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

April 2000

Hauptdiplomklausur
Datenbanksysteme I
Wintersemester 1999/2000

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

Wichtige Hinweise:

1. Prüfen Sie Ihr Klausurexemplar auf Vollständigkeit (14 Seiten).
2. Es sind keine Hilfsmittel zugelassen.
3. Die Klausur dauert 100 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	9	
Aufgabe 3	27	
Aufgabe 4	15	
Aufgabe 5	17	
Aufgabe 6	16	
Aufgabe 7	4	
	100	

1. (12 Punkte)
   Geben Sie für folgende Aussagen an, ob sie wahr oder falsch sind. (Bei einer falschen Antwort werden Punkte abgezogen, die Gesamtpunktzahl kann jedoch nicht unter 0 Punkte sinken.)

	wahr	falsch
Für jedes Relationenschema $\mathcal{R}$ existiert eine 3NF-Zerlegung, die verlustlos und abhängigkeitsbewahrend ist.	☐	☐
Gegeben sei ein Schema $\mathcal{R}(G, H, I, J, K)$ mit einer Menge $\mathcal{F}_\mathcal{R} = \{I \to G, K \to J, G \to H\}$. H ist kein Kandidatenschlüssel des Schemas $\mathcal{R}$.	☐	☐
Eine serielle Historie ist immer auch serialisierbar, rücksetzbar, strikt und vermeidet kaskadierendes Rücksetzen.	☐	☐
Das WAL-Prinzip besagt, daß vor dem Ausschreiben der Logeinträge einer Seite, die modifizierte Seite ausgelagert werden muß	☐	☐
Das strenge Zwei-Phasen-Sperrprotokoll erzeugt immer serielle Historien.	☐	☐
Gegeben sei ein Schema $\mathcal{R}(A, B, C, D)$ mit einer Menge $\mathcal{F}_\mathcal{R}$ funktionaler Abhängigkeiten in höchster Normalform 2NF und ein Schema $\mathcal{S}(A, B, C, D)$ mit einer Menge $\mathcal{F}_\mathcal{S}$ in höchster Normalform 3NF $\Rightarrow \mathcal{F}_\mathcal{R}^+ \neq \mathcal{F}_\mathcal{S}^+$	☐	☐

2. (9 Punkte)
Eine große Versandfirma speichert die gesamten Daten ihrer Lieferungen in einem relationalen Datenbanksystem. Die größte Relation *Posten* enthält folgende Attribute:

- Posten(RechnungsNr, ArtikelNr, Anzahl, Artikelpreis, Auslieferdatum)

Aus Leistungsgründen wurde die Relation nicht in 3NF gebracht, so existiert die funktionale Abhängigkeit $ArtikelNr \rightarrow Artikelpreis$. Beim Einfügen von neuen Tupeln soll aber sichergestellt werden, daß diese Abhängigkeit eingehalten wird. Schreiben Sie einen Trigger, der dies bewerkstelligt.

Zur Erinnerung ist hier noch einmal die Syntax eines Triggers angegeben:

```
create trigger TRIGGER_NAME
       before | after
       delete | insert | update [of ATTRIBUT_LISTE]
       on RELATIONEN_NAME
       [for each row [when PRAEDIKAT]]
       begin
       PL/SQL_BLOCK
       end;
```

Mit **new.ATTRIBUT_NAME** kann der Wert eines Attributs eines neu eingefügten Tupels nach dem Triggereffekt abgefragt werden. Gehen Sie weiterhin davon aus, daß Sie eine Funktion *error("Fehlermeldung")* zur Verfügung haben, mit der eine Fehlermeldung ausgegeben und der SQL-Befehl abgebrochen werden kann.

3. Die Betreiber einer Suchmaschine verwalten Statistiken über ihren Rechnerpool in einer relationalen Datenbank mit folgenden Relationen:

   - Rechner(IP-Adresse, Name, Hauptspeicher, Plattenkapazität, CPU, ServiceID)
   - Auslastung(IP-Adresse, Datum, Auslastung)
   - Ausfall(IP-Adresse, Datum, Grund)
   - Firma(ServiceID, Name, Telefon)

   In der Relation *Rechner* steht die IP-Adresse und der Name des Rechners. Weiterhin sind dort die Hauptspeichergröße (in MByte), die Plattenkapazität (in GByte), die CPU-Art (Intel Pentium III, AMD K-6, Alpha, UltraSparc etc.) und die Firma zu finden, die den Rechner verkauft hat und für den Service zuständig ist. Die Relation *Auslastung* gibt für jeden Tag die Auslastung (in %) des jeweiligen Rechners an. In der Relation *Ausfall* wird vermerkt, an welchen Tagen der Rechner ausgefallen ist (mit dem Grund des Ausfalls). Für ein bestimmtes Datum kann ein Rechner entweder in *Auslastung* oder in *Ausfall* gefunden werden, aber nicht in beiden! Die Relation *Firma* speichert den Namen und die Telefonnummer der Firma, die den Rechner verkauft hat und für den Service zuständig ist.

