Gruppe **A**

Bitte tragen Sie **SOFORT** und **LESERLICH** Namen und Matrikelnr. ein, und legen Sie Ihren Ausweis bereit.

| PRÜFUNG AUS | **MUSTERLÖSUNG** | 21.02.2023 **GRUPPE A** |
|---|---|---|
| Matrikelnr. | Familienname | Vorname |
| | | |

Arbeitszeit: 90 Minuten. Lösen Sie die Aufgaben auf den vorgesehenen Blättern; Lösungen auf Zusatzblättern werden nicht gewertet. **Viel Erfolg!**

| **Aufgabe** | 1 | 2 | 3 | 4 | 5 | 6 | Σ |
|---|---|---|---|---|---|---|---|
| **Max. Punkte** | 12 | 12 | 10 | 12 | 12 | 12 | 70 |
| *Punkte* | | | | | | | |

Bitte die Heftklammer nicht entfernen!
Matrikelnr/Namen auf jedes Blatt eintragen, es erleichtert uns die Punkte einzutragen.

---

**Achtung!**

Für sämtliche Fragen mit Ankreuzmöglichkeiten gilt: Ankreuzen alleine gibt keine Punkte!

Punkte gibt es nur in Zusammenhang mit geforderter Erklärung/Beispiel/...!

---

**Notation:**

In den Aufgaben 1 – 3 wird die folgende (aus der Vorlesung bekannte) Notation für Transaktionen $T_i$ verwendet:

- $r_i(O)$ und $w_i(O)$: Lese- bzw. Schreibzugriff von $T_i$ auf Objekt $O$.
- $b_i$, $c_i$, $a_i$: Beginn (`BEGIN OF TRANSACTION`), Commit (`COMMIT`) bzw. Abbruch (`ABORT`/`ROLLBACK`) von $T_i$.

Die Indizes $_i$ können weggelassen werden, wenn klar ist zu welcher Transaktion eine Operation gehört.

Des weiteren wird das aus der Vorlesung bekannte Format für Logeinträge verwendet:

`[LSN, TA, PageID, Redo, Undo, PrevLSN]` für "normale" Einträge, bzw.
`[LSN, TA, BOT, PrevLSN]` für `BOT` Log-Einträge und
`[LSN, TA, COMMIT, PrevLSN]` für `COMMIT` Einträge.

Kompensations Logeinträge (Compensation Log Records) haben das Format
⟨`LSN, TA, PageID, Redo, PrevLSN, UndoNextLSN`⟩ bzw.
⟨`LSN, TA, BOT, PrevLSN`⟩

Dabei stellt `LSN` die Log-Sequence Nummer dar, `TA` die Transaktion, `PageID` ist die veränderte Seite, `Redo` und `Undo` die für das Redo bzw. Undo benötigten Informationen, `UndoNextLSN` ist die LSN des nächsten Logeintrags der selben Transaktion welcher zurückgesetzt werden soll, und `PrevLSN` die LSN des vorherigen Logeintrags derselben Transaktion.

Im Falle logischer Protokollierung sollen die Änderungen zum aktuellen Datenbestand nur mittels *Addition* bzw. *Subtraktion* angegeben werden, z.B. [·, ·, ·, $X$+=$d_1$, $X$-=$d_2$, ·].

**MatrikelNr:** _____ **Nachname:** _____

**Aufgabe 1:** Eigenschaften von Transaktionen (12)

Gehen Sie davon aus, dass in einem DBMS die folgenden Lock-Levels implementiert sind, und jede Transaktion sich aussuchen kann, in welchem dieser Level sie ausgeführt werden soll.

- **L1:** Share-Locks werden vor jeder Leseoperation angefragt und sofort nach der Leseoperation wieder freigegeben (kein 2PL); 2PL für Exclusive-Locks (unabhängig von Freigaben von Share-Locks).
- **L2:** Gemeinsames 2PL für Share- und Exclusive-Locks, für Exclusive-Locks wird jedoch strenges 2PL verwendet. Lock-Upgrades (eines Share-Locks in einen Exclusive-Lock) sind möglich und zählen nicht als Freigabe.

