Klausur Datenbanken
Wintersemester 2006/2007
Prof. Dr. Wolfgang May
12. Februar 2007, 11-13 Uhr
Bearbeitungszeit: 90 Minuten

Vorname:

Nachname:

Matrikelnummer:

Bei der Klausur sind **keine Hilfsmittel** (Skripten, Taschenrechner, etc.) erlaubt. Handies müssen ausgeschaltet sein. Papier wird gestellt. Benutzen Sie nur die **ausgeteilten**, zusammengehefteten **Blätter** für Ihre Antworten. Schreiben Sie mit blauem/schwarzem Kugelschreiber, Füller, etc.; Bleistift ist nicht erlaubt.

Auf dem letzten Blatt finden Sie eine Datenbasis, die in den Aufgaben 1 und 2 verwendet wird. Trennen Sie es ggf. zur Bearbeitung der Aufgaben ab.

Zum **Bestehen** der Klausur sind **45** Punkte hinreichend.

☐ meine Note soll mit Matrikelnummer so bald wie möglich auf der Vorlesungs-Webseite veröffentlicht werden.

☐ meine Note soll nicht veröffentlicht werden; ich erfahre sie dann aus Munopag/Wopag (bzw. CLZ: beim Prüfungsamt dort zu erfragen).

	Max. Punkte	Schätzung für "4"
Aufgabe 1 (ER-Modell)	15	10
Aufgabe 2 (SQL und Relationale Algebra)	35	17
Aufgabe 3 (Speicherung und Auswertung)	20	7
Aufgabe 4 (Transaktionen)	20	14
Summe	90	48

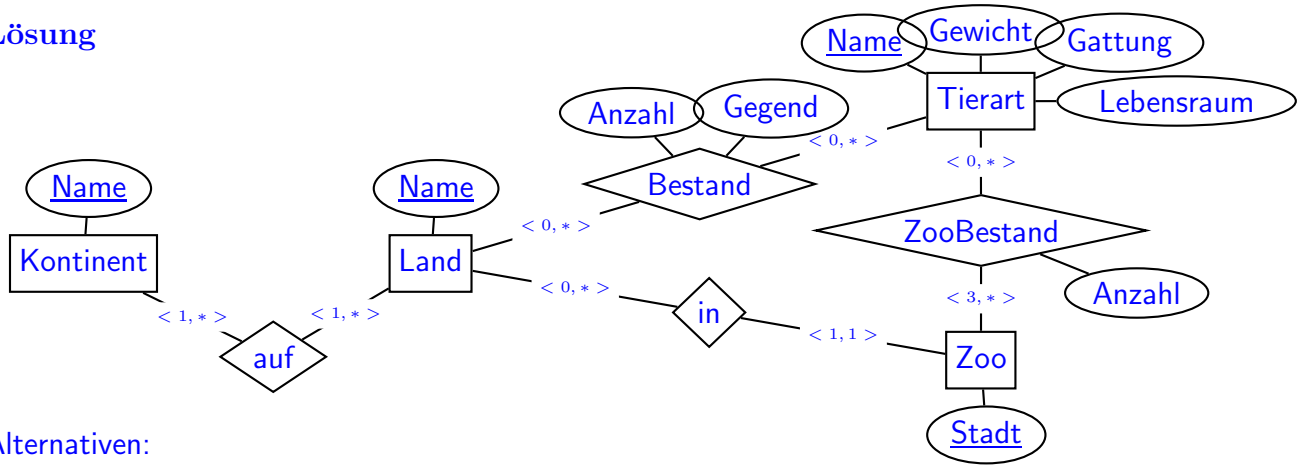
Note:

Aufgabe 1 (ER-Modell [15 Punkte])

Geben Sie ein ER-Modell für den folgenden Sachverhalt an (die für die SQL-Aufgaben angegebene Datenbasis ist ein Teil dieses Szenarios). Geben Sie Schlüssel und Kardinalitäten sinnvoll an.

