Klausur Datenbanken
Wintersemester 2011/2012
Prof. Dr. Wolfgang May
8. Februar 2012, 14-16 Uhr
Bearbeitungszeit: 90 Minuten

Vorname:

Nachname:

Matrikelnummer:

Studiengang:

Bei der Klausur sind **keine Hilfsmittel** (Skripten, Taschenrechner, etc.) erlaubt. Handies müssen ausgeschaltet sein. Papier wird gestellt. Benutzen Sie nur die **ausgeteilten**, zusammengehefteten **Blätter** für Ihre Antworten. Schreiben Sie mit blauem/schwarzem Kugelschreiber, Füller, etc.; Bleistift ist nicht erlaubt.

Zum **Bestehen** der Klausur sind **45** Punkte hinreichend.

☐ meine Note soll mit Matrikelnummer so bald wie möglich auf der Vorlesungs-Webseite veröffentlicht werden.

☐ meine Note soll nicht veröffentlicht werden; ich erfahre sie dann aus FlexNever oder beim zuständigen Prüfungsamt.

	Max. Punkte	Schätzung für "4"
Aufgabe 1 (ER-Modell)	15	11
Aufgabe 2 (Transformation in das Relationale Modell)	16	10
Aufgabe 3 (SQL und Relationale Algebra)	40	22
Aufgabe 4 (Verschiedenes)	19	11
Summe	90	54

Note:

Themenstellung: Supermarkt

Alle Klausuraufgaben basieren auf einem gemeinsamen "Auftrag": In der Klausur soll eine Datenbank eines Supermarktes, die den Warenbestand sowie die Verkäufe erfasst und auch statistische Auswertungen ("Data Mining") über das Kundenverhalten erlaubt, entworfen werden.

1. Zu allen Warenbezeichnungen ist der Preis pro Einheit (kg, Packungen, ...) gespeichert.

   Äpfel kosten 1.99 pro kg, *Kekse* kosten 0.79 pro Packung, *Würstchen* kosten 0.50 pro Stück, *Chips* kosten 0.99 pro Packung, *Bier* kostet 0.50 pro Flasche, Katzenfutter kostet 2.00 pro Packung.

2. Zu allen Warenbezeichnungen wird gespeichert, wieviele Einheiten im Regal sind, und in welcher Abteilung des Ladens sich dieses befindet.

   In der Obstabteilung (=Abteilung 1) befinden sich 12 kg *Äpfel*. In den Regalen in Abteilung 23 befinden sich 63 Packungen *Kekse* und 5 Packungen *Chips*.

3. Zu allen Warenbezeichnungen wird gespeichert, wieviele Einheiten sich im Lager befinden.

   Im Lager befinden sich u.a. 183 Packungen *Kekse* und 10 Packungen *Chips*.

4. Viele Kunden haben eine Kundenkarte des Ladens. Für jede Kundenkarte ist die Kartennummer sowie Name und Adresse des Kunden gespeichert. Es wird angenommen, dass es keine zwei Kunden mit demselben Namen gibt.

   Karl Napf wohnt am *Marktplatz 1* in *Göttingen* und hat die Kundenkarte mit der Nummer 4711. *Emma Napf* wohnt an derselben Adresse und hat die Kundenkarte mit der Nummer 4712.

5. Wenn ein Kunde nach dem Einkaufen an die Kasse kommt, werden alle Waren gescannt und über ihren Barcode oder manuell erfasst. Der Kunde erhält den Kassenzettel, auf dem alle Waren und die Anzahl gekaufte Einheiten gelistet sind.

