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. Nein, die größe eines Sektors ist nicht definiert, er ist wahrscheinlich größer.

- **b)** Block- und Seitenabellen sind in den Speicherzuordnungstrukturen angesiedelt.
  Nein, bei den Seitenzuordnungsstrukturen.

- **c)** Blöcke müssen nur bei direkter Seitenzuordnung gleich groß sein.
  Nein, theoretisch könnten mit entsprechenden Verwaltungsoverhead verschieden große Blöcke verwendet werden.

- **d)** Die Seitenersetzungsstrategie LFU (Least frequently used) hat den Vorteil, dass sich der Puffer schnell an neue Datenlokalitaeten anpasst.
  Nein, denn die Puffereffizients kann dadurch beeintraechtigt werden, dass es sehr lange dauert bis eine, zeitweise stark aber dann ie wieder genutzte Seite aus dem Puffer verdraengt wird.

- **e)** Eine Seite im Datenbankpuffer muss irgendwann auf die Platte zurueckgeschrieben werden..
  Nein, mitunter wurde sie nicht veraendert oder es gibt andere Gruen-

de warum man sie nicht auf die Platte schreiben moechte (rollback aus hoeherer Ebene?).

- **f)** Bei der Verlaengerung eines Satzes kann es vorkommen, dass sich dessen TID aendert.
  Nein, denn die TID wird mitunter bereits woanders verwendet und darf

  deshalb nichtmehr geaendert werden.

- **g)** Ein Zugriff ueber einen Index ist immer schneller als ein Table-Scan.
  Nein, bei genau einem Eintrag nicht.

- **h)** Ein Schluessel in einem Index, darf nicht nicht von fester Laenge sein verwendet werden. [Das - nicht nicht - ist beabsichtigt]
  Nein, (also doch) denn es gibt Indexstrukturen die das ermoeglichen.

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

- **j)** Wenn in einem B*-Baum Werte gesucht werden, muss immer bis zu einem Knoten ohne Kind abgestiegen werden.
  Nein, denn nur in den Blattknoten werden Daten gespeichert.

- **k)** Ein B-Baum kann keine variablen Daten enthalten.
  Nein, ein B-Baum kann im Allgemeinen beliebige Daten und Datenmengen enthalten, also auch Saetze.

- **l)** In einem R-Baum kann es vorkommen, dass wir auf der Suche nach einen Wert mehrere Blattknoten betrachten muessen.
  Nein, R-Baume sind Suchbaeume, daher wissen wir immer in welchen Ast wir absteigen muessen, und es kann nicht sein, dass wir uns mehrere Blaetter auf der Suche nach einem Wert anschauen muessen.

- **m)** Sichten beschleunigen die Aussfuehrung von Anfragen.
  Nein, nur materialisierte Sichten die tatsaechlich im Speicher liegen.

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

- **o)** Bei Forward-Recovery darf kein 'steal' stattfinden.
  Ja, steal beudetet, dass Dinge aus dem Puffer vor Ende der Transaktion zurueckgeschrieben werden, damit haette man evt. einen Inkonsitenten Zustand auf der Platte den man nicht rueckgaengig machen kann weil man Forward-Recovery hat.

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	7	5
Hauptspeicher						
Speicherplatz A	...	4	4	2	2	2
Speicherplatz B	...	1	1	1	1	5
Speicherplatz C	...	9	7	7	7	7
Kontrollzustaende						
Platz A	...	1	0	1	1	1
Platz B	...	1	0	0	0	1
Platz C	...	1	1	1	1	1
Cur. Pointer	...	C	A	B	B	C

b)

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

Funktion	**Parameter**	**Rueckgabewert**	**Beschreibung**
fix	Datei, BlockNo, Mode	Pointer auf Seite	laed in Puffer, lockt fuer andere
unfix	Pointer auf Seite im Puffer	void	gibt Block im Puffer fuer Ersetzung frei

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.

F	F	F	
			0

Seite 0

x	x	x	x
x	x	x	x
x	x	x	x
			0

Seite 1

			0

Seite 2

Satz A: TID(2 , 0)

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

F	F	F	F
F	F	F	F
F	F	F	
			0

Seite 0

F	F	F	F
F	F	B	
		5	0

Seite 1

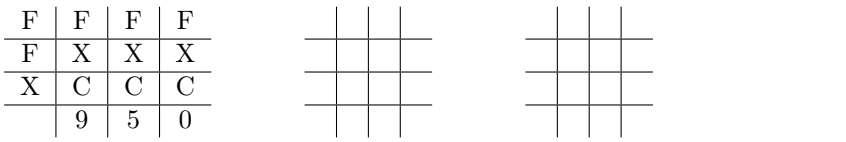
TID	(1,1)	B	B
B	B	B	B
B	B	B	B
B	B	B	0

Seite 2

Satz A: TID(1 , 1)

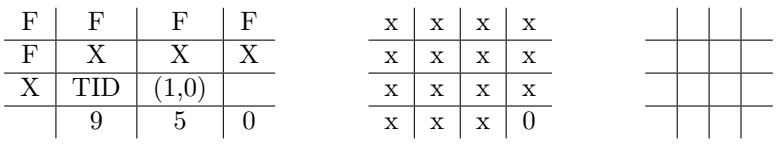
- **c)** Die Satze F,C und X sind bereits vorhanden.

  Ausganszustand



- Seite 0 Seite 1 Seite 2 -

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



- Seite 0 Seite 1 Seite 2 -

  Satz F: TID(0 , 0), Satz X: TID(0 , 1), Satz C: TID(0 , 2)

Aufgabe 4 Indrexsstrukturen

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

Bei der Primaerorganisation werden die Daten tatsaechlich gespeichert, bei der Sekundaerorganisation werden sie nur referenziert, es kann also mehrere Sekundaerorganisationen ohne Datenredundanz geben. **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.

Wenn mir das wer baut fueg ichs ein.

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

- nicht balanciert
- 20 steht auf der falschen Seite, es haette oben bei 18 **rechts** eingetragen werden muessen
- es gibt kein k_{inner} fuer das min/max in Knoten hinhaut

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? Bitmap-Index

Aufgabe 5 Transaktionen

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

Eine in der Realen Welt stattfindende, nicht rueckgaenig machbare Aktion. (?)

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

Sie war wahrscheinlich pruefbar oder wiederholbar.

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?

Wenn kein Zyklus im Abhaengigkeitsgraphen.

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. Während der Transaktion wird auf Eingabedaten von des Operators gewartet und währenddessen sind alle anderen Transaktionen auf die gleiche Datenbanktabelle blockiert. Es kann passieren, dass der Operator extrem lange mit der Eingabe braucht und derweil sind andere DB-Zugriffe blockiert. Die Loesung ist die Transaktion kleiner zu machen und den lokalen counter irgendwie mit in die DB zu commiten sobald man etwas am Bestand aendert.

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

TODO

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 Nein, denn es gehen bei der Projection Informationen verloren, die spaeter fuer die Differenz der Mengen einen Unterschied machen koennten. (Eine Tabelle mit drei Attributen die auf zwei Attribute reduziert/projeziert wurde und in der nur Zeilen gleich scheinen weil das unterscheidende dritte Attribut nicht mehr sichtbar/vorhanden ist.

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 Trolololololol VALUES(" + a + ")";
      stmt.executeUpdate(sql);
 }
}
```

Eher nicht, weil jede versuchte SQL Injection am Integer.parseInt() zerschellt. In der Klausur wars eine, aber ich wollte mal trollen. Man koennte auch argumentieren, dass das hier einen DOS-Angriff ermoeglichen koennte. **b)** Wie kann man das obrige Program SICHER gegen SQL-Injections schuetzen? preparedStmt.setInt(1, ai)

preparedStmt.execute();
Aber alleine die Aussage Prepared-Statements empfinde ich hier als wichtig.