriginal = new Student("Uli", 123456789);
34     Student uliKlon = new Student("Uli", 123456789);
35     map.put(uliOriginal, 11);
36     System.out.println(map.containsKey(uliKlon)); // Wertet zu "true
           " aus
37   }
38 }
```

Aufgabe 2: Motivation von DBMS

Nennen Sie drei typische Probleme, die bei dem Verzicht auf ein Datenbankverwaltungssystem eintreten können. Überlegen Sie sich jeweils ein Beispiel bei dem das Problem auftritt.

Lösung

Im folgenden eine Liste von möglichen Problemen:

Redundanz. Ohne DBMS kommt es leicht zu Redundanz in den Daten und diese kann bei Veränderungen zu Inkonsistenzen führen. Ein DMBS garantiert die **Konsistenz** der Daten.

Beschränkte Zugriffsmöglichkeit. Verknüpfung der Daten ist ohne DBMS sehr schwer, da diese dann oft nicht konsistent modelliert und abgespeichert sind.

Probleme durch Mehrbenutzerbetrieb. Gleichzeitige Änderung der gleichen Daten durch verschiedene Benutzer führt schnell zu Anomalien. Ein DMBS garantiert die logische **Isolation** der Anfragen verschiedener Benutzer.

Datenverlust. Bei Speicherung der Daten in Dateien sind meist höchstens regelmäßige Backups möglich. Ein DBMS garantiert hingegen die **Dauerhaftigkeit** jeder durchgeführten Operation. Weiterhin werden Änderungen ganz oder gar nicht durchgeführt (**Atomarität**).

Integritätsverletzung. In der realen Welt gibt es komplexe Integritätsbedingungen, die sich bei Speicherung in Dateien leicht verletzen lassen.

Sicherheitsprobleme. Ein Beispiel ist der Datenschutz, da nicht alle Benutzer Zugriff auf alle Daten haben sollten. Ein DBMS bietet hierfür Rollen und Zugriffsrechte.

Hohe Entwicklungskosten. Bei der Eigenentwicklung der Datenspeicherung in einem Anwendungsprogramm erfindet man das Rad neu und muss alle oben genannten Probleme lösen. Dies führt zu erheblichen Entwicklungskosten. DBMS hingegen sind getestete Standardkomponenten, die man sehr einfach einsetzen kann.

Die vier wichtigsten Eigenschaften der Datenverarbeitung in einem DBMS haben eine leicht zu merkende Abkürzung: **ACID**. Dies steht für **A**tomicity, **C**onsistency, **I**solation und **D**urability.

Professoren			
PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

Abbildung 1: Professoren in der relationalen Modellierung

Aufgabe 3: Terminologie

Beschreiben Sie die folgenden Begriffe der relationalen Modellierung. Verwenden Sie die Relation Professoren aus Abbildung 1 um Beispiele für die einzelnen Konzepte anzugeben.

Lösung

Hier eine kurze Beschreibung der Konzepte mit Beispielen:

Attribut. Ein Attribut ist eine Eigenschaft der Entitäten der Relation (entspricht einer Spalte, bei der Relation Professoren z.B. Name).

Schlüssel. Ein Schlüssel identifiziert ein Tupel eindeutig (z.B. die PersNr des Professors).

Relation. Eine Relation besteht aus Schema und Ausprägung (die gesamte Tabelle).

Domäne. Der Wertebereich eines Attributs (bei der PersNr z.B. alle `int` Zahlen).

Tupel. Eine Entität einer Relation (entspricht einer Zeile, z.B. Professor Russel).

Schema. Die Attribute einer Relation mit deren Typen (in diesem Fall `{[PersNr: integer, Name: varchar(30), Rang: varchar(2), Raum: integer]}`).

Ausprägung. Die Gesamtheit aller Tupel einer Relation (alle Zeilen bis auf die Kopfzeilen).

Aufgabe 4: Joins

Wir haben die folgende Anfrage in der Vorlesung kennen gelernt. Zur Beantwortung haben wir das Kreuzprodukt der drei Relationen gebildet und dann die Zeilen gestrichen, die die Bedingungen nicht erfüllen. Dies ging relativ schnell, da wir nur zwei Tupel pro Relation hatten und das Kreuzprodukt **Studenten** × **hoeren** × **Vorlesungen** somit aus nur acht Tupeln (2·2·2) bestand.

```
SELECT Name
FROM   Studenten, hoeren, Vorlesungen
WHERE  Studenten.MatrNr = hoeren.MatrNr AND
          hoeren.VorlNr = Vorlesungen.VorlNr AND
      Vorlesungen.Titel = 'Grundzuege';
```

In der vollständigen Ausprägung (siehe Abbildung 2) gibt es 8 Studenten, 12 hören-Einträge und 10 Vorlesungen. Das ergibt ein Kreuzprodukt mit $8 \cdot 12 \cdot 10 = 960$ Kombinationen. Wie könnte man geschickt vorgehen, um die Anfrage trotzdem noch von Hand zu lösen?

Studenten		
MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10
26830	Aristoxenos	8
27550	Schopenhauer	6
28106	Carnap	3
29120	Theophrastos	2
29555	Feuerbach	2

hören	
MatrNr	VorlNr
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
29555	5022
25403	5022

Vorlesungen			
VorlNr	Titel	SWS	gelesen von
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
4630	Die 3 Kritiken	4	2137

Abbildung 2: Ausprägung der Relationen Studenten, hören und Vorlesungen

Lösung

Statt dem Kreuzprodukt der drei Tabellen, wählt man zuerst alle Vorlesungen aus, die die Bedingung `Vorlesungen.Titel = 'Grundzuege'` erfüllen (dies ist nur eine einzige). Das Zwischenergebnis verknüpft man dann mit der `hoeren`-Relation über die Bedingung `hoeren.VorlNr = Vorlesungen.VorlNr`. Das erneute Zwischenergebnis verknüpft man wiederum mit der `Studenten`-Relation über die verbliebene Bedingung `Studenten.MatrNr = hoeren.MatrNr`.

Durch die schrittweise Ausführung der Verknüpfungen (Joins) und indem man Bedingungen, die viele Tupel filtern, zuerst ausführt (z.B. `Vorlesungen.Titel = 'Grundzuege'`), kann man die Größe der Zwischenergebnisse deutlich reduzieren. In einem Datenbanksystem versucht der Optimierer die Anfragebearbeitung mit diesen und ähnlichen Techniken zu beschleunigen. Es ist im Allgemeinen nicht leicht eine effiziente Ausführungsreihenfolge zu finden, besonders wenn die Anfrage sehr viele Relationen miteinander verknüpft.