Beantworten Sie die folgenden Fragen mit "Ja" oder "Nein". Bei *"Ja"* geben Sie bitte eine Historie an (Operationen $b_i$, $c_i$, $r_i(O)$, $w_i(O)$) welche mit dem angenommenen Lock-Level konform ist, und die gesuchte Eigenschaft erfüllt. Bei *"Nein"* begründen Sie Ihre Antwort kurz (1-2 Sätze).

*Hinweis:* Sie brauchen Sperranforderungen und Freigaben nicht angeben – achten Sie nur darauf, dass es für Ihre Historie eine entsprechende gültige Folge von Sperren und Freigaben gibt.

a) Betrachten Sie die Transaktion

$$T_1 \colon b_1, r_1(A), w_1(A), w_1(B), r_1(A), r_1(B), r_1(A), c_1$$

und eine beliebige Transaktion $T_2$. Nehmen Sie an dass $T_1$ im Level L1 läuft und $T_2$ im Level L2.

i) Kann es zu einem *Lost-Update* von $T_2$ kommen? (4 Punkte)

Lost-Update von $T_2$: ⊗ ja ◯ nein

Mögliche Lösung: .....

$b_1, b_2, r_1(A), r_2(A), w_2(A), c_2, w_1(A), w_1(B), r_1(A), r_1(B), r_1(A), c_1$ .....

.....

ii) Kann es zu einem *Unrepeatable Read* von $T_1$ kommen ($T_1$ liest)? (4 Punkte)

Unrepeatable Read von $T_1$: ⊗ ja ◯ nein

Mögliche Lösung: .....

$b_1, b_2, r_1(A), w_1(A), w_1(B), r_1(A), w_2(A), c_2, r_1(B), r_1(A), c_1$ .....

.....

b) Betrachten Sie zusätzlich zu $T_1$ aus a) die Transaktion

$$T_3 \colon b_3, w_3(C), r_3(B), w_3(B), r_3(A), c_3$$

welche im Level L2 läuft. (4 Punkte)

Gibt es eine gültige Historie der beiden Transaktionen welche **nicht** rücksetzbar ist?

Gültige Historie die **nicht** rücksetzbar ist: ⊗ ja ◯ nein

$b_1, b_3, r_1(A), w_1(A), w_1(B), r_1(A), r_1(B), w_3(C), r_3(B), w_3(B), r_3(A), c_3, r_1(A), c_1$ .....

.....

.....

**MatrikelNr:** _____ **Nachname:** _____

**Aufgabe 2:** Protokollierung und Wiederanlauf (Logging und Recovery) (12)

Betrachten Sie die unten angegebenen Logeinträge der vier Transaktionen $T_1$, $T_2$, $T_3$ und $T_4$.
Führen Sie anhand dieser Log-Einträge ein Recovery (nach dem ARIES Verfahren) durch.

**Logeinträge (Archiv)**

| | | | |
|---|---|---|---|
| [#1, $T_1$, BOT, | | | #0] |
| [#2, $T_4$, BOT, | | | #0] |
| [#3, $T_4$, $P_D$, | D+=3, | D–=3, | #2] |
| [#4, $T_2$, BOT, | | | #0] |
| [#5, $T_2$, $P_A$, | A–=2, | A+=2, | #4] |
| [#6, $T_2$, $P_B$, | B–=4, | B+=4, | #5] |
| [#7, $T_3$, BOT, | | | #0] |
| [#8, $T_4$, $P_D$, | D+=1, | D–=1, | #3] |
| [#9, $T_3$, $P_C$, | C+=9, | C–=9, | #7] |
| [#10, $T_4$, $P_A$, | A+=2, | A–=2, | #8] |
| [#11, $T_4$, $P_B$, | B–=2, | B+=2, | #10] |
| [#12, $T_1$, $P_B$, | B+=4, | B–=4, | #1] |
| [#13, $T_2$, $P_A$, | A+=15, | A–=15, | #6] |
| ⟨#14, $T_4$, $P_B$, | B+=2, | #12, | #10⟩ |
| [#15, $T_1$, COMMIT, | | | #12] |
| ⟨#16, $T_4$, $P_A$, | A–=2, | #14, | #8⟩ |
| [#17, $T_3$, $P_C$, | C+=3, | C–=3, | #9] |

Seiten im Hintergrundspeicher:

| $P_A$ LSN: #5 |
|---|
| $A = 20$ |

| $P_B$ LSN: #6 |
|---|
| $B = 23$ |

| $P_C$ LSN: #9 |
|---|
| $C = 40$ |

| $P_D$ LSN: #8 |
|---|
| $D = 17$ |

a) Bestimmen Sie LSNs für $P_A$, $P_B$, $P_C$ und $P_D$ und die Werte $A$, $B$, $C$, $D$ nach der Redo-Phase. (4 Punkte)

| $P_A$ LSN: #16 |
|---|
| $A =$ 35 |

| $P_B$ LSN: #14 |
|---|
| $B =$ 27 |

| $P_C$ LSN: #17 |
|---|
| $C =$ 43 |

| $P_D$ LSN: #8 |
|---|
| $D =$ 17 |

b) Führen Sie die Undo-Phase aus. Erzeugen Sie die Compensation Log Records (CLRs). Nutzen Sie die untenstehenden Zeilen. Es kann sein, dass Sie nicht alle Zeilen benötigen. Verwenden Sie das oben beschriebene Format. (6 Punkte)

⟨#18, $T_3$, $P_C$, C-=3, #17, #9⟩ .....

⟨#19, $T_2$, $P_A$, A-=15, #13, #6⟩ .....

⟨#20, $T_3$, $P_C$, C-=9, #18, #7⟩ .....

⟨#21, $T_4$, $P_D$, D-=1, #16, #3⟩ .....

⟨#22, $T_3$, BOT, #20⟩ .....

⟨#23, $T_2$, $P_B$, B+=4, #19, #5⟩ .....

⟨#24, $T_2$, $P_A$, A+=2, #23, #4⟩ .....

⟨#25, $T_2$, BOT, #24⟩ .....

⟨#26, $T_4$, $P_D$, D-=3, #21, #2⟩ .....

⟨#27, $T_4$, BOT, #26⟩ .....

c) Geben Sie die Werte für A, B, C und D nach Beendigung der Undo-Phase an. (2 Punkte)

A: 22 ..... B: 31 ..... C: 31 ..... D: 13 .....

**MatrikelNr:** _____ **Nachname:** _____

**Aufgabe 3:** Sperrprotokolle (Locking) (10)

*Gegeben sind Folgen von Anfragen zu Exclusive- und Share-Sperren, abgekürzt $X_i(O)$ bzw. $S_i(O)$, Freigaben von Sperren, abgekürzt $relX_i(O)$ bzw. $relS_i(O)$, Lese- und Schreiboperationen, abgekürzt $r_i(O)$ bzw. $w_i(O)$, Beginn der Transaktion, abgekürzt $b_i$, Ende der Transaktion, abgekürzt $c_i$ bzw. $a_i$. Bei Einträgen in der selben Zeile spielt die zeitliche Anordnung der betroffenen Operationen keine Rolle, es kann eine beliebige Reihenfolge angenommen werden. Ein Lock-Upgrade (Shared-Sperre auf Exclusive-Sperre) ist erlaubt, wir nehmen an, dass dann nur die Exclusive-Sperre freigegeben werden muss.*

**(3a) 2-Phasen Sperrprotokoll (2PL): Protokoll korrekt?**
Geben Sie für jede der fünf Transaktionen $T_1$, $T_2$, $T_3$, $T_4$ und $T_5$ an, ob sie das *2-Phasen Sperrprotokoll (2PL)* befolgt. Falls nicht, beschreiben Sie wodurch das 2PL verletzt wird. (5 Punkte)

| $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ |
|---|---|---|---|---|
| $b_1$ | $b_2$ | $b_3$ | $b_4$ | $b_5$ |
| | $S_2(A)$ | | $S_4(A)$ | $X_5(C)$ |
| $S_1(A)$ | $r_2(A)$ | | | $w_5(C)$ |
| $r_1(A)$ | | | | |
| | | | | $X_5(B)$ |
| | $r_2(A)$ | | | $relX_5(C)$ |
| | $X_2(C)$ | | $r_4(A)$ | $w_5(B)$ |
| | $relS_2(A)$ | | | |
| | $w_2(C)$ | | | $relX_5(B)$ |
| $S_1(B)$ | $w_2(C)$ | | $S_4(B)$ | |
| | $relX_2(C)$ | | $r_4(B)$ | |
| | | | $X_4(C)$ | $X_5(D)$ |
| $r_1(B)$ | | | $w_4(C)$ | $w_5(D)$ |
| | | $S_3(C)$ | | $relX_5(D)$ |
| $X_1(D)$ | | $r_3(C)$ | | $c_5$ |
| $w_1(D)$ | $c_2$ | $X_3(C)$ | $relS_4(B)$ | |
| $relX_1(D)$ | | $w_3(C)$ | $relS_4(A)$ | |
| $relS_1(B)$ | | $relX_3(C)$ | $relX_4(C)$ | |
| $c_1$ | | $a_3$ | $c_4$ | |

$T_1$ `released A nicht` .....

$T_2$`: korrektes 2PL.` .....

$T_3$ `sperrt` $C$ `shared (SL),` .....

`obwohl` $T_4$ `exklusive Sperre hält` .....

$T_4$`: korrektes 2PL.` .....

$T_5$`: Locked D nach release auf B` .....

.....

.....

.....

.....

.....

.....

.....

*Hinweis:* Betrachten Sie nicht nur jede Transaktion für sich.

**MatrikelNr:** _____ **Nachname:** _____

**(3b) Striktes 2-Phasen Sperrprotokoll (Strikt 2PL): Sperren bestimmen**

*Zur Kontrolle der Nebenläufigkeit von Zugriffen soll das Strikte 2-Phasen Sperrprotokoll verwendet werden. Unten ist eine Historie gegeben, die für diese Teilaufgaben betrachtet werden soll.*

Geben Sie die Folge von Sperranforderungen und Freigaben an. Falls eine Transaktion wartet, kennzeichnen Sie dieses. Wenn eine Transaktion am Ende der Historie in einem Deadlock ist, nehmen Sie an, dass alle beteiligten Transaktionenen am Ende abgebrochen werden. In diesem Falle, geben Sie die bereits gewährten Sperren frei.

*Zusätzlich zur Notation, wie auf der vorherigen Seite gegeben (***$\mathbf{X}_i(O)$, $\mathbf{S}_i(O)$, $\mathbf{relX}_i(O)$, $\mathbf{relS}_i(O)$***), nutzen Sie* $\mathbf{gS}_i(O)$ *und* $\mathbf{gX}_i(O)$ *um zu notieren dass* $T_i$ *eine angeforderte Shared- bzw. Exclusive-Sperre erhalten hat. Nutzen Sie* $\mathbf{wait}_i$*, um anzuzeigen, dass Transaktion* $T_i$ *angehalten wurde und* $cont_i$*, um anzuzeigen, dass die Transaktion* $T_i$ *fortsetzt. Sie dürfen fortsetzen weglassen, wir empfehlen aber die Verwendung für bessere Lesbarkeit. Nutzen Sie* $\mathbf{abort}_i$*, um anzugeben, dass* $T_i$ *abgebrochen wurde.* (5 Punkte)

| $T_1$ | $T_2$ | $T_3$ | $T_4$ |
|---|---|---|---|
| | | | $b_4$ |
| | | $b_3$ | |
| $b_1$ | | | |
| $r_1(A)$ | | | |
| | | $r_3(A)$ | |
| $w_1(B)$ | | | |
| | $b_2$ | | |
| | $r_2(A)$ | | |
| $w_1(A)$ | | | |
| | | | $r_4(C)$ |
| | $w_2(C)$ | | |
| | | | $a_4$ |
| | $w_2(D)$ | | |
| | $c_2$ | | |
| | | $r_3(B)$ | |

$\mathtt{S}_1(A)$,$\mathtt{gS}_1(A)$,$\mathtt{S}_3(A)$,$\mathtt{gS}_3(A)$,$\mathtt{X}_1(B)$,$\mathtt{gS}_1(B)$,$\mathtt{S}_2(A)$,$\mathtt{gS}_2(A)$,

$\mathtt{X}_1(A)$,$\mathtt{wait}_1$,$\mathtt{S}_4(C)$,$\mathtt{gS}_4(C)$,$\mathtt{X}_2(C)$,$\mathtt{wait}_2$,$\mathtt{relS}_4(C)$,$\mathtt{abort}_4$,

$\mathtt{cont}_2$,$\mathtt{gX}_2(C)$,$\mathtt{X}_2(D)$,$\mathtt{gX}_2(D)$,$\mathtt{relX}_2(D)$,$\mathtt{relX}_2(C)$,$\mathtt{relS}_2(A)$,

$\mathtt{S}_3(B)$,$\mathtt{wait}_3$,$\mathtt{relS}_3(B)$,$\mathtt{relS}_3(A)$,$\mathtt{abort}_3$, .....

