Gruppe **A**

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

PRÜFUNG AUS		13.03.2018
◯ DATENMODELLIERUNG 2 (**184.790**)	◯ DATENBANKSYSTEME (**184.686**)	**GRUPPE A**
Matrikelnr.	Familienname	Vorname

Arbeitszeit: 60 Minuten. Lösen Sie die Aufgaben auf den vorgesehenen Blättern; Lösungen auf Zusatzblättern werden nicht gewertet. **Viel Erfolg!**

Aufgabe 1: Klassifikation von Historien (8)

Betrachten Sie die unten angegebene Historie, welche durch eine Abfolge von Elementaroperationen der vier Transaktionen T_1, T_2, T_3 und T_4 auf den Datensätzen A, B, C und D gegeben ist. Dabei bedeutet $r_i(X)$ dass Transaktion T_i den Datensatz X liest (*read*), $w_i(X)$ dass Transaktion T_i den Datensatz X schreibt (*write*), und c_i steht für das `commit` der Transaktion T_i.

$$w_2(B), r_3(B), r_1(C), w_1(C), w_2(B), w_4(C), a_4, w_2(A), r_3(C), w_3(C), r_1(D), w_3(D), w_2(D), r_1(D), r_2(B), c_2, c_1, c_3$$

a) Geben Sie an, zwischen welchen Transaktionen eine Leseabhängigkeit besteht. D.h., geben Sie für jede Transaktion an, von welchen anderen Transaktionen diese Transaktion liest (falls eine Transaktion von keiner anderen Transaktion liest, streichen Sie das entsprechende Feld bitte durch).

b) Bestimmen Sie anschließend ob die Historie rücksetzbar ist oder nicht, sowie ob sie kaskadierendes Rücksetzen vermeidet oder nicht. Geben Sie jeweils eine kurze Begründung, z.B. an Hand der obigen Historie, an.
(Achtung, ankreuzen alleine ohne eine Begründung gibt keine Punkte!)

a) Leseabhängigkeiten:

T_1 liest von .....

T_2 liest von .....

T_3 liest von .....

T_4 liest von .....

b) Eigenschaften:

Historie ist Rücksetzbar:

◯ ja ◯ nein

Begründung: .....

.....

Historie vermeidet kaskadierendes Rücksetzen:

◯ ja ◯ nein

Begründung: .....

.....

Aufgabe 2: Logging und Recovery (15)

Gegeben ist (auf der nächsten Seite) eine Historie von drei Transaktionen T_1, T_2 und T_3. In dieser Historie bezeichnen A, B, C und D Felder in der Datenbank, während a_i, b_i, c_i und d_i (für $1 \leq i \leq 3$) lokale Variablen der Transaktionen T_i darstellen.

Nehmen Sie an, dass zu Beginn der relevante Datenbestand der Datenbank aus folgenden Werten besteht:

$$A = 10,\ B = 15,\ C = 25 \text{ und } D = 40.$$

a) Geben Sie an, welche Log-Einträge beim Abarbeiten der gegebenen Historie erstellt werden. Tragen Sie die Log-Einträge in die letzte Spalte der Tabelle ein (beachten Sie, dass womöglich einige Zeilen leer bleiben können).

Nehmen Sie zur Vereinfachung an, dass die Felder A, B, C und D jeweils auf den Seiten P_A, P_B, P_C bzw. P_D liegen, und verwenden Sie das Format

`[LSN, TA, PageID, Redo, Undo, PrevLSN]` für "normale" Logeinträge, und das Format

⟨`LSN, TA, PageID, Redo, PrevLSN, UndoNextLSN`⟩ für Kompensations Logeinträge (Compensation Log Records). Für `BOT` und `COMMIT` Log-Einträge kann das Format `[LSN, TA, BOT, PrevLSN]` bzw. `[LSN, TA, COMMIT, PrevLSN]` verwendet werden.

Beachten Sie, dass Undo/Redo-Einträge *relativ* zum Datenbestand mittels *Addition* bzw. *Subtraktion* anzugeben sind, z.B.: [#i, T_j, P_X, X+=d_1, X-=d_2, #k] bedeutet, dass laut i-tem Logeintrag die Transaktion T_j auf ein Datum X auf der Seite P_X schreibend zugreift, so dass beim Redo X um d_1 vergrößert werden müsste und beim Undo X um d_2 verkleinert werden müsste. Außerdem hat der vorangegangene Logeintrag dieser Transaktion die Nummer k.

b) Nehmen Sie an, dass als nächste Operation ein `abort` für die Transaktion T_2 ausgeführt wird. Führen Sie ein lokale Undo aus um die Transaktion zurückzusetzen. Protokollieren Sie die dabei entstehenden Log-Einträge in der unten angegebenen Liste. Schreiben Sie bitte nur einen Logeintrag pro Linie – möglicherweise sind nicht alle angegebenen Zeilen erforderlich.

.....	.....
.....	.....
.....	.....
.....	.....

c) Geben Sie die Werte für A, B, C und D nach dem erfolgreichen Zurücksetzen an.

A: ..... B: ..... C: ..... D: .....

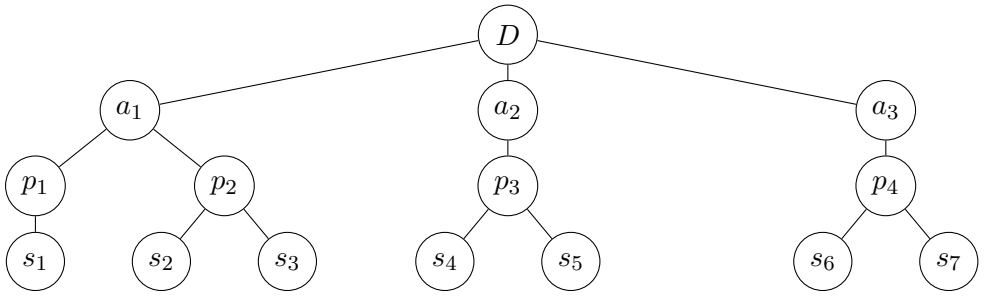
d) Nehmen Sie an, zwischen Operation 19 und 20 wäre ein aktionskonsistenter Sicherungspunkt angelegt worden. Bei welcher Operation würde nach einem späteren Absturz ein globales Redo beginnen, und bis zu welcher Operation müsste ein anschließendes globales Undo unter Umständen zurückgehen?

Redo: ..... *Undo*: .....

Schritt	$\mathbf{T_1}$	$\mathbf{T_2}$	$\mathbf{T_3}$	**Logeinträge**
1			`BOT`	.....
2			$r(A, a_3)$	.....
3			$w(A, a_3 + 30)$	.....
4	`BOT`			.....
5	$r(B, b_1)$			.....
6	$r(C, c_1)$			.....
7		`BOT`		.....
8		$r(D, d_2)$		.....
9		$w(D, d_2 + 25)$		.....
10	$w(C, c_1 - b_1)$			.....
11			$r(C, c_3)$	.....
12	$r(C, c_1)$			.....
13			`COMMIT`	.....
14	$r(D, d_1)$			.....
15		$r(B, b_2)$		.....
16	$w(B, c_1 + d_1)$			.....
17		$w(B, b_2 + d_2)$		.....
18	$w(D, d_1 + 1)$			.....
19	`COMMIT`			.....
20		$w(C, d_2 - 15)$		.....

Aufgabe 3: Mehrbenutzersynchronisation/Locking (12)

Gegeben ist folgende Datenbasis-Hierarchie.



Im folgenden bezeichnet S eine Lesesperre, X eine Schreibsperre, IS eine beabsichtigte Lesesperre und IX eine beabsichtigte Schreibsperre.

a) Betrachten Sie die folgende Sequenzen von Sperranforderungen bzw. Freigaben von Sperren der Transaktionen T_1, T_2 und T_3. Dabei bedeutet $\ell_i(o, A)$ dass die Transaktionen T_i eine Sperre vom Typ A auf das Objekt o anfordert, und $r_i(o, A)$ dass die Transaktion T_i eine Sperre vom Typ A auf das Objekt o freigibt.

$$\ell_1(D, IX), \ell_2(D, IX), \ell_1(a_1, IS), \ell_2(a_2, IX), \ell_3(D, IS), \ell_1(p_1, S), \ell_1(p_2, IS)$$

$$\ell_1(a_3, IX), \ell_2(s_4, X), \ell_3(a_2, IX), \ell_1(s_3, S), r_1(p_2, IS)$$

Geben Sie an, welche Transaktionen das Sperrprotokoll des Multiple-Granularity Locking (MGL) verletzen, und warum diese Transaktionen das Sperrprotokoll nicht einhalten.

Die folgenden Transaktionen verletzen das Sperrprotokoll des MGL: .....

Begründung: .....

.....