- Es gibt Tierarten. Jede *Tierart* hat einen Namen (z.B. "Löwe", "Elch", "Panda", "Beutelteufel", ...). Jede Tierart gehört zu einer *Gattung*, und erreicht ein bestimmtes *Gewicht*: Löwen sind Raubkatzen und bis zu 250kg schwer, Elche sind Huftiere und bis zu 800kg schwer, und Beutelteufel sind Beuteltiere und bis zu 10kg schwer. Tierarten haben ausserdem einen bevorzugten *Lebensraum*. Löwen leben in der Savanne, Elche in der Tundra, Beutelteufel leben im Urwald.
- Es gibt *Länder*. Länder haben einen Namen.
- Es gibt *Kontinente*. Kontinente haben einen Namen, Länder liegen auf Kontinenten.
- Der weltweite, freilebende *Bestand* der verschiedenen Tierarten verteilt sich auf viele Länder, oft auch auf bestimmte Gebiete in diesen Ländern. Z.B. leben 300000 Elche in Schweden, 200000 Elche in Norwegen, aber nur noch ca. 50000 Beutelteufel in Australien und ca. 1600 Pandas in China, genauer gesagt in der zu China gehörenden Gegend "Sichuan".
- Ausserdem gibt es *Zoos*, die insbesondere auch bei der Erhaltung bedrohter Tierarten, wie z.B. dem Panda und dem Beutelteufel eine wichtige Rolle spielen. Zoos liegen in Städten, die wiederum in Ländern liegen; z.B. in Frankfurt in Deutschland. In jedem Zoo werden mindestens 3 Tierarten gehalten.
- In den Zoos gibt es ebenfalls (kleine) Bestände vieler Tierarten. Z.B. leben im Frankfurter Zoo 10 Elche, sowie 2 Pandas (und viele andere).

Lösung



Alternativen:

- Gattung, Lebensraum, Gegend als eigenständige Entitätstypen,
- Stadt als eigenständiger Entitätstyp (Zoo hat dann keine Attribute),
- Kontinent als *mehrwertiges* Attribut von Land (ist kein Key!),
- Bestand/Zoobestand als *schwache Entitätstypen*, die in <1,1>-Beziehungen zu Tierart und Land bzw. Zoo stehen (keine elegante Alternative).

Aufgabe 2 (SQL und Relationale Algebra [35 Punkte])
Verwenden Sie für Aufgabe 2 die Datenbasis, die auf dem letzten Blatt der Klausur angegeben ist.

1. Geben Sie die `CREATE TABLE`-Statements zur Generierung der beiden Tabellen `Zoo` und `ZooBestand` so komplett wie möglich an (10P).

   Lösung

```
CREATE TABLE Zoo                1.5P Gesamte Syntax
( Ort VARCHAR2(20) PRIMARY KEY,                          (1.5P)
  Land VARCHAR2(20) NOT NULL -- KEIN Fremdschluessel auf Land, da (1.5P)
    -- Land dort auch kein Schluessel ist (nur zusammen mit Kontinent)!
  );
CREATE TABLE ZooBestand
( Ort VARCHAR2(20) NOT NULL REFERENCES Zoo(Ort),          (1.5P)
  Tierart VARCHAR2(30) NOT NULL REFERENCES Tierart(Name), (1.5P)
  Anzahl NUMBER NOT NULL CHECK(Anzahl > 0)          (1.5P)
  PRIMARY KEY (Ort,Tierart));              (1P)
```

2. Geben Sie ein **SQL-Statement** an, das die Menge der Namen aller Raubkatzenarten angibt, die in deutschen Zoos gehalten werden. (3P)

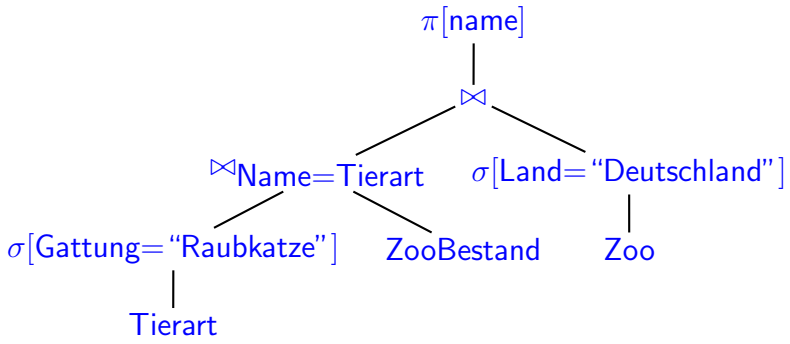
   Lösung

```
SELECT distinct name
FROM Tierart, Zoo, ZooBestand
WHERE Gattung="Raubkatze" AND Tierart.Name = ZooBestand.Tierart
  AND ZooBestand.Ort=Zoo.Ort AND Zoo.Land="Deutschland"
```

   (fehlendes distinct -0.5P)

3. Geben Sie einen **Ausdruck (oder Baum) der relationalen Algebra** an, der (2) beantwortet. (3P)

   Lösung



   Hinweis: das obere Join ist ein Natural Join über "Ort".

4. Geben Sie ein **SQL-Statement** an, das das folgende tut:
"Die Namen aller Tierarten, von denen es weltweit höchstens 50000 freilebende Exemplare gibt". (4 P)

Lösung Die perfekte Lösung ist:

```
SELECT Name
FROM Tierart
WHERE (SELECT sum(Anzahl)
       FROM Bestand
       WHERE Bestand.Tierart = Tierart.Name) < 50000;
```

(die berücksichtigt auch Tierarten, von denen es keine freilebenden Exemplare mehr gibt)

Die naheliegende Lösung

```
SELECT Tierart
FROM Bestand
GROUP BY Tierart
HAVING SUM(Anzahl) <= 50000;
```

wurde auch als korrekt gewertet.

5. Geben Sie ein **SQL-Statement** an, das das folgende tut:
"Alle Namen von Tierarten, die in Namibia frei leben, und nicht in deutschen Zoos vertreten sind". (5P)

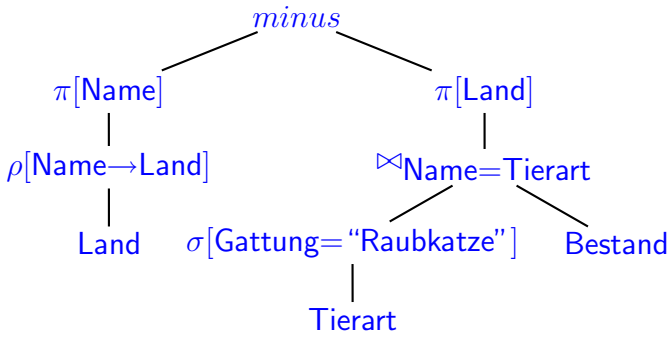
Lösung

```
(SELECT [distinct] Tierart  -- distinct hier nicht notwendig, da
 FROM Bestand               -- jede Tierart nur einmal im Bestand
 WHERE Land="Namibia")      -- fuer Namibia vorkommen kann.
MINUS
(SELECT Tierart
 FROM ZooBestand, Zoo
 WHERE ZooBestand.Ort=Zoo.Ort AND Zoo.Land="Deutschland")

SELECT [distinct] Tierart
FROM Bestand
WHERE Land="Namibia"
  AND Tierart NOT IN
    (SELECT Tierart
     FROM ZooBestand, Zoo
     WHERE ZooBestand.Ort=Zoo.Ort AND Zoo.Land="Deutschland")

SELECT [distinct] Tierart
FROM Bestand
WHERE Land="Namibia"
```

```
AND NOT EXISTS
  (SELECT *
   FROM ZooBestand, Zoo
   WHERE ZooBestand.Ort=Zoo.Ort
     AND Zoo.Land="Deutschland"
     AND ZooBestand.Tierart = Bestand.Tierart)
```

6. Geben Sie einen **Algebra-Ausdruck oder -Baum** an, der die Namen aller Länder ergibt, in denen es keine freilebenden Raubkatzen gibt (5P)

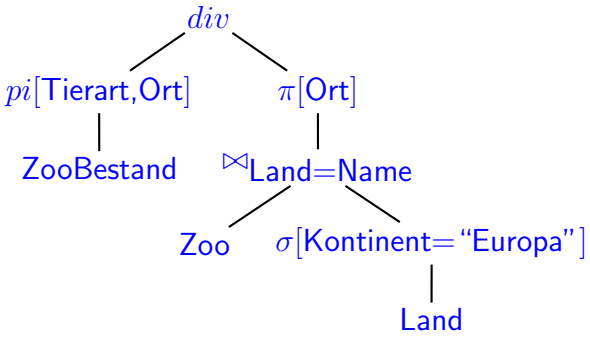
   Lösung

   (Hinweis: wenn im linken Ast "π[Land](Bestand)" anstatt "Land" als Basis genommen wird, fehlen alle Länder in denen es keine freilebenden Tierbestände gibt (z.B. Vatikan, Monaco).)

7. Geben Sie eine **SQL-Anfrage *oder* einen Algebra-Ausdruck/Baum** an (d.h. entscheiden Sie selber, welche Anfragesprache Sie bevorzugen), der folgende Anfrage beantwortet: "Namen aller Tierarten, von denen es in *jedem* europäischen Zoo mindestens ein Exemplar gibt." (5P).

   Lösung Offensichtlich eine relationale Division:

   Als SQL-Anfrage ist es deutlich komplizierter (mit doppelt geschachteltem Minus und/oder NOT EXISTS):

```
SELECT Name
FROM Tierart T
WHERE NOT EXISTS -- europ. Zoo, indem es diese Tierart nicht gibt
```

```
((SELECT Ort
  FROM Zoo, Land
  WHERE Zoo.Land = Land.Name AND Land.Kontinent= ‘‘Europa’’)
 MINUS
 (SELECT Ort
  FROM ZooBestand Z
  WHERE Z.Tierart = T.Name))
```

Aufgabe 3 (Speicherung und Auswertung [20 Punkte])
Gegeben sind zwei Tabellenschemata R und S:

R		
A	B	C

S	
E	F

Die Aufgabe behandelt das Join $R \bowtie_{R.A=S.E} S$.
Physikalisches Datenbankschema:

- Schlüssel von R ist (A, B), Schlüssel von S ist (E, F).
- Der Wertebereich von $R.A$ ist gleich dem Wertebereich von $S.E$, d.h. $\pi[A](R) = \pi[E](S)$.
- Die Tupel von R sind in beliebiger Reihenfolge gespeichert, es existieren keine Indexe auf R.
- Die Tupel von S sind so gespeichert, dass Tupel mit demselben E-Wert benachbart gespeichert sind. Eine weitergehende Ordnung existiert nicht.
- Für das Attribut $S.E$ ist ein B*-Baumindex angelegt, dessen Einträge jeweils auf das erste Tupel mit dem entsprechenden Wert von $S.E$ zeigen.

Daten:

- R enthaelt 2.000.000 Tupel [Hinweis: diese Schreibweise bedeutet in Deutschland "2 Millionen"].
- S enthaelt 4.000.000 Tupel.
- Die Tabellen sind wie üblich seitenweise abgelegt. Eine Seite (4KB) von R enthält 100 Tupel, eine Seite von S enthält 60 Tupel.
- Zu jedem Wert a von A gibt es genau 2 Tupel $\mu \in R$, so dass $\mu[A] = a$ ist.
- Zu jedem Wert e von E gibt es durchschnittlich 4 Tupel $\mu \in S$, so dass $\mu[E] = e$ ist.
- Für den B*-Baum über $S.E$ gilt folgendes: Jedes Blatt enthält durchschnittlich 100 Einträge. Jeder innere Knoten enthält durchschnittlich 200 Einträge.
- Im Hauptspeicher können 1000 Seiten gleichzeitig gehalten werden.

Fragen (jeweils mit kurzer Begründung zu beantworten):

1. Wieviele Werte enthält $\pi[A](R)$? (2P)
2. Wieviele Blätter enthält der B*-Baum über $S.E$? (2P)
3. Wie viele Ebenen hat der Baum?
   Wieviele Knoten enthält der Baum insgesamt? (5P)
4. Die Anfrage $\pi[A, B, F](R \bowtie_{R.A=S.E} S)$ soll berechnet und am Bildschirm ausgegeben werden.
   Wieviele Ergebnistupel bekommt man? (1P)
5. Beschreiben Sie, wie die Auswertung (unter Verwendung des Indexes) am besten vorgeht.

   Wieviele Hintergrundspeicherzugriffe auf die Datenbank werden benötigt?

   (beschreiben Sie am besten zuerst grob, wie Sie vorgehen, und analysieren Sie dann die Zugriffe) (ingesamt 10P)

Lösung Bei der Aufgabe wird nicht erwartet, dass alles komplett richtig ist, sondern dass die Argumentaton vernünftig ist.

1. Es sind 1.000.000 (jeder Wert von $R.A$ kommt genau zweimal vor) Werte (π macht Duplikateliminierung! - die relationale Algebra basiert auf *Mengen*). (2P)
2. Aus $\pi[A](R) = \pi[E](S)$ bzw. der Aussage, dass jeder Wert von $\pi[E](S)$ durchschnittlich 4x vorkommt folgt, dass es insgesamt 1.000.000 Einträge sind.

   Diese benötigen (durchschnittlich 100/Blatt) 10.000 Blätter. (2P)
3. 10.000 Blätter $\rightarrow$ 10.000/200 = 50 Knoten auf erster Ebene, diese hängen direkt unter der gemeinsamen Wurzel hängen (die eben nur 50 Kinder hat, und somit unterdurchschnittlich voll ist – die Wurzel ist der einzige Knoten im B*-Baum, der zu weniger als 50% gefüllt sein darf)).

   Der Baum hat damit 2 Knotenebenen sowie eine Ebene mit Blättern.

   Insgesamt sind es 10000+50+1=10051 Knoten.
4. Jedes R-Tupel wird mit durchschnittlich 4 S-Tupeln kombiniert. Das Ergebnis enthält 8.000.000 Tupel.

   Man benötigt keine Duplikateliminierung, da (A, B) Schlüssel von R ist und $(A = E, F)$ Schlüssel von S ist und somit keine Duplikate bei der Auswertung entstehen können.
5. Grober Ablauf: R durchgehen, für jeden Wert über den Index auf S zugreifen, Ergebnisse (durchschnittlich 4) nehmen, $\pi[A, B, F]$ darauf anwenden und dem Ergebnis hinzufügen. (3P)

   Feinablauf:
   Man liest erstmal die inneren Knoten des Index ein. 51 Zugriffe. Dieser wird immer wieder benötigt, also wird er im Hauptspeicher gehalten.

   Der gesamte Baum hat 10051 Seiten, passt also nicht in den Hauptspeicher!

   Dann geht man geht R seitenweise durch (insgesamt 2.000.000/100=20.000). Für jedes Tupel wird auf den Baum (innere Knoten im Hauptspeicher) zugegriffen, das entsprechende Blatt (ggf.) aus dem Hintergrundspeicher geholt, und damit dann auf S zugegriffen.

   Man wird jeweils eine Seite aus R und eine Seite aus S, sowie die 51 inneren Knoten des Baumes im Hauptspeicher haben, sowie ca. 900 Blätter (von 10000) des Baumes. Also hat man durchschnittlich in 9% der Fälle das gesuchte Blatt bereits im Speicher.

   In S liegen die durchschnittlich 4 benötigten Tupel hintereinander. Bei 60 Tupeln pro Seite hat man durchschnittlich in jedem zwanzigsten Zugriff (Beginn bei Tupel 58,59,60) einen Sprung auf die nächste Seite.

   Damit muss man insgesamt $2.000.000 \cdot (0.91 + 1 + 0.05) = 3.920.000$ mal auf S-Seiten zugreifen. (Hinweis: da $\pi[A](R) = \pi[E](S)$ gilt, ist jeder gesuchte Wert mindestens einmal in S vorhanden.)

   Da die Ergebnisse direkt ausgegeben werden sollen (und keine Duplikateliminierung notwendig ist), ist kein weiterer Zugriff zum Ablegen notwendig (Seitenweise Abspeicherung: Annahme dass 40 Tupel auf eine Seite passen, würde man weitere 200.000 Zugriffe haben).

   Insgesamt also 51+20.000+3.920.000 = 3.940.051 Zugriffe. (7P)

   [Korrekturhinweis: bereits die grobe Schätzung ohne Cache-Hits und Seitensprünge gab volle Punktzahl, weitere Feinheiten wurden mit Zusatzpunkten bewertet.]

Aufgabe 4 (Transaktionen [20 Punkte])
Gegeben ist der folgende Schedule:
$R_2B\ R_1A\ W_1C\ R_1B\ R_3C\ W_2B\ R_3B\ W_1A\ W_3B\ R_2A\ W_2A$

a) Geben Sie den Dependency-Graphen von S an (berücksichtigen Sie dabei auch die Transaktionen T_0 und T_∞ wie in der Vorlesung). (4P)

b) Geben Sie den Konfliktgraphen von S an. (4P)

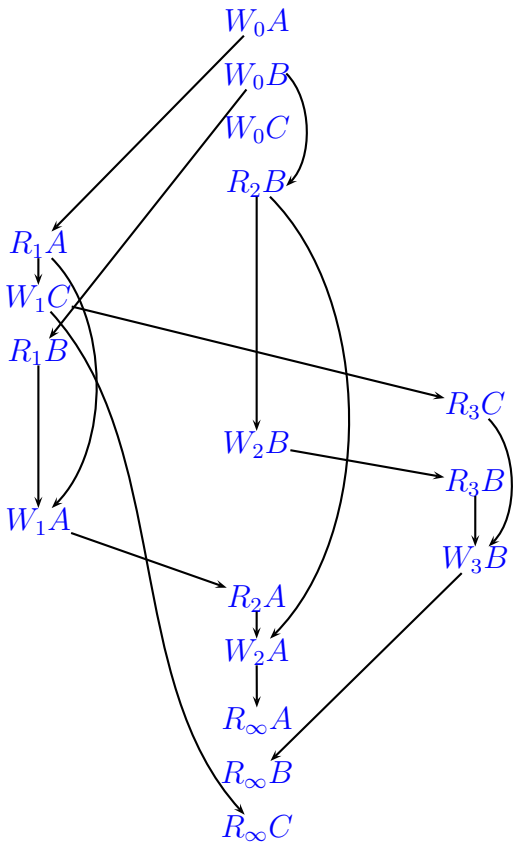
c) Ist S serialisierbar (mit Begründung) (2P)

d) Geben Sie für jedes der folgenden Scheduling-Verfahren an, ob der Schedule nach dem Verfahren möglich ist (mit Begründung; 10P):

- Überwachung des C-Graphen
- Zeitstempel
- Optimistisches Scheduling
- 2-Phasen-Locking

Lösung

- Dependency-Graph



- Konfliktgraph: $T_0 \to T_1 \to T_2 \to T_3 \to T_\infty$

  Der Konfliktgraph is zyklenfrei, also ist S serialisierbar.

  Hinweis: das "blind write" W_1C hat keinen Einfluß auf die Aussage. Nur für die Umkehrung "Wenn der CG zyklisch ist und es keine blind writes gibt, dann ist S nicht serialisierbar" wäre dies relevant (siehe Vorlesung!).