   (a) (4 Punkte)
   Geben Sie die IP-Adressen, Namen und Plattenkapazitäten aller Rechner an, die mehr als 64 MByte Hauptspeicher haben. Sortieren Sie die Tupel absteigend nach Plattenkapazitäten.

Rechner(IP-Adresse, Name, Hauptspeicher, Plattenkapazität, CPU, ServiceID)
Auslastung(IP-Adresse, Datum, Auslastung)
Ausfall(IP-Adresse, Datum, Grund)
Firma(ServiceID, Name, Telefon)

(b) (5 Punkte)
Geben Sie die Tage an, an denen mindestens ein Rechner mit Intel Pentium III-CPU ausgefallen ist. Kein Datum sollte mehr als einmal in dieser Liste auftauchen.

Rechner(IP-Adresse, Name, Hauptspeicher, Plattenkapazität, CPU, ServiceID)
Auslastung(IP-Adresse, Datum, Auslastung)
Ausfall(IP-Adresse, Datum, Grund)
Firma(ServiceID, Name, Telefon)

(c) (8 Punkte)
Geben Sie für jede Firma die Zuverlässigkeit ihrer Rechner an. Die Zuverlässigkeit wird angegeben durch die Anzahl der Tage an denen die Rechner arbeiteten (Einträge in Auslastung) geteilt durch die Anzahl aller Tage (Einträge in Auslastung + Einträge in Ausfall).

Rechner(IP-Adresse, Name, Hauptspeicher, Plattenkapazität, CPU, ServiceID)
Auslastung(IP-Adresse, Datum, Auslastung)
Ausfall(IP-Adresse, Datum, Grund)
Firma(ServiceID, Name, Telefon)

(d) (10 Punkte)
Die Funktionalität der Suchmaschine soll erweitert werden. Dafür wird ein Rechner gesucht, der wenig ausgelastet ist. Geben Sie die IP-Adresse und den Namen des Rechners mit der kleinsten durchschnittlichen Auslastung an.

4. (15 Punkte)
   In Abbildung 2 ist ein konzeptuelles Schema in Form eines ER-Diagramms abgebildet, in Abbildung 1 das zugehörige relationale Schema. Beide Schemata sind jedoch lückenhaft. Vervollständigen Sie jedes Schema mit den Informationen aus dem jeweils anderen Schema.

- B($\underline{b_1}$, b_2, b_3)
- D($\underline{c_1}$, c_2, d_1)
- J($\underline{c_1}$, c_2, j_1, j_2)
- R($\underline{c_1, a_1, b_1}$, r_1)
- T($\underline{b_1, c_1}$, f_1)