.....

b) Nehmen Sie an, die Transaktionen T_1, T_2, T_3 und T_4 möchten folgenden Sperren in der angegeben Reihenfolge erhalten

$$S_1(p_4), X_3(p_2), X_2(s_4), S_4(s_6), X_3(p_4), X_2(s_5), S_1(s_1), S_1(s_2)$$

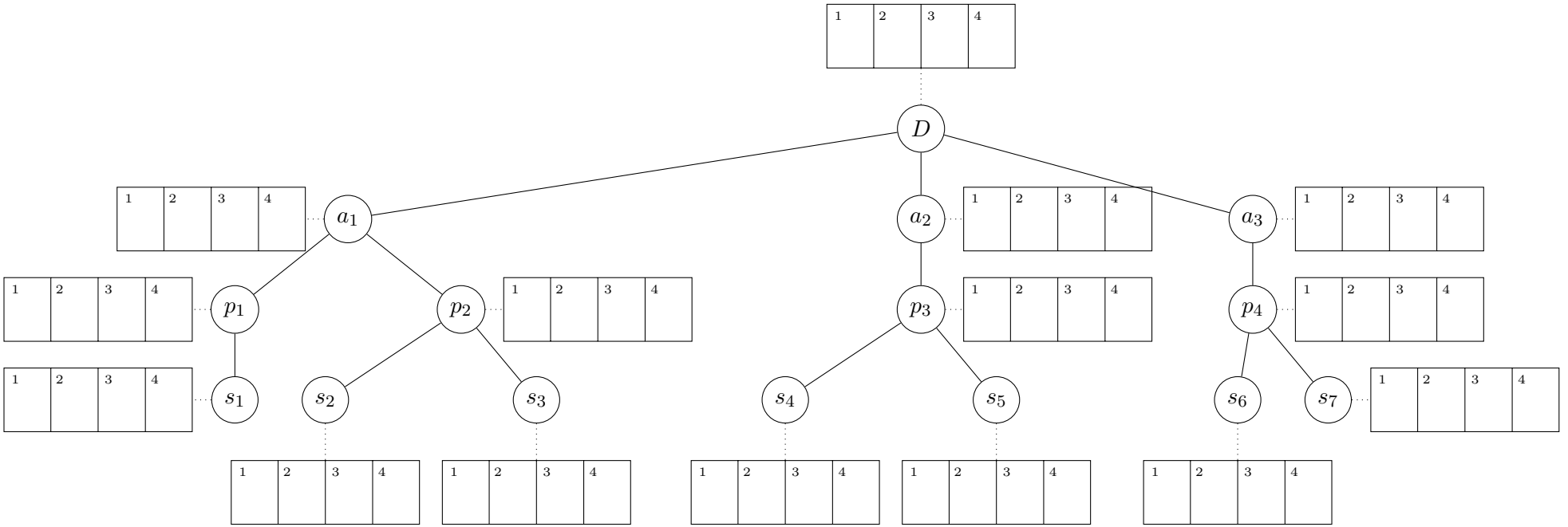
Dabei bezeichnet $S_i(o)$ den Wunsch von Transaktion T_i eine Lesesperre auf das Objekt o zu erhalten, und $X_i(o)$ den Wunsch eine Schreibsperre zu erhalten. Nehmen Sie weiter an, dass die Transaktionen zum Erhalt dieser Sperren nach dem Sperrprotokoll des MGL vorgehen (und dieses auch korrekt einhalten; beachten Sie, dass obige Angabe nur die Sperren beinhaltet, welche die Transaktionen haben wollen, und dass zum Erhalt dieser zusätzliche Sperren nötig sind).

Beschreiben Sie den Zustand am Ende dieser Sperranforderungen, indem Sie in der Grafik auf der nächsten Seite zu jedem Knoten notieren welche Transaktionen welche Sperren auf diesem Knoten halten. Tragen Sie dazu S, X, IS und IX in das Feld mit der Transaktionsnummer ein um auszudrücken, dass die Transaktion die entsprechende Sperre hält. Sollte eine Transaktion eine Sperre angefordert aber nicht erhalten haben, tragen Sie bitte WS, WX, WIS oder WIX ein. Sollte eine Transaktion blockieren, ignorieren Sie alle weiteren Anforderungen dieser Transaktion.

Um nicht umblättern zu müssen noch einmal die geforderten Sperren:

$$S_1(p_4), X_3(p_2), X_2(s_4), S_4(s_6), X_3(p_4), X_2(s_5), S_1(s_1), S_1(s_2)$$

b) MGL Sperren:



Für die Aufgaben 4 – 6 gilt die Datenbankbeschreibung am letzten Blatt dieser Prüfung.

Aufgabe 4: CREATE Statements (13)

Achtung: *Bitte lesen Sie zuerst aufmerksam die Datenbankbeschreibung am letzten Blatt dieser Prüfung.*

Bevor Sie die Tabellen anlegen, erstellen Sie eine Sequenz `seq_pids`. Diese Sequenz soll bei 100 beginnt und in 10er Schritten erhöht wird. Die Sequenz soll beim Überlaufen nicht wiederholt werden.

Geben Sie nun CREATE-Statements mit allen entsprechenden Constraints für die Tabellen `song`, `playlist` und `part_of` an. Wählen Sie entsprechende Datentypen (INTEGER, VARCHAR) für die Attribute. Achten Sie darauf, dass `NULL`-Werte nur dort erlaubt sind, wo sie unbedingt benötigt werden.

Aufgabe 5: Rekursive Abfragen (10)

Evaluieren Sie das folgendes SQL-Statement bezüglich der Datenbankinstanz auf letzten Seite, und geben Sie die Ausgabe der Abfrage an:

```
WITH RECURSIVE t(id,d) AS (
      SELECT p.pid, s.duration
      FROM part_of p NATURAL JOIN song s
   UNION ALL
      SELECT p.parent, t.d
      FROM t JOIN playlist p ON t.id = p.pid
      WHERE p.parent IS NOT NULL
) SELECT id, SUM(d) FROM t GROUP BY id ORDER BY id;
```

Aufgabe 6: PL/SQL Trigger (12)

Erstellen Sie einen PL/pgSQL Trigger `trCheckGenre`, der vor dem Einfügen in die `part_of`-Tabelle für jede einzufügende Zeile die Funktion `fCheckGenre` aufruft.

Diese Funktion soll sicherstellen, dass nur Songs des gleichen Genres in einer Playlist vorhanden sein dürfen. Dies bedeutet:

- Falls noch kein Song in der Playlist vorhanden ist, wird der Song auf jeden Fall zur Playlist hinzugefügt.
- Falls bereits Songs in der Playlist vorhanden sind, muss der Song vom selben Genre wie die Songs in der Playlist sein. Sonst wird dieser nicht hinzugefügt und eine Exception mit dem Text "Song has different genre!" ausgeworfen.

Sie können diese Seite abtrennen und brauchen ihn nicht abgeben!

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

Die folgende Datenbankbeschreibung gilt für die Aufgaben 4 – 6:

Gegeben ist folgendes stark vereinfachtes Datenbankschema zum Speichern von Songs und Playlists.

song(sid, name, artist, duration, genre)
playlist(pid, name, parent: *playlist.pid*)
part_of(sid: *song.sid*, pid: *playlist.pid*)

Jeder Song `song` hat eine eindeutige Identifikationsnummer `sid` und einen Namen `name`. Weiters, wir der Künstler `artist` und die Dauer `duration` des Songs gespeichert. Das Attribut `genre` kann nur die Werte `pop`, `rock` und `classic` annehmen.

Jede Playlist `playlist` hat eine eindeutige Identifikationsnummer `pid`. Diese Identifikationsnummer soll mit Hilfe der Sequenz `seq_pids` vergeben werden. Zusäetzlich hat jede Playlist einen Namen `name` und kann außerdem einer übergeordneten Playlist `parent` angehören.

Die Tabelle `part_of` speichert die Zuordnung von Songs zu Playlists.

Beispielinstanz für Aufgabe 4 – 6:

song				
sid	name	artist	duration	genre
1	Underneath Your Clothes	Shakira	350	pop
2	Single Ladies	Beyonce	250	pop
3	Wannabe	Spice Girls	300	pop
4	Summer of 69	Bryan Adams	200	rock
5	Runaway	Bon Jovi	250	rock
6	In These Arms	Bon Jovi	250	rock
7	Fuer Elise	Beethoven	200	classic
8	Radetzky Marsch	Johann Strauss (Vater)	200	classic
9	Donauwalzer	Johann Strauss (Sohn)	150	classic

part_of	
sid	pid
1	110
2	120
3	120
4	110
4	130
5	130
7	100
8	140
9	140

playlist		
pid	name	parent
100	Complete	
110	Best Of Modern	100
120	Best Of Pop	110
130	Best Of Rock	110
140	Best Of Classic	100