Prof. Dr. Stefan Brass
Institut für Informatik
MLU Halle-Wittenberg

15. Dezember 2011

Datenbanken II B: DBMS-Implementierung

— Hausaufgabe 8A —

Sie sollen jetzt den Anfang von einem Data Dictionary (System-Katalog) anlegen, nämlich die Liste der Relationen in einer Datei. Entwickeln Sie ein Klasse `rel_c` (Relation/Tabelle) mit folgenden Methoden:

- `rel_c(file_t file, str_t name)` (Konstruktor)
- `bool create(int size)`
  Prüft, dass die Datei nicht bereits eine Relation mit dem Namen enthält, und legt dann so eine Relation an, d.h. es wird ein Segment erzeugt und der Relationenname und die Segment-ID in das Data Dictionary eingetragen.

  Sie müssen damit rechnen, dass Sie später noch weitere Informationen über Relationen speichern müssen (z.B. Tupellänge, Anzahl Spalten, und ein Link zur ersten Spalte).

  Falls die Erzeugung erfolgreich war (es gab nicht schon eine Relation mit dem Namen, und ein Segment der gewünschten Größe konnte angelegt werden), wird `true` geliefert, sonst `false`.
- `bool open()`
  Prüft, ob die Datendatei eine Relation mit diesem Namen enthält, und läd ggf. Zugriffsdaten der Relation (d.h. mindestens die Segment-ID). Falls es die Relation gibt, soll `true` geliefert werden, sonst `false`.
- `int seg_id()`
  Liefert die ID des Segmentes, in dem die Relation gespeichert ist. Falls das `rel_c`-Objekt nicht in offenem Zustand ist (d.h. vorher nicht `create()` oder `open()` aufgerufen wurden), soll `-1` geliefert werden.

Diese Schnittstelle ist nur als Vorschlag zu verstehen. Sie könnten natürlich auch einen Konstruktor ohne Parameter anbieten, und dann bei `open` und `create` den Namen der Relation und die Datei angeben.

Wenn Sie wollen, können Sie die Länge von Relationen-Namen einschränken, z.B. auf maximal 18 Zeichen. Dann hätten die Einträge in der Liste der Relationen eine feste Länge (und könnten auch direkt C++-Objekten entsprechen).

Da Sie in Kürze Tupel in Relationen implementieren müssen, bietet es sich natürlich an, sie jetzt schon zu implementieren und auch für das Data Dictionary zu verwenden. (Einen Vorschlag für die Schnittstelle finden Sie in der Projekt-Skizze.). Das ist Ihnen aber freigestellt. Es ist z.B. auch möglich, sich nur Gedanken zu machen, wie Sie Tupel

implementieren wollen, und dann hier die gleiche Technik zu verwenden. Sie können aber auch einen ganz eigenen Blocktyp für die Relationsliste verwenden (d.h. Sie brauchen nicht die gleiche Datenstruktur wie für Relationen zu benutzen). Die Anzahl der Relationen in einer Datei ist normalerweise nicht sehr groß, z.B. 100, insofern gibt es einen Unterschied zu allgemeinen Relationen (diese müssen eine Datei von mindestens 1 GB füllen können).

— Hausaufgabe 8B —

Gegeben sei eine Datenbank mit den Tabellen

- `EMP(`<u>`EMPNO`</u>`, ENAME, SAL, DEPTNO`→`DEPT, MGR`°→`EMP)`
- `DEPT(`<u>`DEPTNO`</u>`, DNAME, LOC)`

Dies ist eine etwas vereinfachte Version der klassischen Oracle-Beispieldatenbank. Nehmen Sie an, die Angestellten-Tabelle `EMP` enthält 10000 Zeilen, und die Abteilungs-Tabelle `DEPT` 25 Zeilen, und die Angestellten pro Abteilung seien gleichverteilt. Ein Vorgesetzter (`MGR`) habe durchschnittlich 10 Untergebene. Nehmen Sie weiter an, dass 10 Datensätze pro Block gespeichert werden (bei beiden Tabellen). Es sei die folgende Anfrage gegeben:

```
SELECT   EMPNO, SAL, DNAME
FROM     EMP E, DEPT D
WHERE    E.DEPTNO = D.DEPTNO
AND      E.MGR = 7839
ORDER BY SAL
```

Beschreiben Sie, wie man die Anfrage auswerten kann, wenn man folgende Indexe hat:

a) Keinen.

b) `EMP(EMPNO)` und `DEPT(DEPTNO)` (beide natürlich `UNIQUE`).

c) Wie b), zusätzlich `EMP(MGR)`.

d) Wenn Sie sich einen Index aussuchen dürften, auch einen mit mehreren Attributen, welchen würden Sie wählen? Dabei soll es nur darum gehen, dass die obige Anfrage möglichst schnell abgearbeitet wird.