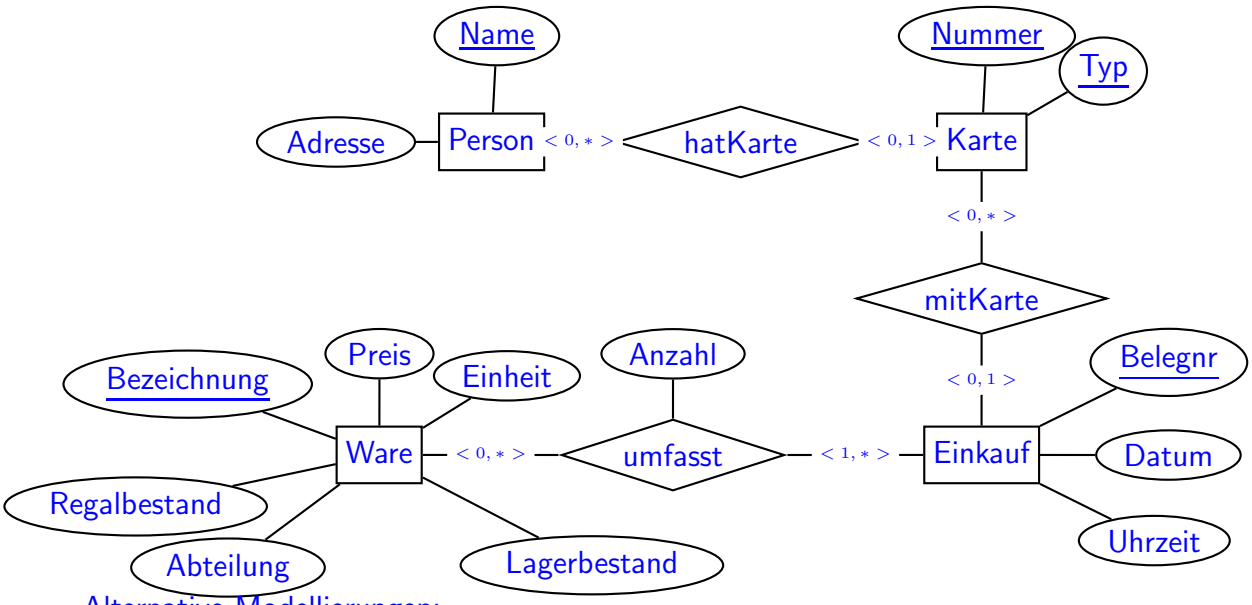
   - Diese Daten werden auch alle –mit der eindeutigen Kassenbelegnummer des Einkaufs– gespeichert.
   - Zu jedem Einkauf wird Datum und Uhrzeit, sowie die Nummer der Kasse gespeichert.
   - Wenn der Kunde bar bezahlt, werden keine weiteren Daten gespeichert.
   - Wenn der Kunde mit ec-Karte bezahlt, wird die Kartennummer gespeichert. Auf diese Weise lassen sich später z.B. alle Einkäufe mit dieser Kartennummer miteinander in Beziehung setzen.
   - Wenn der Kunde mit Kundenkarte bezahlt, wird ebenfalls die Kartennummer gespeichert.

   Karl Napf hat am 6.2.2012 um 17:34 an Kasse 3 folgendes gekauft und mit seiner Kundenkarte Nr. 4711 bezahlt: 2 Flaschen Bier, eine Tüte Chips, 2 Würstchen und eine Packung Katzenfutter. Er bekam den Kassenzettel mit Kassenbelegnummer 57612.

   Eine unbekannte Person hat am 6.2.2012 um 17:35 mit der Kassenbelegnummer 57613 an Kasse 2 eine Packung Kekse gekauft und bar bezahlt.

Aufgabe 1 (ER-Modell [15 Punkte])
Entwickeln Sie ein ER-Modell für das Szenario. Geben Sie darin die Schlüsselattribute sowie die Beziehungskardinalitäten an.

Lösung



Alternative Modellierungen:

- *Person* und *Adresse* als Attribute zu Karte geht auch (man hat dann allerdings bei Personen, die mehrere Kundenkarten besitzen die Adresse redundant gespeichert).
- Den Kartentyp kann man weglassen, wenn man davon ausgeht, dass Kundenkarten andere Nummern haben als ec-Karten. Die Karten"nummern" von ec-Karten sind Zeichenketten, z.B. der Form "123456/123" wobei der hintere Teil die Bankfiliale angibt.
- Attribut "Kartentyp" weglassen, und "ec-karte" als Attribut zu Einkauf (in dem Fall weiss man ja den Namen der Person nicht). Nur Kundenkarten werden dann in einem Entitätstyp "Kundenkarte" mit Attributen "Nummer, Name, Adresse" berücksichtigt (wobei man wieder bei Personen, die mehrere Kundenkarten besitzen, die Adresse redundant speichert).
  Diese Modellierung hat den Nachteil, dass man, wenn eines Tages ein Kartentyp dazukommt (z.B. Kreditkarte) das Schema ändern muss, während bei der obigen Modellierung nur der Kartentyp-Wert "Kreditkarte" neu auftritt.

