Klausur Datenbanken
Wintersemester 2003/2004
Prof. Dr. Wolfgang May
20. Februar 2004, 14-16 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.

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

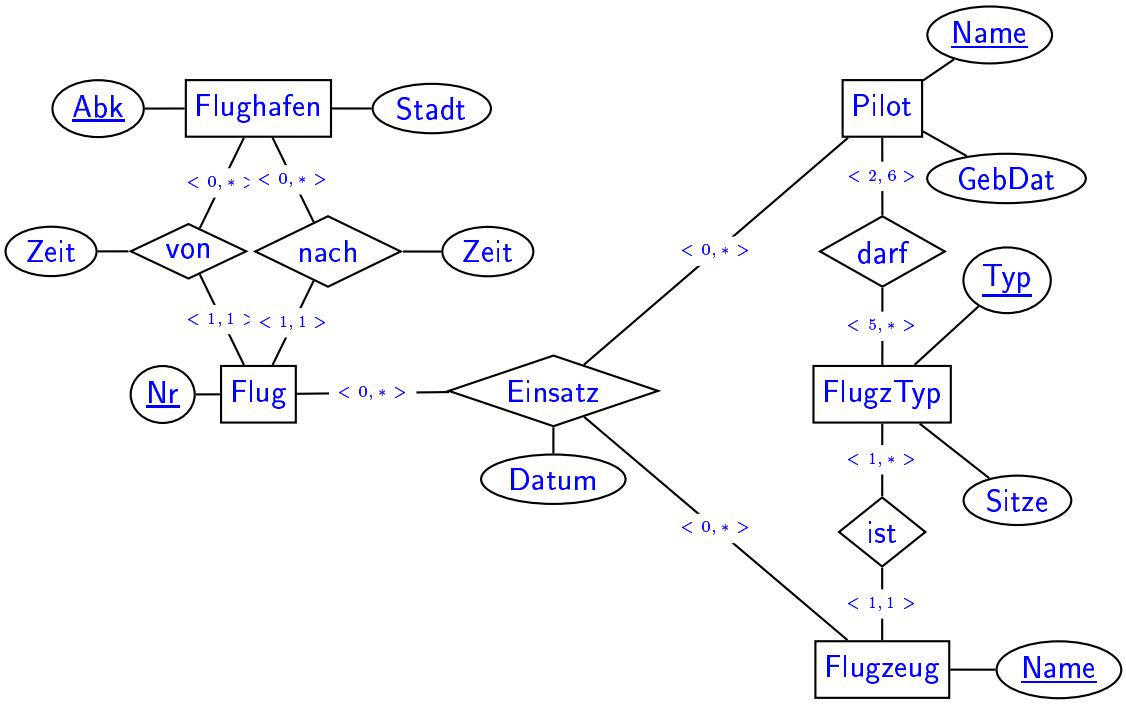
	Max. Punkte	Schätzung für "4"
Aufgabe 1 (ER-Modell, Relationales Modell)	27	15
Aufgabe 2 (SQL)	15	9
Aufgabe 3 (Relationale Algebra)	25	11
Aufgabe 4 (Transaktionen)	23	12
Summe	90	47

Note:

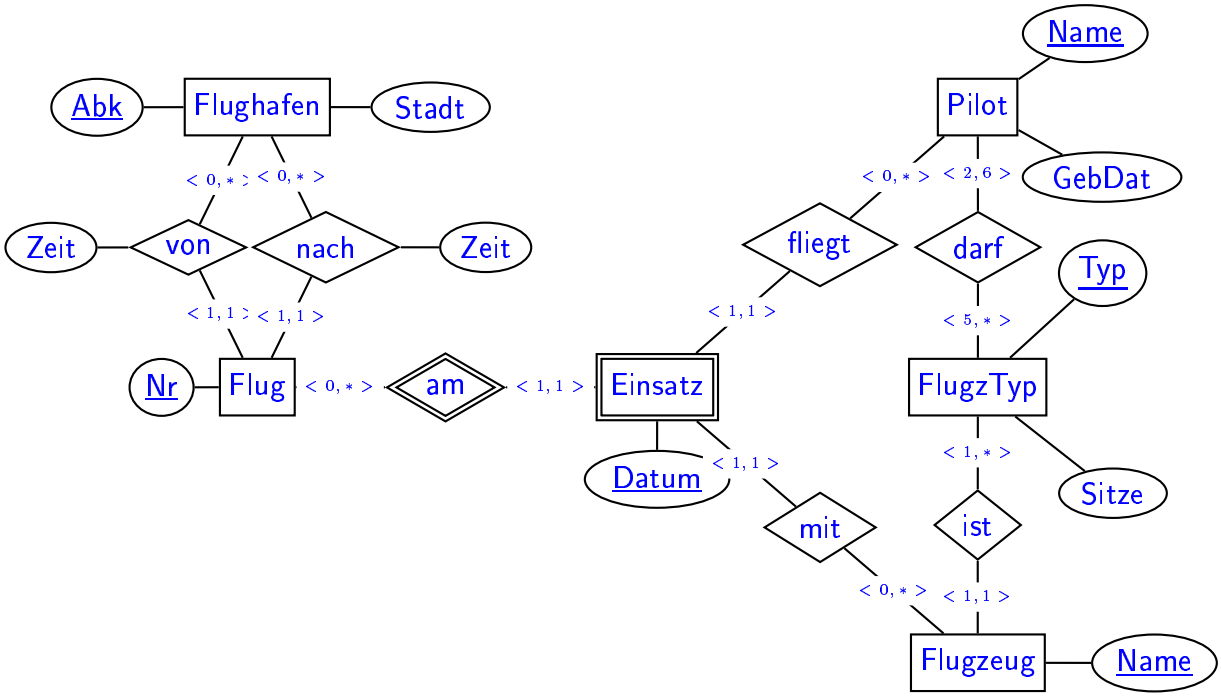
Aufgabe 1 (ER-Modell, Relationales Modell [27 Punkte])
Modellieren Sie die Datenbank einer Fluggesellschaft:

- Geben Sie ein ER-Diagramm für den folgenden Sachverhalt an (11 P):
  - Flughäfen haben ein Kürzel (Schlüssel) und gehören zu einer Stadt (z.B. ("FRA", "Frankfurt") und ("FCO", "Roma Fiumicino")).
  - Flüge haben eine Flugnummer (z.B. "LH 306"), führen von einem Flughafen zu einem anderen, mit jeweils einer festen Abflugs- und Ankunftszeit (z.B. ab *Frankfurt* um 07:30 nach *Roma Fiumicino* mit Ankunft um 09:15).
  - Jeder Flugzeugtyp hat einen Namen (z.B. "747-400") und eine Sitzanzahl (z.B. 430).
  - Piloten haben einen Namen (z.B. "Meier"); dieser ist Schlüsselattribut, ein Geburtsdatum (z.B. "1.1.1960"), und eine Berechtigung, bestimmte Flugzeugtypen zu fliegen (z.B. 747-400 und A310).
  - Jedes einzelne Flugzeug ist von einem bestimmten Flugzeugtyp (z.B. "747-400") und hat einen Namen, (z.B. "Beethoven").
  - bei einem Flug-Einsatz wird ein Flug (z.B. "LH 306") an einem bestimmten Datum (z.B. "20.2.2004") von einem Piloten (z.B. "Meier") mit einem bestimmten Flugzeug (z.B. "Beethoven") geflogen.
- ergänzen Sie das Diagramm um folgende Informationen (2 P):
  - Jeder Pilot darf mindestens zwei und höchstens 6 Flugzeugtypen fliegen.
  - Jeder Flugzeugtyp muss mindestens von 5 Piloten geflogen werden können.
  - Tragen Sie später weitere Kardinalitätsinformationen ein, die für den weiteren Entwurf relevant sind.

Lösung



Andere Möglichkeit: *Einsatz* auch als Entitätstyp. Dies ist dann ein schwacher Entitätstyp – Die Beziehung zu einem Flug ist die identifizierende Beziehung.



Andere Möglichkeit: Flug als Beziehung zwischen zwei Flughäfen; dann als Aggregation modellieren und Ergebnis in Beziehung zu einem Einsatz stellen.

- Transformieren Sie Ihr Diagramm in ein relationales Modell (8 P).
  Geben Sie Relationsnamen und Attribute an und unterstreichen Sie die Schlüsselattribute; z.B. Flughafen(Abk, Stadt). Geben Sie (auch zur Überprüfung Ihres Entwurfs) diejenigen Tupel an, die die im Text enthaltene Information repräsentieren.

