

Probeklausur

Datenbankmanagementsysteme

Name: Vorname:

Matr.Nr: Studiengang:

Aufgabe Nr.	Max. Punkte	Erreichte Punkte
1	10	
2	12	
3	32	
4	32	
5	14	
6	15	
Summe	115	
Note:		

- Bitte füllen Sie zuerst das Deckblatt aus und legen Sie **Studentenausweis** und **Personalausweis** bereit.
- Nach der Einlesezeit von 10 Minuten beginnt die **Bearbeitungszeit von 115 Minuten**.
- Falls der Platz nicht für die Lösungen ausreichen sollte, benutzen Sie bitte die leeren Seiten zwischen den Aufgaben bzw. am Ende der Klausur.

Wir wünschen Ihnen viel Erfolg!

Aufgabe 1: Relationentypen

(10 Punkte)

Erklären Sie, was unter den folgenden Relationentypen verstanden wird:

- a) Basisrelation
- b) Abgeleitete Relation
- c) Virtuelle Relation
- d) Sicht
- e) Schnappschussrelation

Lösung:

- **Basisrelation:**
Benannte Relation, die Bestandteil des konzeptuellen Datenbankschemas ist. Daher existiert für sie nicht nur eine logische, sondern auch eine physische Repräsentation in der DB.
- **Abgeleitete Relation:**
Eine durch Anwendung von relationalen Operatoren aus einer oder mehreren Relationen abgeleitete Relation.
- **Virtuelle Relation:**
Benannte abgeleitete Relation, von der kein physisches Abbild in der DB existiert.
- **Sicht:**
Eine Sicht ist eine benannte abgeleitete Relation. Sie spiegelt die speziell angepasste Sicht einer Benutzergruppe oder Anwendung auf einen Teil der DB wider. Sichten stellen im Normalfall virtuelle Relationen dar.
- **Schnappschussrelation:**
Materialisierte Relation, die genau einen Zustand einer Relation einfriert. Aus logischer Sicht dokumentiert sie ein unveränderliches Abbild (Schnappschuss) eines zu irgendeinem Zeitpunkt einmal gültig gewesenen Realwelt- und damit Datenbankzustandes.

Aufgabe 2: Transaktionen und Deadlocks

(12 Punkte)

- a) Definieren Sie eine Transaktion und erklären Sie kurz die ACID-Eigenschaften.
- b) Was ist ein Deadlock und welche vier Bedingungen müssen erfüllt sein, damit es zu einem Deadlock kommen kann?

Lösung:

a) **Transaktion:**

- eine Folge von logisch zusammengehörigen Operationen auf der DB, die von einer Anwendung durchgeführt werden.
- repräsentiert eine Konsistenzwahrende Einheit:
überführt die DB von einem konsistenten Zustand in einen nicht zwangsläufig unterschiedlichen konsistenten Zustand.

ACID-Eigenschaften:

Atomicity (Unteilbarkeit)

- ~ nur die gesamte Folge von Operationen garantiert die DB-Konsistenz
- ~ wird entweder vollständig oder gar nicht ausgeführt
- ~ beim Fehler werden alle Änderungen rückgängig gemacht

Consistency (Konsistenz)

- ~ die DB wird von / in einen konsistenten Zustand überführt
- ~ inkonsistente Zwischenzustände werden bis Tr.-Ende beseitigt

Isolation (Isolation)

- ~ so als gäbe es nur eine Transaktion
- ~ keine gegenseitige Beeinflussung

Durability (Dauerhaftigkeit)

- ~ ist eine Transaktion erfolgreich abgeschlossen, so sind die Änderungen dauerhaft (auch im Fehlerfall)

b) **Deadlock**

Ein **Deadlock** ist eine Situation, bei der Prozesse wechselseitig darauf warten, dass der jeweils andere Prozess von ihnen benötigte Betriebsmittel freigibt.

Deadlocks können in allen Systemen auftreten, in denen allgemein zugängliche, exklusiv zu nutzende Betriebsmittel an parallel laufende, konkurrierende Prozesse vergeben werden und bei denen die folgenden vier Bedingungen alle erfüllt sind:

- Bedingung des gegenseitigen Ausschlusses
- Bedingung der Nichtentziehbarkeit
- Warte-und-Halte-Fest-Bedingung
- Bedingung der Mehrfachanforderung

Aufgabe 3: Relationale Algebra

(32 Punkte)

1) Gegeben seien zwei Relationen r und s:

	<u>A</u>	<u>B</u>	<u>D</u>	<u>E</u>		<u>B</u>	<u>C</u>	<u>D</u>
	1	2	9	8		4	5	9
r =	1	4	9	8	s =	4	7	8
	1	5	9	6		6	5	7
	1	6	7	6		4	8	9

Berechnen Sie die Ergebnisrelationen folgender Ausdrücke:

a) $\sigma_{B < C} s$

b) $\pi_{E, C}(r \bowtie s)$

c) $\sigma_{A=1}(r \bowtie s)$

d) $\pi_{B, C}(r \bowtie s)$

Lösung:

a) $\sigma_{B < C} s$

B	C	D
4	5	9
4	7	8
4	8	9

b) $\pi_{E, C}(r \bowtie s)$

E	C
8	5
8	8
6	5

c) $\sigma_{A=1}(r \bowtie s)$

A	B	D	E
1	4	9	8
1	6	7	6

d) $\pi_{B, C}(r \bowtie s)$

B	C
2	\otimes
4	5
4	8

5	⊗
6	5

2) Gegeben seien folgende Relationen:

Kunde (kundenNr, kundenName, kundenWohnort)

Produkt (produktNr, produktBezeichnung, produktionsOrt, produktPreis)

Kaufbeleg (belegNr, kundenNr, produktNr, kaufMenge, kaufDatum, verkäuferName)

Formulieren Sie die folgenden Anfragen in relationaler Algebra:

- a) Geben Sie die Produktbezeichnungen und Preise von den Produkten aus, die in Essen produziert werden.

$\pi_{\text{produktBezeichnung, produktPreis}} (\sigma_{\text{produktionsOrt}='Essen'} \text{Produkt})$

- b) Geben Sie alle Daten der Kunden aus, die bisher noch nie etwas beim Verkäufer mit dem Namen Müller gekauft haben.

$\text{Kunde} - (\text{Kunde} \bowtie (\sigma_{\text{verkäuferName}='Mueller'}(\text{Kaufbeleg})))$

- c) Geben Sie alle Daten der Kunden aus, die bisher ausschließlich beim Verkäufer Müller gekauft haben.

$\text{Kunde} - (\text{Kunde} \bowtie (\sigma_{\text{verkäuferName} \neq 'Mueller'}(\text{Kaufbeleg})))$

alternativ: $\text{Kunde} \bowtie (\pi_{\text{KundenNr}}(\text{Kunde}) - \pi_{\text{KundenNr}}(\sigma_{\text{verkäuferName} \neq 'Mueller'}(\text{Kaufbeleg})))$

- d) Geben Sie die Kundennummern und Namen von denjenigen Kunden aus, die Produkte gekauft haben, die an ihrem Wohnort produziert werden.

$\pi_{\text{KundenNr, kundenName}} \sigma_{\text{kundenWohnort}=\text{produktionsOrt}}(\text{Kunde} \bowtie \text{Kaufbeleg} \bowtie \text{Produkt})$

alternativ: $\pi_{\text{KundenNr, kundenName}} (\text{Kunde} \bowtie (\sigma_{\text{kundenWohnort}=\text{produktionsOrt}} \text{Produkt}))$

Aufgabe 4: SQL

(32 Punkte)

Gegeben seien folgende Relationen:

Produkt (produktNr, produktName, preis, hersteller)

Verkaeuer (verkaeuerNr, verkaeuerName, gehalt, abteilung)

Kunde (kundenNr, kundenName, stadt)

Rechnung (rechnungNr, produktNr, anzahl, kundenNr, verkaeuerNr, datum)

1) Geben Sie die folgenden SQL-Anweisungen an:

a) Erzeugen Sie die Tabelle *Rechnung*. Wählen Sie geeignete Datentypen und berücksichtigen Sie die Primär- und Fremdschlüsselattribute.

```
CREATE TABLE Rechnung(  
    rechnungNr INTEGER,  
    produktNr INTEGER,  
    anzahl INTEGER,  
    kundenNr INTEGER,  
    verkaeuerNr INTEGER,  
    datum DATE,  
    CONSTRAINT RechnungPK  
        PRIMARY KEY (rechnungNr, produktNr),  
    CONSTRAINT produktFK  
        FOREIGN KEY (produktNr) REFERENCES Produkt  
    CONSTRAINT kundenFK  
        FOREIGN KEY (kundenNr) REFERENCES Kunde,  
    CONSTRAINT verkaeuerFK  
        FOREIGN KEY (verkaeuerNr) REFERENCES Verkaeuer );
```

b) Fügen Sie ein Mobiltelefon von Siemens mit dem Namen SL55 in die Produkt-Relation ein. Es soll 800 Euro kosten und die Produktnummer 7 haben.

```
INSERT INTO Produkt VALUES(7, 'SL55', 800.00, 'Siemens');
```

Formulieren Sie die folgenden beiden SQL-Anfragen unter Verwendung von mengen-orientierten Unteranfragen mit geeigneten Prädikaten (z.B. IN, EXISTS, ANY, SOME, ALL):

c) Geben Sie die Nummern und Namen aller Verkäufer aus, die Kunden aus Essen bedient haben.

```
SELECT verkaeuerNr, verkaeuerName  
FROM Verkaeuer NATURAL JOIN Rechnung  
WHERE kundenNr IN (SELECT kundenNr  
                    FROM Kunde  
                    WHERE stadt='Essen');
```

```
SELECT verkaeuerNr, verkaeuerName  
FROM Verkaeuer NATURAL JOIN Rechnung  
WHERE EXISTS (SELECT *  
              FROM Kunde  
              WHERE stadt='Essen' AND  
                    Kunde.kundenNr=Rechnung.kundenNr);
```

d) Geben Sie das (die) billigsten Produkt(e) von dem Hersteller „Siemens“ an.

```
SELECT *  
FROM Produkt  
WHERE preis ≤ ALL (SELECT preis  
                   FROM Produkt  
                   WHERE hersteller='Siemens');
```


2) Formulieren Sie die folgenden SQL-Anfragen unter Verwendung von geeigneten Aggregatfunktionen:

- a) Geben Sie die Gesamtzahl der von Kunden aus Essen gekauften Produkte an.

```
SELECT SUM(Anzahl) AS ProduktZahlEssen
FROM Kunde NATURAL JOIN Rechnung
WHERE Stadt='Essen' ;
```

- b) Geben Sie für jede Abteilung die Anzahl der dort angestellten Verkäufer an.

```
SELECT abteilung, COUNT(verkaeuferNr) As anzahlVerkaeufer
FROM Verkaeufer
GROUP BY abteilung;
```

- c) Geben Sie den minimalen und den maximalen Gehalt der Verkäufer der Elektroabteilung an.

```
SELECT MIN(gehalt) AS minGehaltElektro,
       MAX(gehalt) AS maxGehaltElektro
FROM Verkaeufer
WHERE abteilung = 'Elektroabteilung' ;
```

- d) Geben Sie für alle Verkäufer der Elektroabteilung, die weniger als 100 unterschiedliche Kunden bedient haben, ihre Verkäufersnummern, Verkäufersnamen und die Anzahl der von ihnen bedienten Kunden an.

```
SELECT verkaeuferNr, verkaeuferName,
       COUNT(DISTINCT kundenNr) AS kundenAnzahl
FROM Verkaeufer NATURAL JOIN Rechnung
WHERE abteilung = 'Elektroabteilung'
GROUP BY verkaeuferNr
HAVING COUNT(DISTINCT kundenNr) < 100 ;
```

Aufgabe 5: Synchronisationsverfahren

(14 Punkte)

Gegeben sei das folgende Schedule:

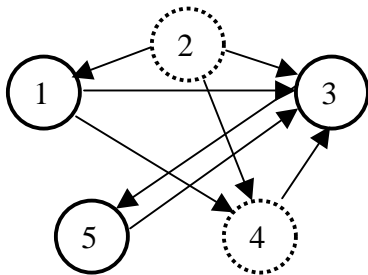
$R_5(y) R_3(y) W_2(z) R_1(x) W_1(z) R_2(x) R_4(z) W_3(z) R_4(x) W_5(y) W_3(y)$

1)

- Erstellen Sie für dieses Schedule den Abhängigkeitsgraphen.
- Ist sie serialisierbar (begründen Sie Ihre Antwort)?
Geben Sie wenn möglich die äquivalente serielle Schedule an.
- Hat dieses Schedule wirkungslose Transaktionen?
Nennen Sie alle wirkungslosen Transaktionen, falls vorhanden.
Wäre das Schedule serialisierbar, falls wirkungslose Transaktionen ausgelassen würden?