$\mathtt{X}_1(A)$,$\mathtt{relX}_1(A)$,$\mathtt{relX}_1(B)$,$\mathtt{abort}_1$ .....

.....

.....

.....

.....

.....

.....

**MatrikelNr:** _____ **Nachname:** _____

**Für die Aufgaben 4 – 6 gilt die Datenbankbeschreibung auf diesem Blatt.**

**Aufgabe 4:** Erstellen eines Datenbankschemas mittels SQL und Aufgaben zu Schlüsselabhängigkeiten (12)

a) Folgendes Schema sei gegeben:

| | |
|---|---|
| `players` | (name, pid, country, mostGoals: *contracts.id* ) |
| `contracts` | (id, playerName: *players.name*, playerId: *players.pid*, salary, club) |
| `games` | (id: *contracts.id*, gid, coach: *coaches.name*, goals, time) |
| `coaches` | (name, totalGoals, country) |

Ein Universitätsassistent muss für eine Aufgabe in einer Datenbanksysteme-Prüfung ein relationales Schema finden, und entscheidet sich mangels neuer Ideen für ein generisches Schema aus Spieler:innen, Verträgen, Spielen, und Trainer:innen.

Eine Spieler:in (`players`) ist eindeutig gekennzeichnet durch Name (`name`) und Spieler:in-ID (`pid`). Zusätzlich wird das Heimatland (`country`) gespeichert.

Spieler:innen können Verträge (`contracts`) mit mehreren Clubs abschließen. Jeder Vertrag ist eindeutig durch eine ID gekennzeichnet. Die ID muss eine Sequence sein, beginnend mit 5, und in 10er Schritten fortlaufend. Es wird ebenfalls gespeichert welche Spieler:in den Vertrag (`playerName, playerId`) abschließt. Daneben wird das vereinbarte Gehalt (`salary`) festgehalten, sowie der Name des Clubs (`club`). `salary` soll als eine Dezimalzahl mit maximal 2 Kommastellen implementiert werden.

Ferner wird bei jeder Spieler:in auch der Vertrag festgehalten, in welchem die meisten Tore erzielt wurden (`mostGoals`). *Für die Lösung dieser Aufgabe muss nicht geprüft werden ob `mostGoals` tatsächlich auf den Vertrag mit den meisten Toren verweist.*

Eine Spieler:in wird unter jedem Vertrag in Spielen (`games`) eingesetzt. Jedes Spiel ist eindeutig gekennzeichnet durch die Kombination der ID des zugehörigen Vertrags und einer Spiel-ID (`gid`). Es wird für jedes Spiel festgehalten wieviele Tore die Vertragsspieler:in erzielt hat (`goals`), wieviele Minuten diese im Einsatz war (`time`) und unter welcher Trainer:in (`coach`) gespielt wurde. Die erzielten Tore müssen mindestens 0 sein.

Jede Trainer:in (`name`) hat einen eindeutigen Namen. Daneben wird die gesamte Anzahl der erzielten Tore aller zugeordneten Spieler:innen (`totalGoals`) und das Land (`country`) in der Datenbank festgehalten.
Die Anzahl der Tore muss bei jedem Eintrag in der Tabelle zwischen 0 und 5000 liegen. *Für die Lösung dieser Aufgabe muss nicht geprüft werden ob `totalGoals` tatsächlich die gesamte Anzahl aller assoziierten Tore festhält.*

Geben Sie die nötigen SQL Statements an, um obiges Schema (mit allen Konsistenzbedingungen) anzulegen. Wählen Sie passende Typen für Attribute. **Die Abkürzung VC statt VARCHAR(100) ist erlaubt.**