- Überstreichen Sie Fremdschlüsselattribute (falls möglich mit einer anderen Farbe) (2 P).

Lösung

Flughafen(Abk, Stadt)	Flughafen(FRA, Frankfurt); Flughafen(FCO, Roma)
Flug(Nr, von, abZeit, nach, anZeit)	Flug(LH306, FRA, 07:30, FCO, 09:15)
FlugzTyp(Name, Sitze)	FlugzTyp(747-400, 430)
Pilot(Name, GebDat)	Pilot(Meier, 1.1.1960)
Flugzeug(Name, Typ)	Flugzeug(Beethoven, 747-400)
darf(Pilot, Flugzeugtyp)	darf(Meier, 747-400)
Einsatz(Flug, Datum, Flugzeug, Pilot)	Einsatz(LH 306, 20.2.2004, Beethoven, Meier)

(Korrektur: 3 P für Werte)

- Wählen Sie eine beliebige Tabelle aus, die mindestens 2 Fremdschlüssel enthält, und geben Sie die Tabellendefinition (in SQL: CREATE TABLE ...) sowie die referentiellen Integritätsbedingungen (in SQL: FOREIGN KEY ... REFERENCES ...) an (2+2 P).
(die Syntax muss nicht unbedingt SQL sein, aber es muss klar sein, was Sie meinen).

Lösung

```
CREATE TABLE einsatz
( Flug VARCHAR2(8) REFERENCES Flug(Nr),
  Datum DATE,
  Flugzeug VARCHAR2(20) REFERENCES Flugzeug(name),
  Pilot VARCHAR2(20) REFERENCES Pilot(Name))
```

Andere Notation z.B.:
einsatz.flug → flug.nr
einsatz.flugzeug → flugzeug.name
einsatz.pilot → pilot.name

Die folgende Datenbasis wird in Aufgaben 2 und 3 verwendet.

Betrachten Sie die folgende Datenbasis (eine Erweiterung der Pizzaservice-Datenbank aus der Vorlesung): Ein Pizzeria-Großhandel hat Lieferverträge mit Pizzerien, an die er jede Woche dieselben Produkte ausliefert (Annahme: es gibt keine zwei Pizzerien gleichen Namens, also ist *Name* Schlüssel der Relation *Kunde*).

Kunde		
Name	Ort	Strasse
Bella Italia	Göttingen	Weender Str. 8
Casino Grande	Kassel	Am Weinberg 14
Da MafIA	Göttingen	Lotzestrasse 16-18
Venezia	Kassel	Königsstrasse 111
:	:	:

Preisliste	
Produkt	Preis
Pizza	5.00
Lasagne	6.00
Gnocchi	4.50
Salat	3.00
:	:

Liefervertrag		
Pizzeria	Produkt	Anzahl
Bella Italia	Pizza	10
Bella Italia	Lasagne	15
Bella Italia	Salat	20
Casino Grande	Pizza	12
Casino Grande	Salat	15
Da MafIA	Gnocchi	60
Venezia	Pizza	20
:	:	:

Aufgabe 2 (SQL [15 Punkte])

1. Geben Sie eine SQL-Anfrage an, die die Namen aller Pizzerien ergibt, die Lasagne geliefert bekommen (1P).

   Lösung

   ```
   select pizzeria
   from liefervertrag
   where produkt = 'Lasagne'
         and Anzahl <> 0;  [darf auch fehlen]
   ```

2. Geben Sie eine SQL-Anfrage an, die die Menge aller Paare (Ort, Produkt) ergibt, so dass dieses Produkt an den angebenen Ort geliefert wird (2 P).

   Lösung

   ```
   select ort, produkt
   from kunde, liefervertrag
   where kunde.name = liefervertrag.pizzeria
   ```

   (fehlendes `distinct`: - 1/2 P)

3. Geben Sie eine SQL-Anfrage an, die für jede Stadt angibt, wieviele Portionen insgesamt in diese Stadt geliefert werden (3 P).

   Lösung

   ```
   select ort, sum(anzahl)
   from kunde, liefervertrag
   where kunde.name = liefervertrag.pizzeria
   group by ort
   ```

4. Geben Sie eine SQL-Anfrage an, die alle Namen und Orte der Pizzerien ergibt, die keinen Salat geliefert bekommen (4 P).