Lösung

a)



- Nicht serialisierbar, da Zyklus vorhanden.
- T2 und T4 sind wirkungslos. Nein, nicht serialisierbar, da Zyklus weiter vorhanden.

2) Welche (um die Sperren erweiterte) Ergebnisschedule würden von dem Zweiphasensperrprotokoll mit Preclaiming erzeugt?

Lösung

$R_5(y) R_3(y) W_2(z) R_1(x) W_1(z) R_2(x) R_4(z) W_3(z) R_4(x) W_5(y) W_3(y)$

T1: $rl_1(x), wl_1(z)$ T2: $rl_2(x), wl_2(z)$ T3: $wl_3(y), wl_3(z)$ T4: $rl_4(x), rl_4(z)$, T5: $wl_5(y)$

	Wartend	x	y	z
$R_5(y)$			$wl_5(y)$ $R_5(y)$	
$[R_3(y)]$	$R_3(y)$			
$W_2(z)$		$rl_2(x)$		$wl_2(z)$ $W_2(z)$ $ul_2(z)$
$R_1(x)$		$rl_1(x)$ $R_1(x)$ $ul_1(x)$		$wl_1(z)$
$W_1(z)$				$W_1(z)$ $ul_1(z)$
$R_2(x)$		$R_2(x)$ $ul_2(x)$		
$R_4(z)$		$rl_4(x)$		$rl_4(z)$ $R_4(z)$ $ul_4(z)$
$[W_3(z)]$	$W_3(z)$			
$R_4(x)$		$R_4(x)$ $ul_4(x)$		
$W_5(y)$			$W_5(y)$ $ul_5(y)$	
$> R_3(y)$			$wl_3(y)$ $R_3(y)$	$wl_3(z)$
$> W_3(z)$				$W_3(z)$ $ul_3(z)$
$W_3(y)$			$W_3(y)$ $ul_3(y)$	

Aufgabe 6: JDBC

(15 Punkte)

In einer Datenbank existiert die Tabelle mit dem folgenden Schema (alle Spalten sind vom Typ varchar(20)):

Studenten(Matrikelnr, Vorname, Nachname, Studienfach)

Geben Sie ein Java-Programm an, das zur Ausgabe des Ergebnisses der folgenden Abfrage mit Hilfe von JDBC notwendig ist:

```
SELECT * FROM Studenten;
```

Der Treiber des Herstellers ist in der Klasse "COM.ibm.db2.jdbc.app.DB2Driver" realisiert. Der URL der Datenbank ist „jdbc:db2:studivwt“. Bauen Sie die Verbindung unter Verwendung des Benutzernamens „administrator“ mit dem Passwort „adminPW“ auf. Geben Sie alle Datensätze auf die Standardausgabe (System.out.print()) aus. Es ist ausreichend, wenn Sie die Spalten Matrikelnr und Nachname ausgeben.

Lösung:

```
/*
Vgl. auch Folien für weitere Erläuterungen.
Codeausschnitt: */
//Treiber laden
try {
    Class.forName("COM.ibm.db2.jdbc.app.DB2Driver");
}
catch (ClassNotFoundException e) {
    /* Handle Exception */
}
//Verbindung zur Datenbank aufbauen
Connection con =
DriverManager.getConnection("jdbc:db2:studivwt","administrator",
"adminPW")
//Statement erzeugen
Statement stmt=con.createStatement();
//Abfrage ausführen
ResultSet rs = stmt.executeQuery("SELECT * FROM Studenten");

//Spalten auslesen
while(rs.next()){
    System.out.println("Kundennummer=" + rs.getString("Matrikelnr"));
    System.out.println("Kundennummer=" + rs.getString("Nachname"));
}
//ResultSet, Statement und Connection schließen.
rs.close();
stmt.close();
con.close();
```