Abbildung 1: lückenhaftes relationales Schema

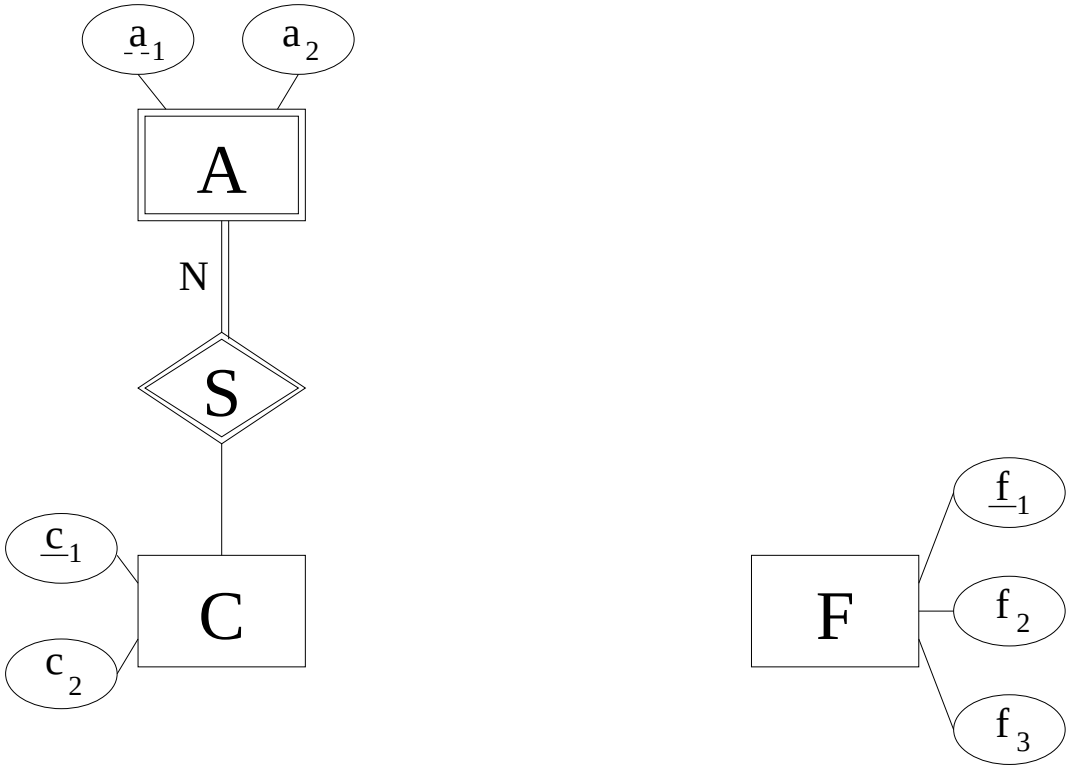

Abbildung 2: lückenhaftes ER-Schema

5. Betrachten wir noch einmal die Versandfirma aus Aufgabe 2. Für diese Aufgabe werden die Relationen

   - Posten(<u>RechnungsNr, ArtikelNr</u>, Anzahl, Artikelpreis, Auslieferdatum)
   - Rechnung(<u>RechnungsNr</u>, Gesamtpreis, DatumBezahlung, KundenNr)
   - Kunde(<u>KundenNr</u>, Name, Adresse, Telefon)

   betrachtet, die auf verschiedenen Rechnern in der Zentrale und mehreren Zweigstellen gespeichert sind. In einer Filiale soll jetzt geprüft werden, ob der Kunde mit der KundenNr 08/15 schon alle seine Rechnungen bezahlt hat. Die Relation *Posten* liegt auf Server S_1 in der Zentrale, die Relationen *Rechnung* und *Kunde* auf Server S_2 in der Filiale. Dazu wird folgende Anfrage in der Filiale gestartet (wo auch das Ergebnis der Anfrage ausgegeben werden soll):

```
select K.Name, K.Adresse, P.ArtikelNr, P.Auslieferdatum
from   Posten P at S1, Rechung R at S2, Kunde K at S2
where  K.KundenNr = 08/15
and    K.KundenNr = R.KundenNr
and    R.DatumBezahlung is NULL
and    R.RechnungNr = P.RechnungsNr;
```

   (a) (2 Punkte)
   Welche Transparenz liegt hier vor?

   (b) (5 Punkte)
   Geben Sie für die Anfrage einen kanonischen Operatorbaum an.

(c) (10 Punkte)
Die Größen der Relationen sehen folgendermaßen aus: *Posten* 641 GByte, *Rechnung* 54 GByte, *Kunde* 45 GByte. Zur Übertragung steht lediglich eine Leitung mit 64 KByte/s zur Verfügung, d.h. komplette Relationen können nicht übertragen werden (die Übertragung der kleinsten Relation würde schon ca. 8 Tage dauern!). Geben Sie einen optimierten Operatorbaum an, der die Übertragungszeit in erträglichem Rahmen hält.

6. In einem Datenbanksystem wird ein sehr konservativer Scheduler mit "Preclaiming" eingesetzt, d.h. Transaktionen werden erst begonnen, wenn alle ihre Sperranforderungen schon bei Transaktionsbeginn erfüllt werden können. Transaktionen, deren Anforderungen nicht erfüllt werden können, werden an eine Warteschlange angehängt. Transaktionen deren Anforderungen erfüllt werden können, werden sofort gestartet (d.h. sie gelangen nie in die Warteschlange). Immer wenn eine Transaktion beendet wird und ihre Sperren freigibt, wird in der Warteschlange (von vorne) solange gesucht bis eine Transaktion gefunden wird, deren Anforderungen vollständig erfüllt werden können. Diese Transaktion wird dann gestartet.

   (a) (6 Punkte)
   Betrachten Sie folgendes Szenario: die Transaktionen mit ungeraden Nummern greifen auf das Datenelement A zu, die Transaktionen mit geraden Nummern greifen auf das Datenelement B zu.

T_1	T_2	T_3	T_4	T_5	...
BOT_1					
$r_1[A]$					
$w_1[A]$					
	BOT_2				
COMMIT_1					
	$r_2[B]$				
	$w_2[B]$				
		BOT_3			
	COMMIT_2				
		$r_3[A]$			
		$w_3[A]$			
			BOT_4		
		COMMIT_3			
			$r_4[B]$		
			$w_4[B]$		
				BOT_5	
			COMMIT_4		
				$r_5[A]$	
				$w_5[A]$	
					$\mathrm{BOT}_{\ldots}$
				COMMIT_5	
					...

   Deadlocks werden mit Hilfe des "Preclaiming" vermieden, allerdings kann es zu sogenannten Livelocks kommen. Geben Sie ein Beispiel einer Transaktion an, die in diesem Szenario nie zum Zuge kommt.

(b) (6 Punkte)
Wie können Sie das oben angegebene Verfahren ändern, so daß keine Livelocks mehr auftreten?

(c) (4 Punkte)
Was sind weitere Nachteile des "Preclaiming"-Verfahrens?

7. (a) (2 Punkte)
Was sind Präfix-B^+-Bäume?

(b) (2 Punkte)
Sind auch Präfix-B-Bäume möglich? Begründen Sie Ihre Antwort kurz.