   Lösung

   ```
   select name, ort
   from Kunde
   where name not in
     (select pizzeria
      from liefervertrag
      where Produkt = 'Salat');

   select name, ort
   from Kunde k
   ```

```
where not exists
  (select *
   from liefervertrag l
   where Produkt = 'Salat'
     and l.pizzeria = k.name);   [haeufiger Fehler: letzte Zeile vergessen]

(select name, ort
 from kunde k)
minus
(select name, ort
 from kunde k
 where name in
  (select pizzeria
   from liefervertrag
   where Produkt = 'Salat'));
```

5. Geben Sie in natürlicher Sprache an, was die folgende Anfrage ergibt (3 P). Geben Sie weiterhin eine Zeile des Ergebnisses für die oben angegebene Beispieldatenbasis an (2 P).

```
select ort, sum(preisliste.preis * liefervertrag.anzahl)
from kunde, preisliste, liefervertrag
where kunde.name = liefervertrag.pizzeria
  and preisliste.produkt = liefervertrag.produkt
group by ort;
```

Lösung Gesamtpreis der wöchentlichen Lieferung für jeden Ort.
Ergebnistupel:
Bella Italia: 10 x Pizza zu 5E, 15 Lasagne zu 6E, 20 Salat zu 3E: 200E
Casino Grande: 12 x Pizza zu 5E, 15 Salat zu 3E: 105E
MafIA: 60 x Gnocchi zu 4.50E: 270E
Venezia: 20x Pizza = 100 E
Also Göttingen: 470 E, Kassel: 205 E

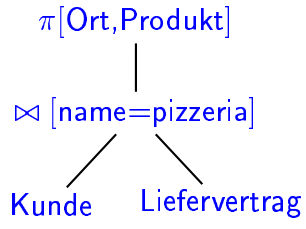
Aufgabe 3 (Relationale Algebra [25 Punkte])
Gegeben ist wieder die "Pizza-Großhandel"-Datenbasis aus Aufgabe 2.

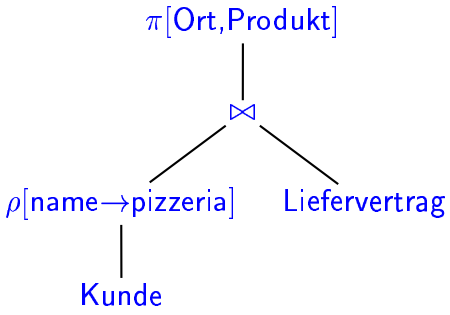
1. Geben Sie einen Algebra-Ausdruck oder -Baum an, der die Menge der Namen aller Pizzerien ergibt, die *Lasagne oder Gnocchi* geliefert bekommen (2P).

   Lösung
   π[pizzeria](σ[produkt = 'Gnocchi' or produkt = 'Lasagne'](liefervertrag))
   oder π[pizzeria](σ[produkt = 'Gnocchi'](liefervertrag))
   $\cup$ π[pizzeria](σ[produkt = 'Lasagne'](liefervertrag))

2. Geben Sie einen Algebra-Ausdruck oder -Baum an, der alle Paare (Ort, Produkt) ergibt, so dass dieses Produkt an den angebenen Ort geliefert wird (2 P).

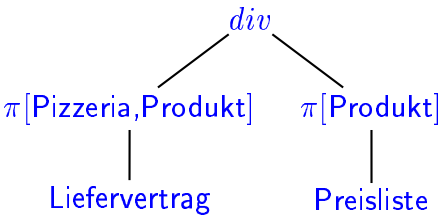
   Lösung

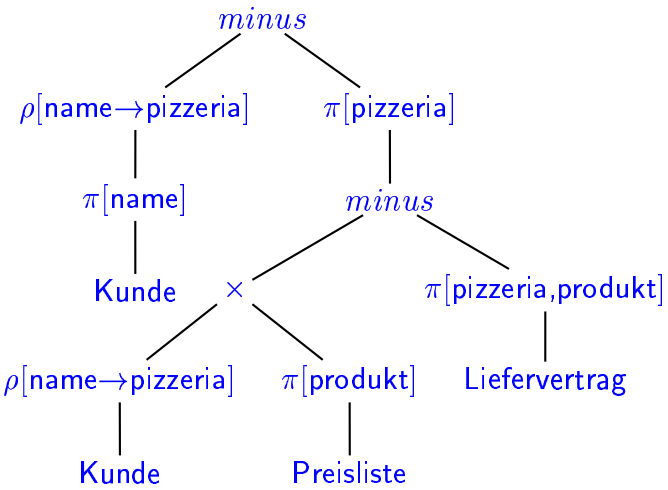




3. Geben Sie einen Algebra-Ausdruck oder -Baum an, der die Namen aller Kunden angibt, deren Lieferung *alle angebotenen Produkte* enthält (2 P).

   Lösung





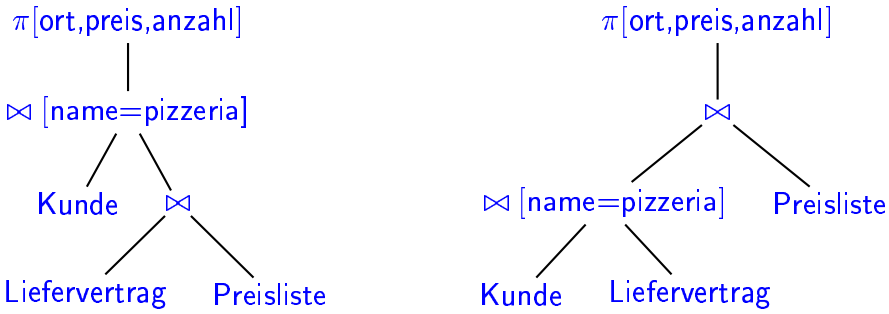
   Wichtig: Projektion von Liefervertrag muss die Anzahl rausnehmen
   Das Ergebnis hat nur noch eine Spalte "Pizzeria", also keine weitere Projektion mehr notwendig.

4. Geben Sie einen Algebra-**Baum** (Sie können auch einen Ausdruck angeben, Baum ist aber für den nächsten Aufgabenteil praktischer) für den "Hauptteil"

```
select ort, preis, anzahl
from kunde, preisliste, liefervertrag
where kunde.name = liefervertrag.pizzeria
  and preisliste.produkt = liefervertrag.produkt
```

der Anfrage aus Aufgabe 2.5 an (3 P).

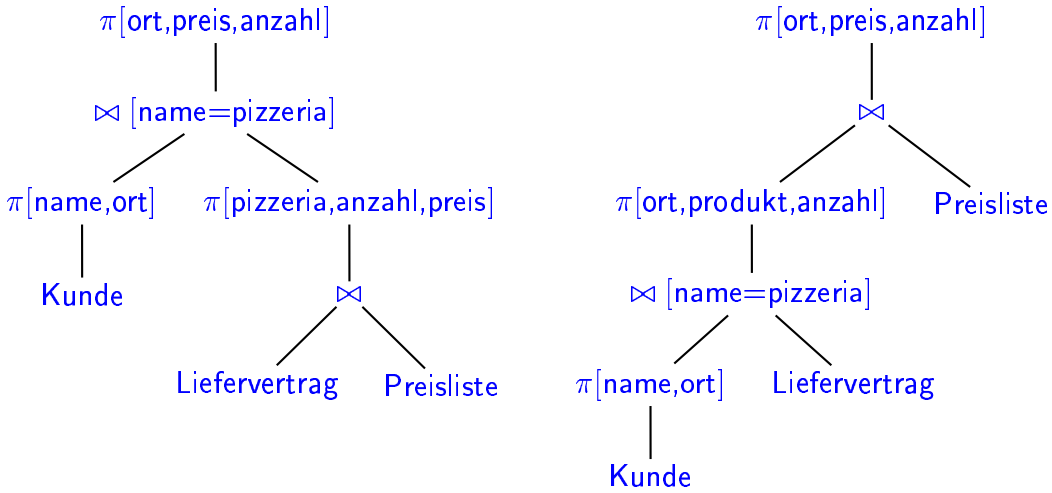
Lösung



Hinweis: das Join ohne angegebene Bedingung ist automatisch ein Equijoin über *Produkt*.

5. Ergänzen Sie den Algebra-Baum aus Aufgabenteil (4) mit Projektionen, so dass die Zwischenergebnisse möglichst klein sind (3 P).

