Lehrstuhl Praktische Informatik III
Universität Mannheim
Prof. Dr. Guido Moerkotte
Sven Helmer

April 2000

Hauptdiplomklausur Datenbankpraktikum Wintersemester 1999/2000

Name: .....
Vorname: .....
Matrikelnummer: .....
Studienfach: .....

Wichtige Hinweise:

1. Prüfen Sie Ihr Klausurexemplar auf Vollständigkeit (10 Seiten).
2. Es sind keine Hilfsmittel zugelassen.
3. Die Klausur dauert 66 Minuten.
4. Jede Aufgabe ist auf dem zugehörigen Aufgabenblatt (und ggf. auf separaten Lösungsblättern) zu bearbeiten.
5. Vermerken Sie Ihren Namen und Ihre Matrikelnummer auf jedem Aufgaben- (bzw. Lösungsblatt). Blätter ohne Namens- und Matrikelnummerangabe werden nicht bewertet.
6. Das Deckblatt sowie alle Aufgabenblätter (evtl. Lösungsblätter) sind abzugeben.

	maximale Anzahl Punkte	erreichte Anzahl Punkte
Aufgabe 1	12	
Aufgabe 2	6	
Aufgabe 3	20	
Aufgabe 4	28	
	66	

1. (12 Punkte)
   Eine Fahrradwerkstatt verwaltet ihr Sortiment an Ersatzteilen in einer relationalen Datenbank (sprich: DB2). Ihr verwendetes Datenbankschema umfaßt unter anderem folgende drei Relationen:

   - Teil(<u>TeilNr</u>, Name, Gewicht, Länge, Farbe, HNr)
   - besteht_aus(<u>TeilNr, UnterteilNr</u>, Anzahl)
   - Hersteller(<u>HNr</u>, Name, Adresse)

   Die Relation *Teil* enthält allgemeine Daten zu den angebotenen Teilen. In der Relation *besteht_aus* wird festgehalten aus welchen Unterteilen ein Ersatzteil aufgebaut ist. *Hersteller* umfaßt Daten des Herstellers eines Ersatzteils.

   Geben Sie die TeilNr und den Namen aller Teile an, an denen der Hersteller Peugeot direkt oder indirekt beteiligt ist (d.h. Peugeot stellt das Teil selbst oder ein Unterteil davon her).

2. (a) (3 Punkte)
   Angenommen Sie haben ein C-Programm `embed.sqc` mit Embedded SQL. Was passiert beim Aufruf des Kommandos "PREP embed.sqc"?

   (b) (3 Punkte)
   Was ist der Unterschied zu "BIND embed.sqc"?

3. (20 Punkte)
Geben Sie ein Programm *peugeot.sqc* in Embedded SQL (in SQL-2, d.h. ohne rekursive Aufrufe) für die rekursive Anfrage aus Aufgabe 1(a) an. Sie dürfen dabei auf die folgenden vordefinierten Funktionen zurückgreifen.

```
void basisfall() {
   EXEC SQL insert into temp
            select t.teilnr, t.name
            from   teil t, hersteller h
            where  h.name = 'Peugeot'
            and    t.hnr = h.hnr;
}

void rekschritt() {
   EXEC SQL insert into temp
            select t.teilnr, t.name
            from   teil t, besteht_aus b, temp p
            where  t.teilnr = b.teilnr
            and    b.unterteilnr = p.teilnr
            and    t.teilnr not in (select teilnr from temp);
}

void anzahlteile() {
   EXEC SQL select count(distinct teilnr) into :anzahl
            from   temp;
}

int tuplespresent() {
   if(strncmp(sqlca.sqlstate, "02000", 5)) {
      /* es gibt noch Tupel */
      return 1;
   }
   else {
      /* es gibt keine Tupel mehr */
      return 0;
   }
}
```

Der Name der Datenbank lautet *Sortiment*. Die Relation *temp* wurde bereits mit folgendem Befehl erzeugt:

```
create table temp (TeilNr integer,
                   Name   varchar(80));
```

Geben Sie außerdem noch eine Funktion *ausgabetemp()* an, die die Relation *Temp* nach Abarbeitung auf stdout (printf) ausgibt.

```
#include <...>

EXEC SQL INCLUDE SQLCA;

int main() {

   EXEC SQL BEGIN DECLARE SECTION;
      int anzahl;
```

4. In einer Datenbank werden Prüfungsergebnisse verwaltet. Zu diesem Zweck wurde ein eigener Datentyp *Note* eingeführt:

```
CREATE DISTINCT TYPE note AS DOUBLE WITH COMPARISONS;
```

   (a) (2 Punkte)
   Um auch Durchschnittsnoten berechnen zu können, wurde die AVG-Funktion für den Typ *Note* umgesetzt:

```
CREATE FUNCTION AVG(note) RETURNS note
       SOURCE AVG(DOUBLE);
```

   Wie nennt sich diese Art von Funktion?

   (b) (8 Punkte)
   Schreiben Sie eine externe Funktion, die die Noten in textueller Form ausgibt. Die Noten sollen folgendermaßen umgesetzt werden:

	bis	1,49	sehr gut
1,5	bis	2,49	gut
2,5	bis	3,49	befriedigend
3,5	bis	4,49	ausreichend
4,5	bis		durchgefallen

   Die CREATE FUNCTION Anweisung und der Prozedurkopf sind bereits angegeben:

```
CREATE FUNCTION notentext(note)
   RETURNS char(15)
   EXTERNAL NAME 'noten!notentext'
   NOT VARIANT
   NO EXTERNAL ACTION
   NOT NULL CALL
   LANGUAGE C
   FENCED
   PARAMETER STYLE DB2SQL
   NO SQL;
```

```
char* notentext(
   double* noteIn
   char*   textOut
   short*  nullNoteIn
   short*  nullTextOut
   char*   sqlstate,
   char*   fnName,
   char*   specificName,
   char*   message)
```

(c) (6 Punkte)
Erläutern Sie den Unterschied zwischen

i. VARIANT/NOT VARIANT

ii. FENCED/NOT FENCED

iii. NULL CALL/NOT NULL CALL

(d) (12 Punkte)
In der Datenbank existieren folgende Relationen:

- Studenten(MatrNr, Name)
- Ergebnisse(MatrNr, Teilfach, Ergebnis)

Das Attribut *Ergebnis* ist vom Typ *Note*.

Formulieren Sie eine SQL-Anfrage, die die Matrikelnummern, Namen und Durchschnittsnoten (in Textform) der Studenten ausgibt, die bestanden haben. Ein Student hat bestanden, wenn der Durchschnitt aller Teilfachprüfungen besser als 4,49 ist und er in maximal einer Teilfachprüfung durchgefallen ist.

Syntax für Cursoranweisungen: