Prof. Dr.-Ing. Dr. h. c. T. Härder
Fachbereich Informatik
AG Datenbanken und Informationssysteme
Universität Kaiserslautern

Übungsblatt 1 – Lösungsvorschläge

Unterlagen zur Vorlesung: „wwwdvs.informatik.uni-kl.de/courses/DBSREAL/“

Aufgabe 1: Der Transaktionsbegriff

Was ist eine Transaktion? Was sind ihre wesentlichen Eigenschaften?

Eine Transaktion ist eine ununterbrechbare Operation, die dauerhaft einen konsistenten Zustand einer Datenbank in einen (nicht notwendigerweise unterschiedlichen) konsistenten Zustand überführt.

Ihre wesentlichen Eigenschaften sind die ACID-Eigenschaften: A(tomicity), I(solation), C(onsistency), D(urability). .

Beurteilen Sie anschließend folgende Aussagen oder Fragen unter ACID-Gesichtspunkten:

1. „Mein Transaktionsprogramm wurde abgebrochen, und nun ist meine Datenbank zerschossen.“

   Widerspricht dem Atomicity-Gesichtspunkt von ACID-Transaktionen.

2. „Leider wurde ihre erfolgreich abgeschlossene Transaktion zurückgesetzt, da das Datenbanksystem abgestürzt ist.“

   Widerspruch zu Durability.

3. „Seit dem Abort meiner Transaktion sind deren Änderungen überhaupt nicht mehr vorhanden!“

   Genau so soll es sein!

4. „Eine andere Transaktion hat Änderungen meiner Transaktion überschrieben. Darf ich jetzt meine Transaktion überhaupt noch beenden, oder muß ich sie abbrechen?“

   Die andere Transaktion kann wegen der Isolation die Änderungen überhaupt nicht sehen. Wegen der Isolation kann ich auf jeden Fall meine Transaktion abbrechen.

5. „Nachdem diese mysteriöse Transaktion *M* gelaufen ist, kann keine weitere Transaktion mehr laufen, da sie – trotz erfolgreichem Abschluß – die Datenbank zerstört hat. “

   Widerspruch zu Consistency.

6. „Die Bank mußte leider feststellen, daß Geld ausgezahlt wurde, obwohl die Überweisungstransaktion abgebrochen wurde.“

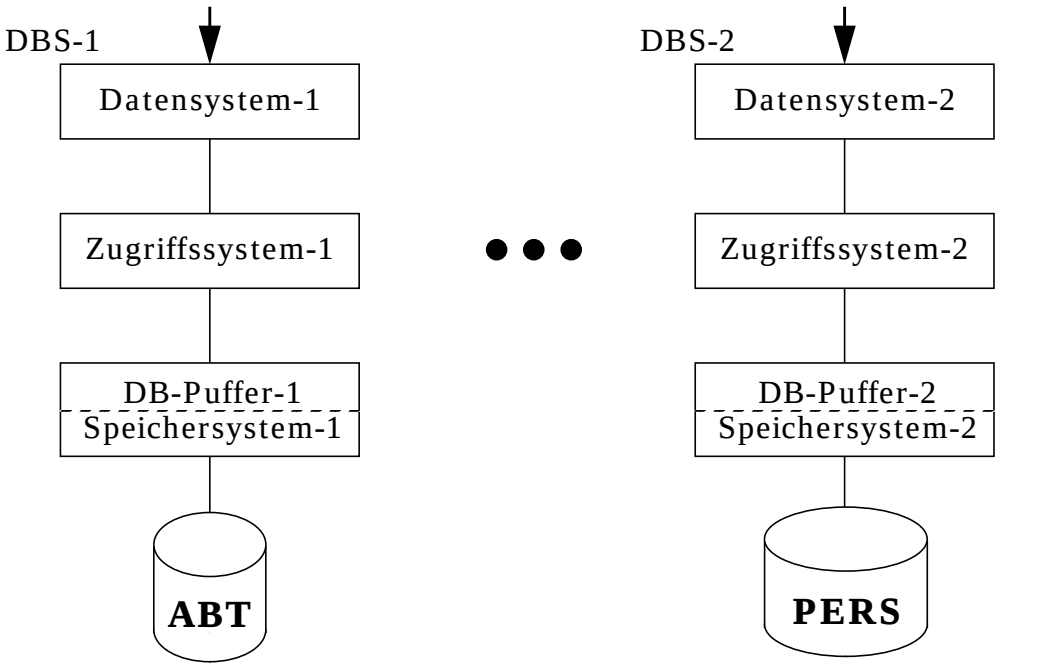
   Die berühmten *real-life actions*. Sind mit ACID nicht zu erfassen, evtl. muß eine Kompensationstransaktion angestoßen werden.

Aufgabe 2: Function Request Shipping 141m

Gehen Sie von der Verteilung einer „PERS – ABT“- Datenbank auf zwei Rechner aus. Welche Kommunikationshäufigkeit und welches Kommunikationsvolumen (Anfragen und Antworten) ergibt sich bei *Function Request Shipping* auf den Ebenen Datensystem, Zugriffssystem und Speichersystem (DB-Puffer), wenn folgende Zuordnung von Anfragen zu Rechnern erfolgt?

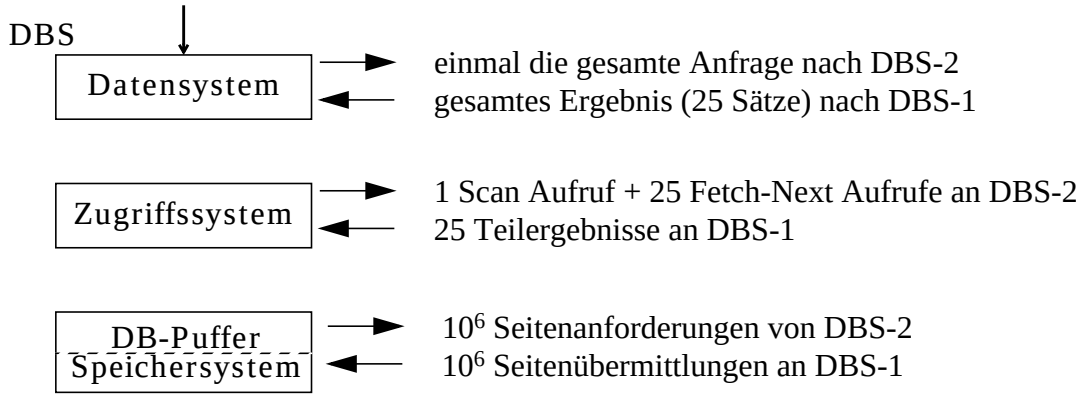
Welche Anforderungen werden auf den einzelnen Ebenen verschickt?

```
SELECT  COUNT (*)                    SELECT   *
FROM    PERS                         FROM    PERS P, ABT A
WHERE   ANR = 'K55' AND              WHERE   P.ANR = A.ANR
        BERUF = 'SYSPROG'
```

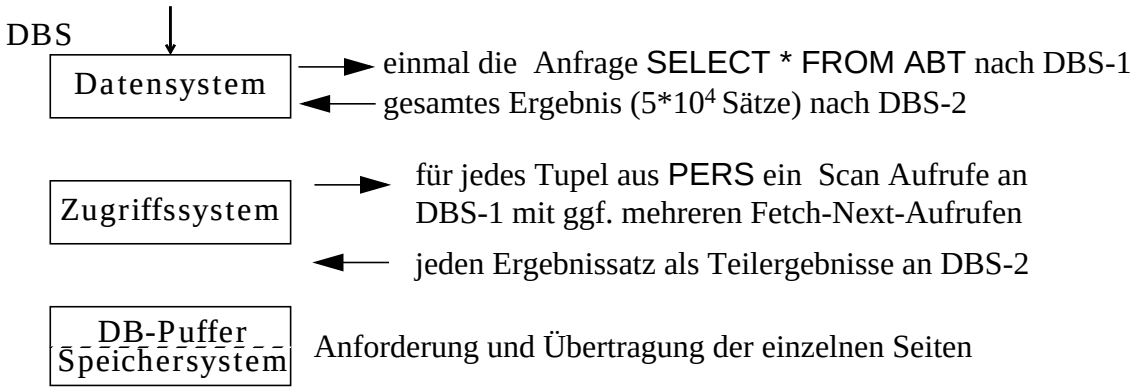


Annahmen: PERS-Datei habe 10^6 Seiten
ABT-Datei habe 10^4 Seiten
PERS-Relation habe 10^7 Tupel
ABT-Relation habe $5*10^4$ Tupel
P.ANR = 'K55' AND P.BERUF = 'SYSPROG': 25 Sätze in PERS

Anfrage1 auf DBS-1:



Anfrage 2 auf DBS-2:



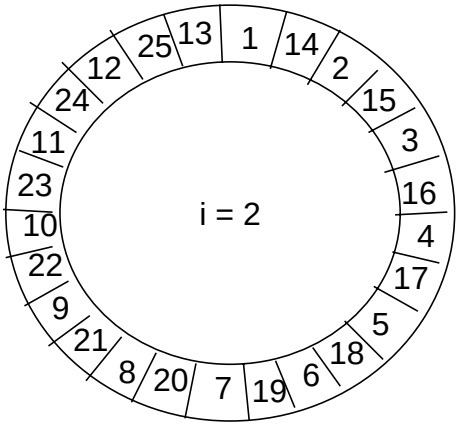
(hier ist es z.B. auch möglich, im Zugriffssystem zunächst alle Sätze von ABT aus DBS-1 zu lesen, d. h. einen Scan über alle Sätze von ABT durchzuführen)

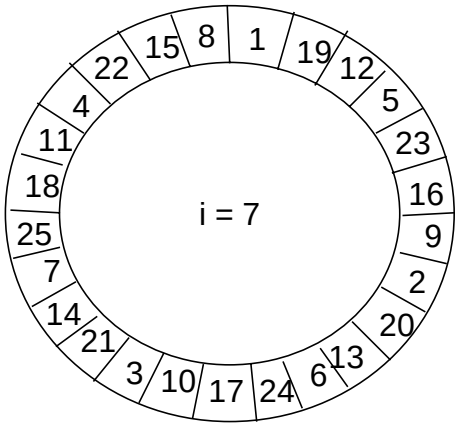
Aufgabe 3: Blockadressierung auf externen Speicher 184

Zur Unterstützung der sequentiellen Verarbeitung werden die Blöcke auf einer Spur nicht sequentiell, sondern versetzt angeordnet, so daß während einer Umdrehung der Platte mehrere Blöcke sequentiell gelesen werden können. Eine Spur auf der Platte bestehe aus m = 25 Slots mit je 4 KBytes. Jeder Slot nimmt einen Block auf.

a) Wie werden Blöcke bei einem Versetzungsmaß I von 1, 2, 3, 4, 6, 7, 8 oder 9 auf einer Spur abgelegt?

Zwei Beispiele





b) Wieviele Umdrehungen sind erforderlich, um die m Blöcke in ihrer logischen Reihenfolge (sequentiell) zu lesen?

Pro Umdrehung können j= m/i Blöcke gelesen werden (so wird i gewählt). Insgesamt sind also m/j = i Umdrehungen notwendig.

c) Welche Vor-/Nachteile ergeben sich beim zufälligen Lesen einzelner Blöcke?

Keine (siehe auch Datenbanksysteme - Konzepte und Techniken der Realisierung, S. 79 f.)

Aufgabe 4: Zugriffsaufwand bei Dateien

182

Gegeben sei folgende SQL-Anfrage:

```
SELECT  *
FROM    PERS
WHERE   PNR = '123456'
```

Folgende Annahmen sollen gelten:

Die Relation PERS sei in Datei D1 abgelegt. Diese Datei bestehe aus 10^5 Seiten.

Für das Attribut PNR in PERS sei ein Index IPERS(PNR) als B*-Baum definiert. Dieser B*-Baum mit h*=3 sei in der Datei D2 abgelegt, die aus 10^6 Seiten bestehe. Die Blätter des B*-Baumes enthalten die Seitennummern Pi in der Datei D1, in der sich die zugehörigen PERS-Sätze befinden.

Wieviele (externe) Seitenzugriffe sind für D1 und D2 zur Beantwortung der SQL-Anfrage notwendig bei:

a) dynamischer Extent-Zuordnung

b) dynamischer Blockzuordnung (Vektor)

c) dynamischer Blockzuordnung (UNIX-Dateisystem):

a) D2 : 3 Zugriffe (Durchlauf des B*-Baums)

   D1 : 1 Zugriff
   Hier werden die Seitenadressen intern berechnet (die Tabelle ist klein!).

b) D2 : 6 Zugriffe

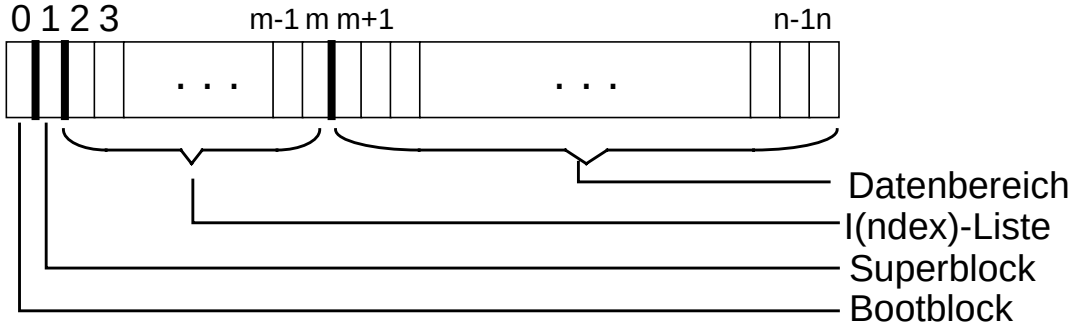
   D1 : 2 Zugriffe
   Der Vektor ist hier so groß, daß jeder Vektor-Zugriff auch einen Seitenzugriff erfordert.

c) D2 : 12 Zugriffe (3 * 4)

   D1 : 4 Zugriffe
   Hier wird stets der ungünstigste Fall betrachtet.

Kurzinfo: Das Dateisystem in UNIX

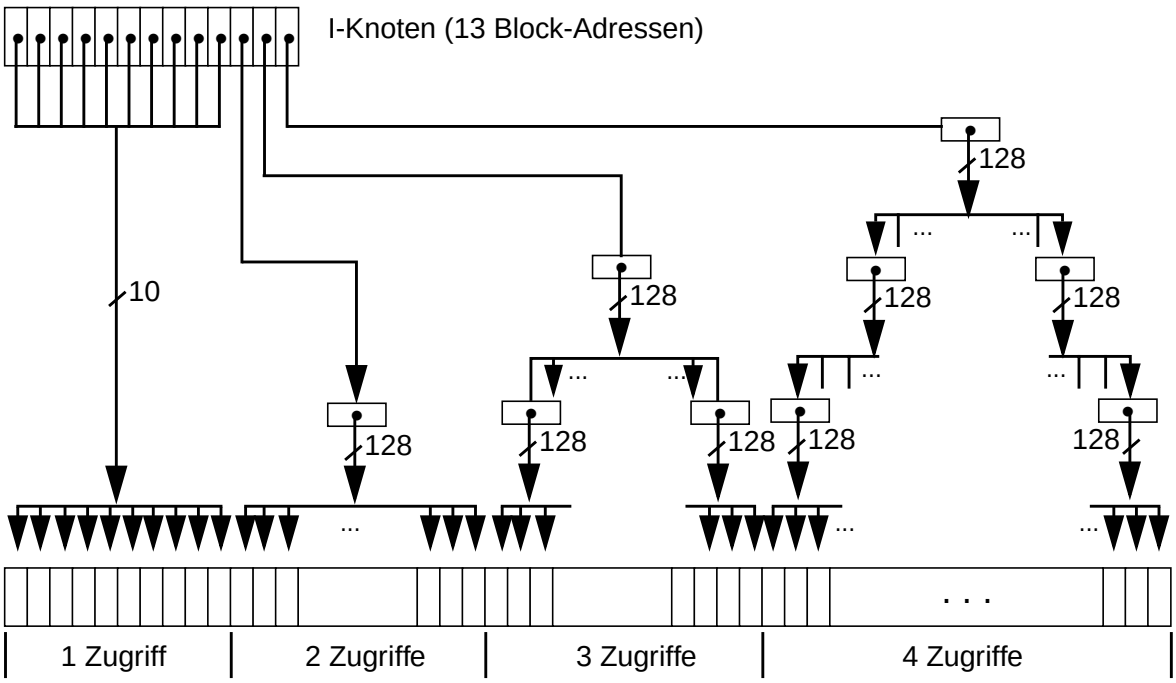
- Prinzipieller Aufbau (zur Organisation von Plattenspeicher)



- Wichtige Beschreibungsaspekte

  Directory-Einträge ordnen dem Datei-Namen einen Index I ($1 \leq I \leq m-1$) zu. Jeder Knoten der I-Liste enthält folgende Informationen:

  a. Identifikation des Datei-Eigentümers
  b. Schutzbits
  c. Adressen von 13 physischen Blöcken
  d. Größe der Datei in Bytes
  e. Zeitpunkt der Erstellung, letzten Referenz, letzten Modifikation
  f. Anzahl der Verweise auf die Datei
  g. Art der Datei



Die Dateiblöcke der untersten Ebene sind nur logisch zusammenhängend