  Diese Modellierung wurde von der Mehrzahl der Teilnehmer gewählt. Sie ist auch bei den meisten Anfragen einfacher, weil man den Kartentyp nicht mitschleppen muss.

Aufgabe 2 (Transformation in das Relationale Modell [16 Punkte])

a) **Lösen Sie diesen Aufgabenteil auf dem *letzten* Blatt und trennen dieses ab (und geben es am Ende mit ab!).** Dann haben Sie dieses Blatt separat zugreifbar um später damit die Aufgaben 2b, und 3 (SQL, Relationale Algebra+SQL) zu lösen.

Geben Sie an, welche Tabellen (mit Attributen, Schlüsseln etc.) Ihre Datenbank enthält (keine SQL `CREATE TABLE`-Statements, sondern einfach grafisch). (10 P)

Markieren Sie dabei auch Schlüssel (durch unterstreichen) und Fremdschlüssel (durch überstreichen).

Geben Sie die Tabellen mit jeweils mindestens zwei Beispieltupeln (z.B. denen, die sich aus dem Aufgabentext ergeben, und weiteren erfundenen) an.

Lösung

- Tabellen für die Entitätstypen "Person" und "Karte" können zusammengefasst werden.

Karte

Nummer	Typ	Name	Strasse	Ort
4711	Kundenkarte	Karl Napf	Marktplatz 1	Göttingen
4712	Kundenkarte	Emma Napf	Marktplatz 1	Göttingen
123456/123	ec-Karte	NULL	NULL	NULL

Ware

Bezeichnung	Preis	Einheit	Abteilung	Regalbestand	Lagerbestand
Äpfel	1.99	kg	1	12	25
Kekse	0.79	Pck	23	63	183
Chips	0.99	Pck	23	5	10
:	:	:	:	:	:

Einkauf

Belegnr	Datum	Uhrzeit	Kasse	Karte	Kartentyp
57612	6.2.2012	17:34	3	4711	Kundenkarte
57613	6.2.2012	17:35	2	NULL	NULL
57614	6.2.2012	17:36	4	123456/123	ec-Karte
:	:	:	:	:	:

gekauft

Belegnr	Ware	Anzahl
57612	Bier	2
57612	Chips	1
57613	Kekse	1
:	:	:

Alternative:

Bei Modellierung "Einkauf" mit Attribut "ec-Karte" und Beziehung "Kundenkarte":

Einkauf					
Belegnr	Datum	Uhrzeit	Kasse	ec-Karte	Kundenkarte
57612	6.2.2012	17:34	3	NULL	4711
57613	6.2.2012	17:35	2	NULL	NULL
57614	6.2.2012	17:36	4	123456/123	NULL
:	:	:	:	:	:

b) Geben Sie das `CREATE TABLE`-Statement für diejenige Tabelle (bzw. die Tabellen), in der bei Ihnen die Daten über die gekauften Waren abgespeichert sind, so vollständig wie möglich an (6 P).

Lösung

```
CREATE TABLE gekauft                                          Basis 2P
( Belegnr NUMBER REFERENCES Einkauf(Belegnr),                 1P FKEY
  Ware    VARCHAR2(20) REFERENCES Ware(Bezeichung),           1P FKEY
  Anzahl  NUMBER NOT NULL CHECK (anzahl > 0),     1/2P NOT NULL + 1/2P CHECK
CONSTRAINT gekauftkey PRIMARY KEY (Belegnr, Ware))            1P PKEY
```

Aufgabe 3 (SQL und Relationale Algebra [40 Punkte])
Verwenden Sie für diese Aufgabe die von Ihnen entworfene relationale Datenbasis. Keine der Antworten soll Duplikate enthalten.

a) Geben Sie **eine SQL-Anfrage und einen Algebra-Ausdruck** an, die alle ec-Kartennummern angibt, mit denen am 6.2.2012 an Kasse 2 bezahlt wurde. (2+2 P)

Lösung

```
select distinct karte
from einkauf
where datum = "6.2.2012"
  and kasse = 2
  and kartentyp="ec"
```

Bei Modellierung "Einkauf" mit Attribut "ec-Karte" und Beziehung "Kundenkarte":

```
select distinct ec_karte
from einkauf
where datum = "6.2.2012"
  and kasse = 2
  and ec_karte is not null      -1/2P wenn das fehlt
```

$$\pi[\text{karte}]$$
$$\sigma[\text{datum} = \text{“6.2.2012”} \text{ and } \text{kasse} = 2 \text{ and } \text{kartentyp} = \text{“ec”}]$$
$$\text{einkauf}$$

b) Geben Sie **eine SQL-Anfrage** an, die alle Warenbezeichnungen ergibt, die am 6.2.2012 bei mindestens 20 Einkäufen an Kasse 2 gekauft wurden. (4 P)

Lösung

```
select ware
from Einkauf, gekauft
where Einkauf.belegnr = gekauft.belegnr
  and datum = "6.2.2012"
  and kasse = 2
group by ware
having count(*) > 19
```

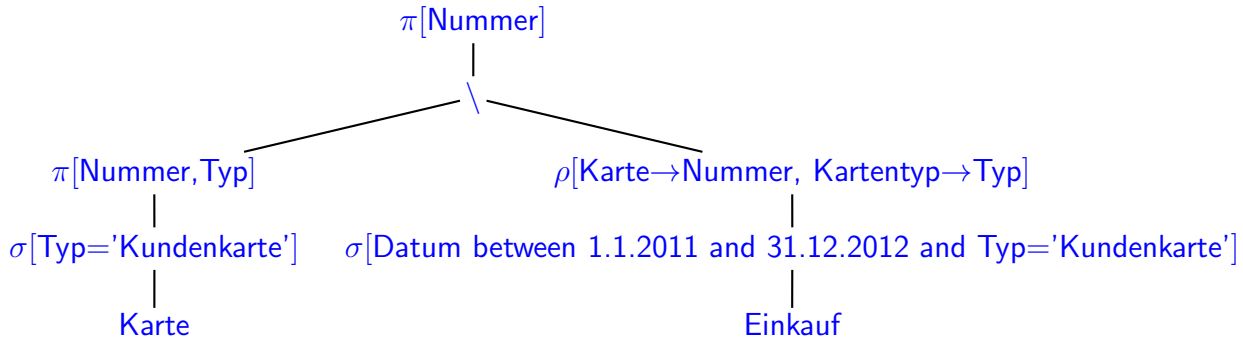
```
select bezeichnung     (oder select distinct ware from einkauf)
from ware
where (select count(*)
       from Einkauf, gekauft
       where Einkauf.belegnr = gekauft.belegnr
```

```
and gekauft.ware = ware.bezeichnung
and datum = "6.2.2012"
and kasse = 2) > 19
```

c) Geben Sie **eine SQL-Anfrage und einen Algebra-Ausdruck oder -Baum** an der die Nummern derjenigen Kundenkarten ausgibt, die im Jahr 2011 nicht verwendet wurden (3+3 P).

Lösung

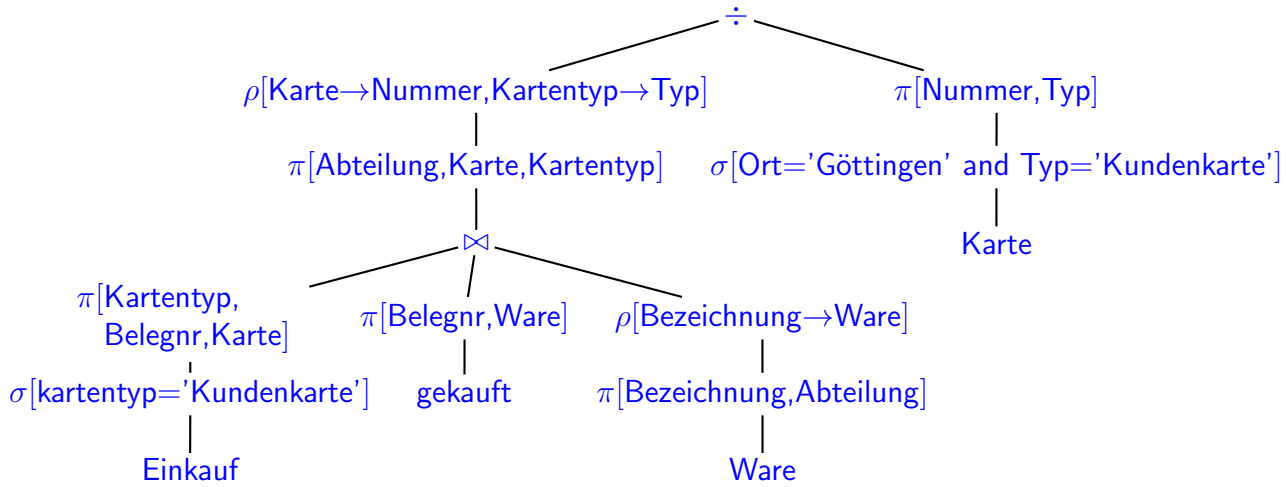
```
select nummer
from karte
where typ='Kundenkarte'
  and not exists (select *
                  from Einkauf
                  where Datum between 1.1.2011 and 31.12.2011
                    and karte.Nummer = Einkauf.Karte
                    and karte.Typ = Einkauf.Kartentyp)

select nummer
from karte
where typ='Kundenkarte'
  and nummer not in (select nummer
                     from Einkauf
                     where Datum between 1.1.2011 and 31.12.2011
                       and karte.Nummer = Einkauf.Karte
                       and karte.Typ = Einkauf.Kartentyp)

(select nummer
 from karte
 where typ='Kundenkarte')
minus
(select nummer
 from karte
 where karte.typ='Kundenkarte'))
   and exists (select *
               from Einkauf
               where Datum between 1.1.2011 and 31.12.2011
                 and karte.Nummer = Einkauf.Karte
                 and karte.Typ = Einkauf.Kartentyp))
```



d) Geben Sie **eine SQL-Anfrage und einen Algebra-Ausdruck oder -Baum** an, der die Nummern derjenigen Abteilungen ausgibt, in denen mit jeder Kundenkarte, deren Inhaber in Göttingen wohnt, mindestens ein Artikel eingekauft wurde. (4+4P)

Lösung

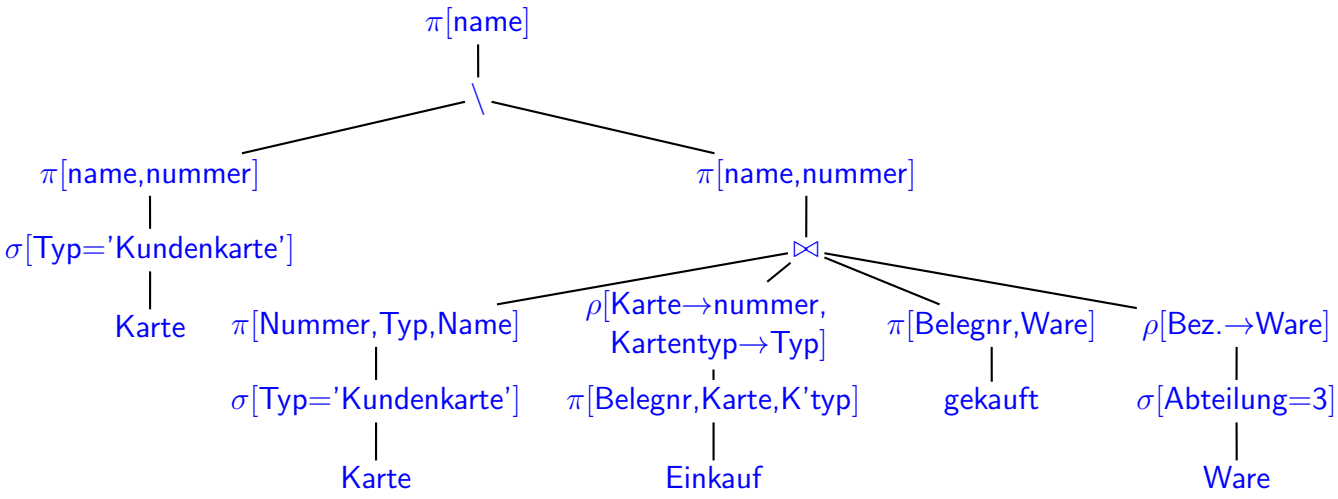
```
select distinct bereich
from ware w1
where not exists
  (select *
   from karte k
   where typ='Kundenkarte'
     and ort='Goettingen'
     and not exists   -- einkauf mit dieser karte in dieser Abteilung
         (select *
          from Einkauf e, gekauft g, ware w2
          where e.belegnr = g.belegnr
            and g.Ware = w2.Bezeichnung
            and w2.Abteilung = w1.Abteilung
            and e.Karte = k.Nummer
            and e.Kartentyp=k.Typ))
```



e) Geben Sie **eine SQL-Anfrage und einen Algebra-Ausdruck oder -Baum** an, der die Namen derjenigen Kunden ausgibt, die eine Kundenkarte besitzen, und mit dieser nie einen Artikel aus der Fleischtheke (=Abteilung 3) bezahlt haben (4+4P).

Lösung

```
select distinct name
from Karte
where typ="kundenkarte"
  and not exists
 (select *
  from gekauft, Einkauf, Ware
  where gekauft.Belegnr = Einkauf.Belegnr
    and gekauft.Ware = Ware.Bezeichnung
    and Ware.Abteilung = 3
    and Einkauf.Karte = Karte.Nummer
    and Einkauf.Kartentyp = Karte.Typ)

(select name from Karte
  where typ="kundenkarte")
minus
  (select name
   from Karte, Einkauf, gekauft, Ware
   where Einkauf.Kartentyp = Karte.Typ
     and Karte.Typ='Kundenkarte'
     and Karte.Nummer = Einkauf.Karte
     and Einkauf.Kartentyp = Karte.Typ
     and Einkauf.Belegnr = gekauft.Belegnr
     and gekauft.Ware = Ware.Bezeichnung
     and Ware.Abteilung = 3)
```



Hinweis: wenn eine Person 2 Kundenkarten hat, von denen sie nur mit einer mal einen Artikel aus der Fleischtheke eingekauft hat, muss die Person im Ergebnis enthalten sein. (-1/2 P)

f) Geben Sie **eine SQL-Anfrage** an, die die Bezeichnungen aller Waren ausgibt, von denen die Anzahl der im Lager und im Regal vorhandenen Einheiten geringer als die durchschnittliche pro Tag von dieser Ware verkaufte Menge ist. (6 P)

Lösung

```
SELECT Bezeichnung
FROM Ware
WHERE (lagerbestand + regalbestand) <
  (SELECT sum(Anzahl)
   FROM gekauft
   WHERE Ware = Ware.name) / (SELECT count(distinct datum)
                              FROM Einkauf)
```

Der folgende Ausdruck ist SQL-mäßig interessant, ergibt aber ein (etwas) falsches Ergebnis:

```
SELECT Bezeichnung
FROM Ware
WHERE (lagerbestand + regalbestand) <
  (SELECT avg(sum(Anzahl))
   FROM gekauft, Einkauf
   WHERE gekauft.Belegnr=Einkauf.Belegnr
     and gekauft.Ware = Ware.name
   GROUP BY Einkauf.datum)
```

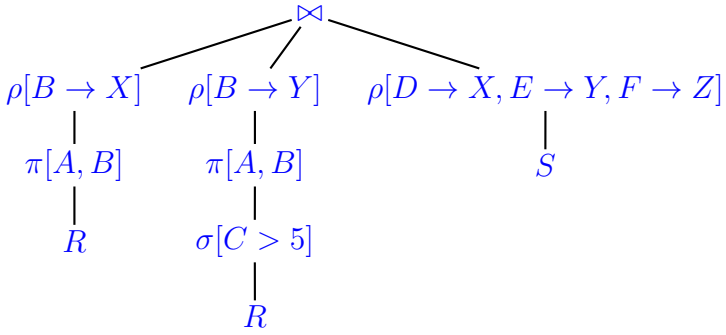
Sum(Anzahl) wird ueber jede Gruppe (=jeden Tag einzeln) gebildet, avg(...) dann ueber alle Gruppen (= ueber alle vorkommenden Tage). Wenn eine Ware nicht an jedem Tag verkauft wird (z.B. nur samstags), ergibt sich ein zu hoher "Durchschnitt". (-1/2 P)

g) **Etwas Theorie:** Gegeben sind zwei Relationen $R(A, B, C)$ und $S(D, E, F)$. Die Attribute $R.A$, $R.B$, $S.D$, $S.E$ sind Zeichenketten, $R.C$ und $S.F$ sind numerisch.