(8 Punkte)

```
CREATE SEQUENCE seq_id INCREMENT BY 10 MINVALUE 5 NO CYCLE;

CREATE TABLE players(
   name          VARCHAR(100),
   pid           INTEGER,
   country       VARCHAR(100),
   mostGoals     INTEGER,
   PRIMARY KEY (name,pid)
);

CREATE TABLE contracts(
   id            INTEGER DEFAULT nextval('seq_id') PRIMARY KEY,
   playerName    VARCHAR(100),
   playerId      INTEGER,
   salary        NUMERIC(8,2) NOT NULL,
   club          VARCHAR(100) NOT NULL,
   FOREIGN KEY (playerName,playerId) REFERENCES players(name,pid)
);

ALTER TABLE players ADD CONSTRAINT c_mostGoals
   FOREIGN KEY (mostGoals) REFERENCES contracts(id)
   DEFERRABLE INITIALLY DEFERRED;


CREATE TABLE coaches(
   name          VARCHAR(100) PRIMARY KEY,
   totalGoals    INTEGER NOT NULL CHECK(totalGoals BETWEEN 0 AND 5000),
   country       VARCHAR(100) NOT NULL
);

CREATE TABLE games(
   id            INTEGER REFERENCES contracts(id),
   gid           INTEGER,
   coach         VARCHAR(100) REFERENCES coaches(name),
   goals         INTEGER NOT NULL CHECK(goals >= 0),
   time          INTEGER NOT NULL,
   PRIMARY KEY (id,gid)
);
```

*Hinweis: Achten Sie bei den Statements auf die Reihenfolge.*

b) Für diese Aufgabe müssen Sie sich mit verschiedenen Schlüsselabhängigkeiten auseinandersetzen.

(4 Punkte)

i) Gegeben ist folgendes Schema:

`game` (id: *level.game*, hardestLevel: *level.name*)
`level` (game: *game.id*, name)

Es ist gefragt, dass Sie folgende Tabellen so erweitern, **dass die resultierende Datenbank-Instanz das Schema erfüllt**. Hierbei stellen die 2 Tabellen den vollständigen Inhalt der Datenbank dar. Um Punkte für diesen Teil zu erhalten, müssen Sie die Tabellen vollständig ausfüllen.

**level**

| **game** | **name** |
|---|---|
| 220 | Blocks |
| 330 | `Stone Age` |
| `100` | `Space` |
| 450 | Tunnels |
| `510` | Mushrooms |

**game**

| **id** | **hardestLevel** |
|---|---|
| 510 | `Mushrooms` |
| `220` | `Blocks` |
| 100 | Space |
| `330` | Stone Age |
| 450 | `Tunnels` |

**Achtung:** Die Lösung ist nicht einzigartig, es wird nur eine beliebige korrekte Instanz verlangt. **Achtung:** Die Lösung ist nicht einzigartig, es wird nur eine mögliche Lösung gezeigt.

ii) Gegeben ist folgendes Schema:

`Q` (id, W: *W.id*, E: *E.id*)
`W` (id, Q: *Q.id*)
`E` (id, W: *W.id*)

Es ist gefragt, dass Sie folgende Tabellen so erweitern, **dass die resultierende Datenbank-Instanz das Schema erfüllt**. Hierbei stellen die 3 Tabellen den vollständigen Inhalt der Datenbank dar. Um Punkte für diesen Teil zu erhalten, müssen Sie die Tabellen vollständig ausfüllen.

**Q**

| **id** | **W** | **E** |
|---|---|---|
| `1` | 2 | 1 |
| 2 | 3 | 3 |
| 3 | `1` | `2` |

**W**

| **id** | **Q** |
|---|---|
| `1` | 3 |
| `2` | `2` |
| `3` | 1 |

**E**

| **id** | **W** |
|---|---|
| `1` | 1 |
| 2 | `2` |
| `3` | `3` |

**Achtung:** Die Lösung ist nicht einzigartig, es wird nur eine beliebige korrekte Instanz verlangt. **Achtung:** Die Lösung ist nicht einzigartig, es wird nur eine mögliche Lösung gezeigt.

**MatrikelNr:** _____ **Nachname:** _____

**Aufgabe 5:** Rekursive Abfragen (12)

Gegeben ist die folgende Rekursive Abfrage auf dem Datenbank-Schema von Aufgabe 4 a):

```
WITH RECURSIVE tmp(playerName, country) AS
(
 SELECT  p.name, p.country
 FROM    players p
 WHERE   p.name = 'Valentina Sommer' AND p.country = 'United Kingdom'
UNION
 SELECT  p.name, p.country
 FROM    tmp t, contracts c1, contracts c2, players p, games g1, games g2
 WHERE   t.playerName = c1.playerName
         AND c1.id = g1.id
         AND c2.id = g2.id
         AND g1.time = g2.time
         AND c2.playerName = p.name
)
SELECT  t.playerName, t.country
FROM    tmp t
```

**Werten Sie diese Abfrage auf der Datenbank-Instanz, die auf der Seite 13 (letztes Blatt der Prüfung) angegeben ist, aus. Wir empfehlen das letzte Blatt abzutrennen.**

| playerName | country |
|---|---|
| Valentina Sommer | United Kingdom |
| Juliane Dietrich | Austria |
| Frank Baumann | United States |
| Jakob Mueller | Netherlands |

**MatrikelNr:** _____ **Nachname:** _____

**Aufgabe 6:** PL/SQL Trigger (12)

Betrachten Sie die Beispielinstanz auf **Seite 13** (letztes Blatt der Prüfung). Wir empfehlen das Blatt abzutrennen.

In jeder der folgenden Teilaufgaben ist ein SQL Statement gegeben, das **über die Beispielinstanz** ausgeführt wird. Nehmen Sie jeweils nur den Bestand entsprechend des letzten Blattes an, d.h. Inserts in Aufgabe a haben keinen Einfluss auf Aufgabe b usw. Auch die Funktionen und Trigger beziehen sich nur auf die jeweiligen Teilaufgaben.

**Geben Sie die Ausgabe der `SELECT`-Statements an. Falls ein Fehler auftreten würde, geben Sie an welcher Fehler auftritt.**

a) (4 Punkte)

```
CREATE OR REPLACE FUNCTION fInsertGamesTwo() RETURNS TRIGGER AS $$
BEGIN

        UPDATE coaches SET totalGoals = totalGoals + NEW.goals
        WHERE NEW.coach = name;

        RETURN NEW;

END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER tInsertGamesTwo AFTER INSERT ON games
       FOR EACH ROW EXECUTE PROCEDURE fInsertGamesTwo();
```

---

```
INSERT INTO games(id, gid, coach, goals, time)
VALUES (65, 4, 'Smith', 5, 30),
       (15, 4, 'Wilson', 1, 93),
       (15, 5, 'Wilson', 1, 95),
       (25, 5, 'Miles', 2, 80);

SELECT * FROM coaches
  WHERE country = 'United Kingdom' AND totalGoals > 0;
```

```
 name | totalgoals | country
--------+------------+----------------
Tilden | 2 | United Kingdom
Smith | 6 | United Kingdom
Wilson | 5 | United Kingdom
Miles | 2 | United Kingdom
```

b) (4 Punkte)

```
CREATE OR REPLACE FUNCTION fInsertGames() RETURNS TRIGGER AS $$
DECLARE
        old_g INTEGER;
        curr_pid INTEGER;
        new_g INTEGER;
BEGIN
        SELECT SUM(g.goals), p.pid
               INTO old_g, curr_pid
        FROM games g, players p, contracts c
        WHERE NEW.id = c.id AND c.playerId = p.pid AND
              p.mostGoals = g.id
        GROUP BY p.pid;

        SELECT SUM(goals) INTO new_g
        FROM games WHERE NEW.id = id;

        IF new_g > old_g THEN
                UPDATE players SET mostGoals = NEW.id
                WHERE curr_pid = pid;
        END IF;
        RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER tInsertGames AFTER INSERT ON games
       FOR EACH ROW EXECUTE PROCEDURE fInsertGames();
```

---

```
INSERT INTO contracts(id, playerName, playerId, salary, club)
VALUES (70, 'Frank Baumann', 12, 200000.50, 'FC Barca'),
       (75, 'Valentina Sommer', 13, 100000.50, 'Real Madrid'),
       (80, 'Juliane Dietrich', 11, 300000.50, 'FC Barca'),
       (85, 'Thomas Stidl', 14, 400000.50, 'Real Madrid');

INSERT INTO games(id, gid, coach, goals, time)
VALUES (70, 4, 'Martins', 1, 50),
       (75, 4, 'Smith', 2, 70),
       (75, 5, 'Smith', 2, 70),
       (80, 4, 'Wilson', 4, 70),
       (85, 5, 'Taylor', 1, 70);

SELECT name, pid, mostGoals FROM players WHERE pid < 15;
```

```
name | pid | mostgoals
-----------------+-----+-----------
Thomas Stidl | 14 | 45
Frank Baumann | 12 | 5
Valentina Sommer | 13 | 75
Juliane Dietrich | 11 | 80
```

c) (4 Punkte)

```
CREATE OR REPLACE FUNCTION fExpensiveContracts() RETURNS TRIGGER AS
$$
BEGIN

    IF NEW.club = 'Real Madrid' AND NEW.salary > 400000 THEN
        INSERT INTO contracts
        VALUES (NEW.id, NEW.playerName, NEW.playerId,
                NEW.salary / 2, NEW.club);
        RETURN NULL;
    ELSE
        RETURN NEW;
    END IF;

END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER tExpensiveContracts
    BEFORE INSERT
    ON contracts
    FOR EACH ROW
EXECUTE PROCEDURE fExpensiveContracts();
```

---

```
INSERT INTO contracts(id, playerName, playerId, salary, club) VALUES
  (70, 'Frank Baumann', 12, 500000.00, 'FC Barca'),
  (75, 'Valentina Sommer', 13, 840000.00, 'Real Madrid'),
  (80, 'Juliane Dietrich', 11, 800000.00, 'FC Barca'),
  (85, 'Thomas Stidl', 14, 600000.00, 'Real Madrid');


SELECT id, salary, playerid FROM contracts
  WHERE club = 'FC Barca' OR club = 'Real Madrid';
```

```
 id | salary | playerid
----+-----------+----------
70 | 500000.00 | 12
75 | 210000.00 | 13
80 | 800000.00 | 11
85 | 300000.00 | 14
```

---

**Viel Erfolg**

Gesamtpunkte: 70

**MatrikelNr:** _____ **Nachname:** _____

**Beispielinstanz für Aufgabe 5 und Aufgabe 6:**

**Sie können diesen Zettel abtrennen und brauchen ihn nicht abgeben!**

**Diesen Zettel daher bitte nicht beschriften! (Lösungen auf diesem Zettel werden nicht gewertet!)**

**players**

| name | pid | country | mostGoals |
|---|---|---|---|
| Juliane Dietrich | 11 | Austria | 25 |
| Valentina Sommer | 13 | United Kingdom | 15 |
| Thomas Stidl | 14 | Germany | 45 |
| Frank Baumann | 12 | United States | 5 |
| Alice Wallner | 15 | Italy | 35 |
| Elisabeth Stoll | 16 | Belgium | 65 |
| Jakob Mueller | 17 | Netherlands | 55 |

**games**

| id | gid | coach | goals | time |
|---|---|---|---|---|
| 5 | 1 | Smith | 1 | 90 |
| 5 | 2 | Smith | 0 | 40 |
| 15 | 1 | Wilson | 2 | 55 |
| 15 | 2 | Wilson | 1 | 90 |
| 15 | 3 | Taylor | 0 | 70 |
| 25 | 1 | Miles | 0 | 55 |
| 35 | 1 | Hemming | 1 | 45 |
| 45 | 1 | Tilden | 2 | 20 |
| 55 | 1 | Martins | 0 | 10 |
| 55 | 2 | Martins | 1 | 70 |
| 65 | 1 | Smith | 0 | 85 |

**contracts**

| id | playerName | playerId | salary | club |
|---|---|---|---|---|
| 5 | Frank Baumann | 12 | 500000.5 | FC Lions |
| 15 | Valentina Sommer | 13 | 450000.75 | SK Castle |
| 25 | Juliane Dietrich | 11 | 300000.9 | FC Bulls |
| 35 | Alice Wallner | 15 | 250000.5 | FC Pool |
| 45 | Thomas Stidl | 14 | 450000.65 | SK Tempo |
| 55 | Jakob Mueller | 17 | 400000.56 | SK Burg |
| 65 | Elisabeth Stoll | 16 | 200000.55 | FC Sun |

**coaches**

| name | totalGoals | country |
|---|---|---|
| Smith | 1 | United Kingdom |
| Wilson | 3 | United Kingdom |
| Martins | 1 | Austria |
| Taylor | 0 | United Kingdom |
| Miles | 0 | United Kingdom |
| Hemming | 1 | Germany |
| Tilden | 2 | United Kingdom |