Allgemeine Informationen

What's this

Dies ist eine studentische, prüfungsschwierigkeit immitierende Aufgabenzusammenfassung die Themen der Vorlesung IDB. Es handelt sich insbesondere nicht um einen sogenannten "Braindump". Die Aufgaben orientieren sich an den Aufgaben in der ursprünglichen Klausur insofern, dass hier zum Beipsiel anstatt der Frage 'Sind Kühe Lila?' die Frage 'Machen Kuehe Schokolade?' steht. Auf beide Fragen ist die Antwort: Nein, du schaust zuviel Milkawerbung.

Maintaining

Der Latex-Source dieses PDFs wird auf *https://gitlab.cs.fau.de/ik15ydit/latexandmore* (ein Account ist noetig) maintain't. Da die Aufgaben alle neu formuliert und konzipiert wurden, kann es sein, dass uns an einigen Stellen Fehler Unterlaufen sind. Solltet ihr solche Fehler finden oder generell Anmerkungen haben koennt ihr mit einem Account auf *gitlab.cs.fau.de* eine Issue aufmachen oder einen Pullrequest stellen.

Aufgabe 1 - Kunterbunte Schic *hicks* ten

Begründe warum folgende Aussagen korrekt oder inkorret sind. Es gibt nur einen Punkt wenn Antwort UND Begruendung richtig sind.

- **a)** Ein Sektor auf einem Speicherzylinder ist genau groß genug fuer einen Satz.

- **b)** Block- und Seitenabellen sind in den Speicherzuordnungstrukturen angesiedelt.

- **c)** Blöcke müssen nur bei direkter Seitenzuordnung gleich groß sein.

- **d)** Die Seitenersetzungsstrategie LFU (Least frequently used) hat den Vorteil, dass sich der Puffer schnell an neue Datenlokalitaeten anpasst.

- **e)** Eine Seite im Datenbankpuffer muss irgendwann auf die Platte zurueckgeschrieben werden..

- **f)** Bei der Verlaengerung eines Satzes kann es vorkommen, dass sich dessen TID aendert.

- **g)** Ein Zugriff ueber einen Index ist immer schneller als ein Table-Scan.

- **h)** Ein Schluessel in einem Index, darf nicht nicht von fester Laenge sein verwendet werden. [Das - nicht nicht - ist beabsichtigt]

- **i)** Die Hashtabelle beim Linearen Hashing, wird, wenn sie vergroessert wird, jedes mal um den einen festen Wert n erhoeht.

- **j)** Wenn in einem B*-Baum Werte gesucht werden, muss immer bis zu einem Knoten ohne Kind abgestiegen werden.

- **k)** Ein B-Baum kann keine variablen Daten enthalten.

- **l)** In einem R-Baum kann es vorkommen, dass wir auf der Suche nach einen Wert mehrere Blattknoten betrachten muessen.

- **m)** Sichten beschleunigen die Aussfuehrung von Anfragen.

- **n)** In den ACID-Eigenschaften einer Transaktionen steht das A fuer Authencity.

- **o)** Bei Forward-Recovery darf kein 'steal' stattfinden.

Aufgabe 2 Ich weiss wo deine TID wohnt

0.1 a)

Deine Mutter hat **Clock** als Seitenersetzungsstrategie verwendet und kapiert jetzt ihren eigenen Code nicht mehr, deswegen musst du die Seitenersetzungsstrategie jetzt unten nachbauen. Du hast 3 Plaetze in deinem Haupspeicher, die haben jeweils einen Kontrollzustand (also auch 3 insgesamt) unten ist dann noch dein Pointer bei welchem Platz du gerade bist. GoGoGo!!!

Input	...	4	7	2	1	5
Hauptspeicher						
Speicherplatz A	...	4				
Speicherplatz B	...	1				
Speicherplatz C	...	9				
Kontrollzustaende						
Platz A	...	1				
Platz B	...	1				
Platz C	...	1				
Cur. Pointer	...	C				

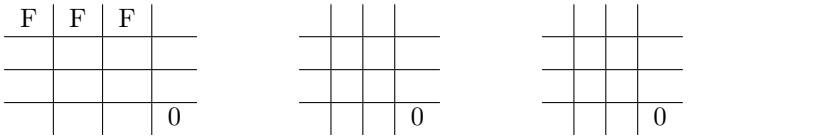
b)

Gebe an welche Schnittstellen ein Datenbankpuffer **nach oben im Schichtenmodell** anbieten muss. (Welche Parameter, welche Rueckgabewerte und wozu sie dient.)

Aufgabe 3 TID-Adressierung

TIDs nach Regeln aus der Vorlesung ergaenzen/eintragen und TID angeben. Teilaufgaben sind unabhaengig. Buchstaben stehen fuer bereits vorhandene Saetze

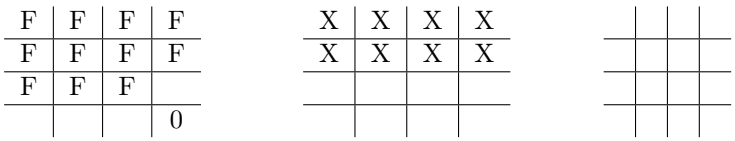
- **a)** Fuegen einen Satz A mit Laenge 12 ein.



\- Seite 0 Seite 1 Seite 2 -

Satz A: TID(,)

- **b)** Ausgangsbelegung mit Saetzen F und X. Fuegen den Satz B, Laenge 14 ein.



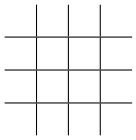
\- Seite 0 Seite 1 Seite 2 -

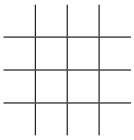
Satz A: TID(,)

- **c)** Die Satze F,C und X sind bereits vorhanden.

  Ausganszustand

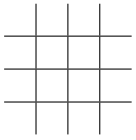
F	F	F	F
F	X	X	X
X	C	C	C
	9	5	0

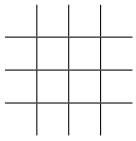


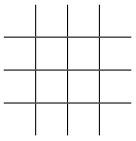


\- Seite 0 Seite 1 Seite 2 -

  Satz C@TID(0,2) auf Laenge 15.







\- Seite 0 Seite 1 Seite 2 -

  Satz F: TID(,), Satz X: TID(,), Satz C: TID(,)

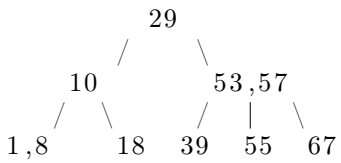
Aufgabe 4 Indrexsstrukturen

a) Erklaeren sie den Unterschied zwischen Primaer und Sekundaerorganisation im Bezug auf Datenspeicherung.

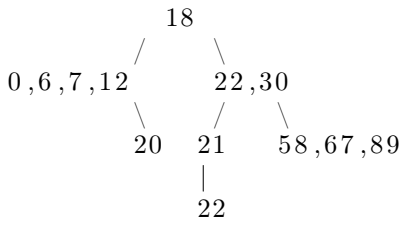
b) Fuege in einen leeren **B*-Baum** mit $k_{inner} = 2$ und $k_{leaf} = 1$ die Tupel **(8,A)** **(17,Z)** **(1,U)** in dieser Reihenfolge ein. Einzelschritte sind nicht noetig.

.

c) Loesche aus dem folgenden B-Baum den Schluessel 10. Zwischenschritte sind nicht noetig.



d) Nennen sie vier Gruende, warum der folgende Baum kein korrekter **B-Baum** ist.



e) Wir haben eine Tabelle in der wir Studenten verwalten wollen, darin gibt es folgende Attribute.

Student(Geschlecht, Jahrgang, Dumm, FSI-Mitglied, Gehirngeschaedigt)

Weil wir jetzt schnell die Korrelanz zwischen einer FSI-Mitgliedschaft und dem geistigen Zustand eines Studenten rausfinden wollen moechten wir einen Index ueber diese Attribute anlegen: **Dumm, FSI-Mitglied, Gehirngeschaedigt**. Welche Indexform eignet sich dafuer besonders gut?

Aufgabe 5 Transaktionen

a) Was ist eine Real-World-Action im Datenbank Kontext

b) Welche Eigenschaften hat eine RWA die als Transaktion modeliert wurde wahrscheinlich gehabt?

c) Ablauf von Transaktionen

rN(x) bedeutet, Transaktion N liest Element x. wN(x) bedeutet, Transaktion N schreiben Element x. Zeichne einen Abhaengigkeitsgraphen, es muss bei jeder Kante ersichtlich sein wie sie Zustande gekommen ist.

- w3(c)
- r3(b)
- w3(a)
- w2(b)
- r1(a)
- r2(b)
- w3(b)
- r1(a)
- w1(b)
- r2(c)

d) Wann ist ein Ablauf serialisierbar?

e) Gegeben ist der folgende Ablauf im Kontext einer fiktiven Lagerverwaltung: (repraesentativ in Java-like Syntax, alle Methoden des fiktiven Datenbank Frameworks scheissen eine FailureException.)

```
class Failure extends Exception;
class Gegenstand{...};
while(true}{
    try{
        startTransaction();
        setLocalCounterToZero();
        while(!terminatedByOperator()){
            Gegenstand g = queryInputFromOperator(); //RWA
            if(isInInventory(g)){
                incrementLocalCounter();
                decrementInventoryCountInDatabase();
            }else{
                displayErrorMessage();
                continue;
            }
        }
        writeNewEntryToDatabase(getUser(),getLocalCounter());
        commitTransaction(); //die Transaktion wird beendet
    }catch(Failure f)
        continue;
    }
}
```

Erklaere welches Problem hier Problem hier auftreten kann und wie es zu loesen waere.

f) Inwiefern begrenzen Action-Consisten-Checkpoints Undo- und Redooperationen?

Aufgabe 6 SQL und Mengen

a) Operatorengraphen zu SQL-Kot unten. (muss nicht optimiert werden)

```
SELECT *
FROM (
        SELECT X.b, X.e, Z.c
        FROM R JOIN S on R.k = S.k
        WHERE S.u = "Mimimi"
        UNION
        SELECT T.b, T.e, T.c
        FROM T
) X
WHERE X.b < 0;
```

b) Warum gilt folgendes nicht allgemein? (ein bis zwei Saetze, Gegenbeispiel reicht nicht)

PROJ(A − B , X) = PROJ(A , X) − PROJ(B , X)

'-' steht hier fuer die Differenz zweier Mengen

Aufgabe 7 Programmschnittstelle

a) Ist das folgende Programm anfaellig fuer SQL-Injections? Begruenden sie.

```
public static void main(String[] args) {
 Connection conn = null;
 Statement stmt = null;
 try{
      Class.forName("com.mysql.jdbc.Driver");
      conn = DriverManager.getConnection(DB_URL, USER, PASS);
      stmt = conn.createStatement();
      Scanner s = new Scanner();
      String tmp = s.readLine();
      String a = Integer(Integer.parseInt(tmp)).toString();
      sql = "INSERT INTO Trololololol VALUES(" + a + ")";
      stmt.executeUpdate(sql);
 }
}
```

b) Wie kann man das obrige Program SICHER gegen SQL-Injections schuetzen?