Geben Sie einen **Algebra-Ausdruck oder -Baum** an, der die folgendermassen spezifizierte Tupelmenge selektiert (4 P):

$\{t \in \text{Tup}(A, X, Y, Z)$ so dass
(es gibt ein Tupel $t_1 \in R$ mit $t[A] = t_1[A]$ und $t[X] = t_1[B]$) und
(es gibt ein Tupel $t_2 \in R$ mit $t[A] = t_2[A]$ und $t[Y] = t_2[B]$ und $t_2[C] > 5$) und
(es gibt ein Tupel $t_3 \in S$ mit $t[X] = t_3[D]$ und $t[Y] = t_3[E]$ und $t[Z] = t_3[F]$) $\}$

Lösung



Aufgabe 4 (Verschiedenes [19 Punkte])
Der einschlägig vorbestrafte Ede K.[1] ("Brechstangen-Ede") hat sich während seines letzten Gefängnisaufenthalts fortgebildet und auf e-Business umgesattelt ("eEde"). Unter anderen hat er die Kartenleser der betrachteten Supermarktkette manipuliert, so dass sie ihm Kartennnummer und PIN per Funk übermitteln. Ede speichert diese Daten in einer eigenen relationalen Datenbank.

a) Geben Sie das SQL-Statement an, mit dem er seine entsprechende Tabelle erzeugt (2 P).

Lösung

```
CREATE TABLE pins
(cardnumber VARCHAR2(20) PRIMARY KEY,
 pin        VARCHAR2(6)  NOT NULL );
```

b) Geben Sie das SQL-Statement an, das in Edes Datenbank ausgeführt werden muss, wenn ein neuer Eintrag, z.B. mit der Kartennummer "123456/123" und der PIN "0802" angelegt werden soll (2 P).

Lösung

```
INSERT INTO TABLE pins VALUES('123456/123', '0802');
```

Nun wieder weg von Ede, zur "normalen" Datenbank des Supermarktes:

c) Mit den Kassenbuchungen wird der Bestand der noch im Regal befindlichen Waren ständig automatisch aktualisiert. Geben Sie das SQL-Statement an, das auf der Datenbank ausgeführt werden muss, wenn der Verkauf von 5 Packungen Chips an der Kasse registriert wird. (3 P).

Lösung

```
UPDATE Ware
SET Regalbestand = Regalbestand - 5
WHERE Bezeichnung = 'Chips'

UPDATE Ware
SET Regalbestand = (SELECT Regalbestand
                    FROM Ware
                    WHERE Bezeichnung = 'Chips') + 5
WHERE Bezeichnung = 'Chips'
```

[1] Ähnlichkeiten des Namens mit einer Supermarktkette, bei der im Herbst 2011 in Göttingen und anderen norddeutschen Städten mehrfach per Skimming ec-Karten-PINs ausgelesen wurden sind rein zufällig

d) Warum ist es in der Anwendung wichtig, dass das verwendete Datenbanksystem Transaktionen unterstützt? (3 P).

Lösung Die Regal- und Lagerbestände werden von verschiedenen Agenten/Prozessen (mehrere Kassen, mehrere Mitarbeiter, die Waren vom Lager in die Regale einräumen) potentiell parallel geändert.

e) **(3-Ebenen-Architektur)**

- Was bedeutet der Begriff "Index" auf der physikalischen Ebene der 3-Ebenen-Architektur? (2 P)
- Welche häufigen Arten von Indexen gibt es in Datenbanken (2 P)?
- Beschreiben Sie eine Art etwas genauer (2 P).
- Geben Sie für das von Ihnen in Aufgabe 2 entworfene relationale Modell ein Beispiel an, auf welcher Spalte/Spalten man einen solchen Index hätte, und geben Sie eine Anfrage an, die diesen nutzt (mit kurzer Beschreibung, wie sie ihn nutzt) (3 P)

Lösung

- Ein Index ist eine zusätzlich abgelegte Datenstruktur, die die Werte einer oder mehrere Spalten/Attribute einer Tabelle/Relation indiziert um einen schnelleren Suchzugriff zu ermöglichen.
- (B*)-Baum- und Hash-Indexe.
- Ein Baumindex enthält alle Werte des Attributes in einem Suchbaum. Von dort aus wird dann auf das/die Tupel, die den entsprechenden Attributwert enthalten, verwiesen.
  Ein Hashindex weist jedem Wert eines Attributes eine Hash-Partition zu. Aus dieser wird für jeden der auf die Partition abgebildeten Werte auf das/die Tupel, die den entsprechenden Attributwert enthalten, verwiesen.
- z.B. Index über gekauft.Belegnr (was ja Fremdschlüssel auf Einkauf.Belegnr ist). Die Anfrage

```
select ware, anzahl
from gekauft
where belegnr="57613"
```

  würde diesen Index nutzen, um effizient alle zu diesem Beleg gekauften Artikel aufzuzählen.