Lösung



6. Ist die folgende Gleichung allgemeingültig (7 P)?

$$\sigma[cond](R_1 \bowtie R_2) = (\sigma[cond](R_1)) \bowtie R_2$$

- Falls ja: Beweisen Sie sie.
- Falls nein: Begründen Sie, warum sie nicht gilt und geben Sie ein Gegenbeispiel an.
- Geben Sie ein Beispiel an, in dem die Gleichung gilt.
- Hinweis: Sie dürfen die Beispiele anhand der obigen Datenbank oder anhand der Mondial-Datenbank angeben.

Lösung Die Gleichung ist so nicht allgemeingültig.

Sie gilt nicht, wenn die Bedingung (unter anderem) Attribute von R_2 verwendet. In diesem Fall kann die Bedingung auf R_1 nicht sinnvoll ausgewertet werden, das Ergebnis der rechten Seite ist dann je nach Implementierung/Interpretation leer (Algebra) oder der Ausdruck schon syntaktisch nicht definiert (SQL).

Beispiel: Alle Lieferungen von Salat nach Göttingen:
σ[Produkt="Salat" $\wedge$ Ort="Göttingen"](Liefervertrag $\bowtie$ Kunde) $\neq$
(σ[Produkt="Salat"$\wedge$ Ort="Göttingen"](Liefervertrag)) $\bowtie$ Kunde)

Die Gleichung gilt, wenn die Bedingung nur Attribute der ersten Relation betrifft. In diesem Fall ist es sinnvoll, sie bereits vor dem Join auszuwerten.

Beispiel: Alle Lieferungen von Salat:
σ[Produkt="Salat"](Liefervertrag $\bowtie$ Kunde) =
(σ[Produkt="Salat"](Liefervertrag)) $\bowtie$ Kunde)

7. Betrachten Sie ein Join von Liefervertrag mit Preisliste über die Produkt-Spalte (um z.B. für einen oder für alle Kunden den wöchentlichen Betrag zu berechnen). Beschreiben Sie, wie Sie *diesen Join* durch einen Index unterstützen können (6 P).

   - Welche Index-Datenstruktur wählen Sie. Geben Sie kurz die Komplexität der relevanten Operationen an.
   - Beschreiben Sie, wie der Join dann mit Hilfe des Indexes ausgewertet wird.

   Lösung Index auf das Attribut "Produkt" von "Preisliste".
   Baum-Index, üblicherweise B*-Baum: Operationen in $n \cdot \log n$.
   Oder Hash-Index. Operationen in $O(1)$, evtl. Überlaufbehandlung.

   Join: äußere Schleife über die Lieferverträge. Für jeden Liefervertrag wird auf den passenden Eintrag in der Preisliste in $O(\log |\text{Preisliste}|)$ oder $O(1)$ zugegriffen und das kombinierte Tupel in das Zwischenergebnis aufgenommen.

Aufgabe 4 (Transaktionen [23 Punkte])
Gegeben seien zwei Transaktionen:

- $T_1 = R_1A; A{=}A{-}10; W_1A; R_1B; B{=}B{+}10; W_1B$
- $T_2 = R_2A; A{=}A{-}20; W_2A; R_2C; C{=}C{+}20; W_2C$

sowie der folgende Schedule (nur die R/W-Aktionen sind aufgezählt):

$S = BOT_1\ R_1A\ W_1A\ BOT_2\ R_2A\ R_1B\ W_2A\ R_2C\ W_2C\ \text{commit}_2\ W_1B\ \text{commit}_1$.

(BOT_i: Begin of Transaction i; commit_i = erfolgreiches Ende von Transaktion i)

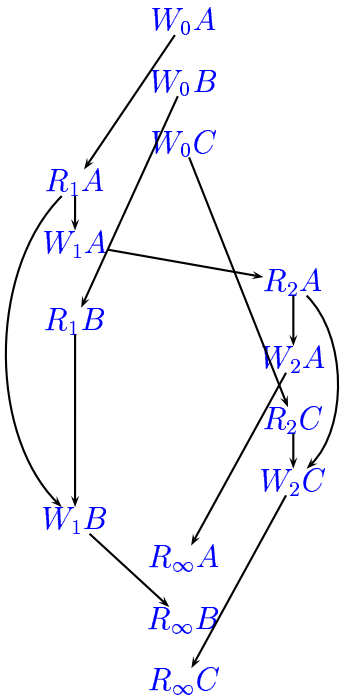
1. Geben Sie einen beliebigen seriellen Schedule S' an, der T_1 und T_2 enthält (1 P).
2. Seien die Startwerte $A = 50$, $B = 100$, $C = 5$ gegeben. Geben Sie die Werte von A, B und C nach dem Ablauf von S an (1 P).
3. Geben Sie den Dependency-Graphen von S an (berücksichtigen Sie dabei auch die Transaktionen T_0 und T_∞ wie in der Vorlesung) (5 P).
4. Geben Sie den Konflikt-Graphen von S an (4 P).
5. Ist S serialisierbar (mit Begründung; 1 P)?
6. Kann S von einem Scheduler, der das *2-Phasen-Protokoll* verwendet, erzeugt werden (3 P)?
   - Falls ja: Fügen Sie geeignete Lock- und Unlock-Befehle ein.
   - Falls nein: Begründen Sie.
7. Kann S von einem Scheduler, der das strenge *2-Phasen-Protokoll* verwendet, erzeugt werden (2 P)?
   - Falls ja: Fügen Sie geeignete Lock- und Unlock-Befehle ein.
   - Falls nein: Begründen Sie.
8. Kann S von einem Scheduler, der das *Zeitstempel-Protokoll* (in der Vorlesung auch als TO bezeichnet) verwendet, erzeugt werden? Begründen Sie Ihre Antwort. (2 P)
9. Betrachten Sie den folgenden Ablauf, in dem T_1 abgebrochen werden muss:

   $S = BOT_1\ R_1A\ W_1A\ BOT_2\ R_2A\ R_1B\ W_2A\ R_2C\ W_2C\ \text{commit}_2\ W_1B\ \text{abort}_1$.

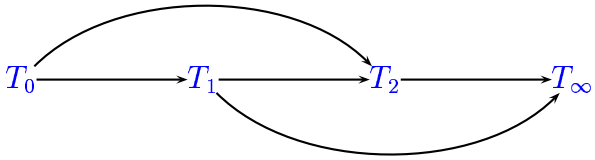
   - Was muss jetzt getan werden? (3 P)
   - Geben Sie die Werte von A und B am Ende aller in diesem Fall auszuführenden Aktivitäten an (1 P).

Lösung

- T_1 T_2 ist einer:
  R_1A W_1A R_1B W_1B R_2A W_2A R_2C W_2C
  (der zweite ist T_2 T_1).
- Endzustand: $A = 20$, $B = 110$, $C = 25$.
- Dependency-Graph:



Conflict Graph:



Der Conflict-Graph ist zyklenfrei, also ist S C-serialisierbar. Da die Transaktionen keine blind-writes enthalten, folgt daraus auch direkt die Serialisierbarkeit.

- 2-phasig: ja, mit folgenden Lock/Unlock-Aktionen:

  $S = BOT_1\ L_1A\ L_1B\ R_1A\ W_1A\ U_1A\ BOT_2\ L_2A\ R_2A$
  $R_1B\ W_2A\ L_2C\ R_2C\ W_2C\ U_2A\ U_2C$ commit$_2$ $W_1B\ U_1B$ commit$_1$.

  Wichtig ist, dass T_1 B früh genug lockt, bevor es A freigibt.
- streng 2-phasig: nein.
  Unter dem strengen 2-Phasen-Protokoll kann T_1 A erst beim commit freigeben - also könnte T_2 A nicht vorher locken/lesen/schreiben.
- Zeitstempel: Die "natürlich" in der Reihenfolge der BOT vergebenen Zeitstempel $T(1) = 1$, $T(2) = 2$ erlauben diesen Schedule. Wann immer T_1 auf etwas zugreift, ist dessen Zeitstempel noch 0 oder 1. (T_2 kommt immer nach T_1.)
- Alles was T_1 gemacht (geschrieben) hat muss zurückgenommen werden. Da T_2 bereits das von T_1 geschriebene A gelesen hat, muss T_2 (obwohl es bereits committed ist!) rückgängig gemacht werden. Die Werte danach sind wieder die Ursprungswerte. Hinweis: die Datenbank muss sich merken, dass beide Transaktionen de facto nicht ausgeführt wurden.

  Weitere korrekte Antwort: T_2 wird sofort von neuem ausgeführt, da sie bereits committed war. Es ergeben sich dann die Werte $A = 30$, $B = 100$, $C = 25$.