- Scheduling:
  - CG: ja, folgt aus dem Ergebnis von (c). Der CG ist zu jedem Zeitpunkt zyklenfrei. (2P)
  - Zeitstempel: nein. T_2 beginnt zuerst und bekommt $Z_2 = 1$, T_1 bekommt $Z_1 = 2$. mit W_1A bekommt A damit $Z_A = 2$, und R_2A später mit $Z_2 = 1$ ist nicht erlaubt. T_2 bekommt damit ein Rollback (und kann neu gestartet werden). (2P)
  - Optimistisches Verfahren: Beim Commit von T_1 fällt auf, dass es einen Konflikt von R_1B mit W_2B sowie von W_1C mit R_3C gibt. (2P)
  - 2PL: Nein. T_2 kann A (von T_1) erst bekommen (=lock), nachdem es B an T_3 abgegeben (=unlock) hat.

    Zwei Lösungen sind möglich, die beide dasselbe Ergebnis haben: Der Schedule ist unter 2PL nicht möglich (4P):
    - Unterscheidung zwischen Read- und Read+Write-Locks:
      Es geht gut bis $LR_2B\ R_2B\ LR_1A\ R_1A\ LW_1C\ W_1C\ LR_1B\ R_1B\ \underline{LW_1A}\ \underline{UW_1C}$ $LR_3C\ R_3C\ \underline{U_2B}\ LW_2B\ W_2B\ \underline{U_2B}\ LR_3B\ R_3B\ W_1A\ LW_3B\ W_3B\ \underline{U_1A}$, aber dann kann T_2 A nicht mehr locken um $R_2A\ W_2A$ ausführen zu können.
    - keine Unterscheidung zwischen R- und R+W-Locks:
      $L_2B\ R_2B\ L_1A\ R_1A\ L_1C\ W_1C$ jetzt wartet T_1 auf L_1B (T_2 hat B gelockt). R_1B ist also nicht als nächste Aktion möglich. Selbst wenn es jetzt erst mit $L_3C\ R_3C\ W_2B\ U_2B\ L_3B\ R_3B\ W_3B$ weitergehen kann, stoppt es spätestens bei R_2A, da T_2 dazu A locken müsste - das wiederum T_1 "gehört". Man befindet sich in einem Deadlock.

[Trennen Sie dieses Blatt am besten vor Beginn der Bearbeitung ab]
Die folgende Datenbasis wird in Aufgabe 2 verwendet.

Diese Datenbasis enthält die Daten über die weltweit freilebenden sowie in Zoos gehaltenen Bestände von verschiedenen Tierarten.

Tierart

Name	Gattung	Gewicht	Lebensraum
Elch	Huftier	800	Tundra
Warzenschwein	Huftier	120	Savanne
:	Huftier	:	:
Rotes Känguruh	Beuteltier	80	Savanne
Beutelteufel	Beuteltier	10	Urwald
:	Beuteltier	:	:
Panda	Bär	150	Wald
:	:	:	:

Land

Name	Kontinent
Deutschland	Europa
Schweden	Europa
China	Asien
Australien	Australien
Namibia	Afrika
:	:

Bestand

Tierart	Land	Gegend	Anzahl
Elch	Schweden		300000
Elch	Norwegen		200000
Elch	:		:
Warzenschwein	Namibia		1000000
Warzenschwein	:		:
Rotes Känguru	Australien		8000000
Beutelteufel	Australien	Tasmanien	50000
Panda	China	Sichuan	1600
:	:	:	:

Zoo

Ort	Land
Frankfurt	Deutschland
Berlin	Deutschland
Paris	Frankreich
Sydney	Australien
Peking	China
:	:

Dabei wird angenommen, dass der Name einer Stadt weltweit eindeutig ist

ZooBestand

Ort	Tierart	Anzahl
Frankfurt	Elch	5
Berlin	Warzenschwein	7
Berlin	Panda	2
Peking	Panda	10
Paris	Warzenschwein	10
Sydney	Beutelteufel	30
:	:	: