ILIAS-Beispielklausur "Einführung in Datenbanken"
Januar 2020
Basierend auf der Klausur von Wintersemester 2010/2011
Prof. Dr. Wolfgang May

- Diese Klausur dient als Beispiel- und Experimentalklausur mit dem ILIAS-System. Sie ist daher umfangreicher als üblich (Dauer: auf Papier ca. 120-150 Minuten, in Ilias ca. 150-180 Minuten)
- Bearbeiten Sie zuerst Aufgabe 1 (ER-Diagramm), dann *entweder* Aufgabe 2 (Tabellen in Textmode) oder Aufgabe 3 (Tabellen grafisch z.B. mit draw.io), und dann Aufgaben 4 bis 14 (diese bauen auf Aufgaben 1-3 auf); Aufgaben 15-21 sind davon unabhängig.
- Für Ausdrücke der relationalen Algebra sind in dieser Probeklausur die einfachen Aufgaben im Textmodus angelegt (schreiben Sie einfach Klammerausdrücke mit "pi", "sigma", "rho", "join" für die Operatoren), die komplexeren als pdf-upload (Bäume mit draw.io, oder sonstwas).

	Max. Punkte	Schätzung für "4"
Aufgabe 1 (ER-Modell)	20	
Aufgabe 2 (Transformation in das Rel. Modell (Textmode))	9	
Aufgabe 3 (Transformation in das Rel. Modell (grafisch))	9	
Aufgabe 4 (Relationales Modell CREATE TABLE)	7	
Aufgabe 5 (SQL und Relationale Algebra (a1))	2	
Aufgabe 6 (SQL und Relationale Algebra (a2))	2	
Aufgabe 7 (SQL und Relationale Algebra (b1))	3	
Aufgabe 8 (SQL und Relationale Algebra (b2))	3	
Aufgabe 9 (SQL und Relationale Algebra (c))	2	
Aufgabe 10 (SQL und Relationale Algebra (d1))	3	
Aufgabe 11 (SQL und Relationale Algebra (d2))	3	
Aufgabe 12 (SQL und Relationale Algebra (e))	7	
Aufgabe 13 (SQL und Relationale Algebra (f))	6	
Aufgabe 14 (SQL und Relationale Algebra (f2-upload))	6	
Aufgabe 15 (Abstrakt: SQL und Algebra)	5	
Aufgabe 16 (Theorie-Aufgabe)	5	
Aufgabe 17 (Algebra-Anfragebäume/Multiple-Choice)	8	
Aufgabe 18 (Schemaentwurf/Multiple-Choice)	4	
Aufgabe 19 (Ausdruckskraft - (Umordnungs-Beispiel))	4	
Aufgabe 20 (Transaktionen)	6	
Aufgabe 21 (Abb. ER-Modell auf Relationales Modell (a))	2	
Aufgabe 22 (Abb. ER-Modell auf Relationales Modell (b))	6	
Summe	122	

Note:

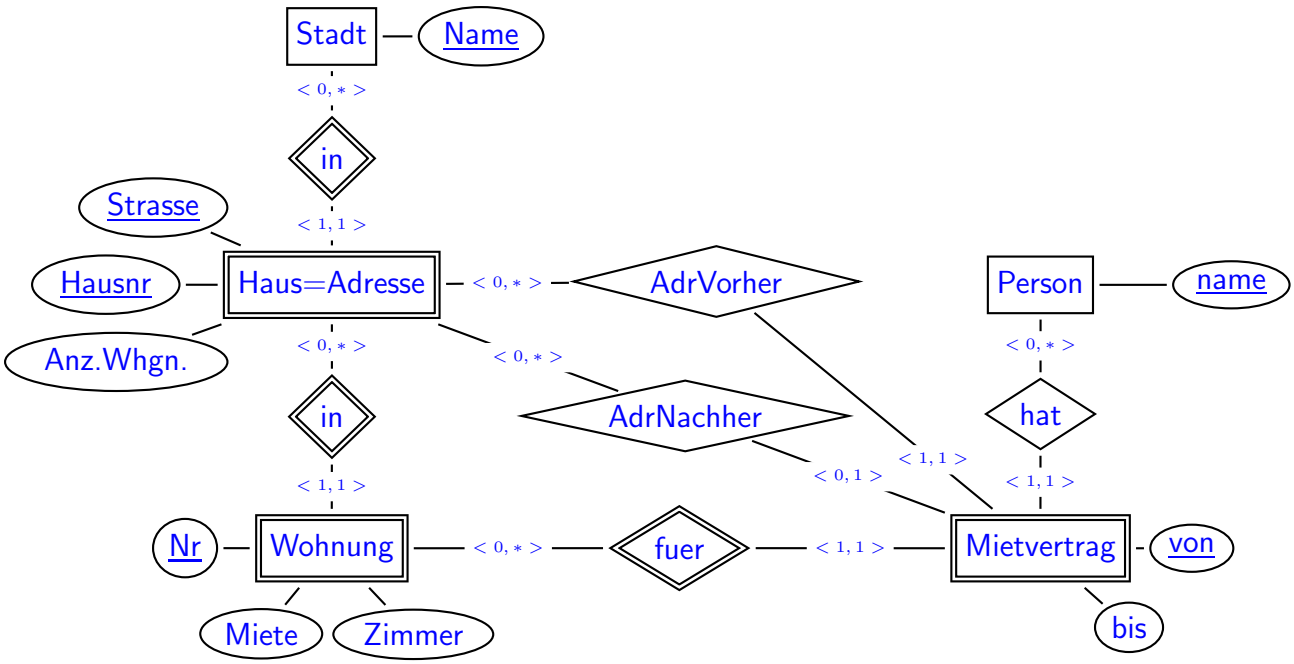
Themenstellung: Wohnungsgenossenschaft

Alle Klausuraufgaben basieren auf einem gemeinsamen "Auftrag": In der Klausur soll eine Datenbank einer Wohnungsgenossenschaft, die Mietwohnungen in Mehrfamilienhäusern in verschiedenen Städten in ganz Deutschland besitzt, entworfen werden:

1. Es wird angenommen, dass Städte durch ihren Namen eindeutig identifiziert sind.
2. Zu den einzelnen im Besitz der Genossenschaft befindlichen Häusern sind Adresse (Strasse mit Hausnummer) und Stadt abgelegt. Ausserdem ist gespeichert, wieviele Wohnungen in dem Haus sind.
   (a) Das Gebäude *Hauptstrasse 100* in *Göttingen* besteht aus 50 Wohnungen.
   (b) Das Gebäude *Rheinstrasse 53* in *Köln* besteht aus 83 Wohnungen.
3. Jede Wohnung hat eine Nummer innerhalb des Hauses. Zu jeder Wohnung sind die Anzahl der Zimmer, die Quadratmeter und die monatliche (Kalt)Miete gespeichert.
   (a) Die Wohnung mit der Nummer 42 des Hauses *Hauptstrasse 100* in *Göttingen* ist eine 3-Zimmer-Wohnung mit 80qm für 500 Euro im Monat.
   (b) Die Wohnung mit der Nummer 43 desselben Hauses ist eine 2-Zimmer-Wohnung mit 50qm für 320 Euro im Monat.
   (c) Die Wohnung mit der Nummer 17 des Hauses *Rheinstrasse 53* in *Köln* ist ein 1-Zimmer-Apartment mit 36qm und kostet 400 Euro im Monat.
4. Für jeden Mietvertrag ist gespeichert, wer welche Wohnung (und seit wann) gemietet hat (es wird angenommen, dass immer auf eine einzige Person als Mieter auftritt, d.h. den Mietvertrag unterschrieben hat). Ausserdem ist abgelegt, ob und zu welchem Zeitpunkt das Mietverhältnis gekündigt ist.
   (a) *Karl Napf* wohnt seit 1.4.2008 in der Wohnung Nummer 42 des Hauses *Hauptstrasse 100* in *Göttingen*. Das Mietverhältnis ist nicht gekündigt.
   (b) *Hans Dampf* wohnt seit 1.8.2001 in der Wohnung Nummer 43 des Hauses *Hauptstrasse 100* in *Göttingen*. Er hat die Wohnung zum 30.4.2011 gekündigt.
   (c) *Lieschen Müller* wohnt seit 1.3.1980 in der Wohnung Wohnung Nr. 17 des Hauses *Rheinstrasse 53* in Köln. Sie hat die Wohnung zum 28.2.2011 gekündigt.
5. Wenn ein neuer Mietvertrag abgeschlossen wird, wird dabei auch die (zu dem Zeitpunkt noch aktuelle) Adresse des Mieters sowie der Beginn des Mietverhältnisses gespeichert.
   (a) *Nils Pferd*, der zur Zeit in *Im Stall 4, Hintertupfing* wohnt (die Wohnung gehört nicht der Wohnungsgenossenschaft), hat einen Mietvertrag für die Wohnung Nummer 43 im Haus *Hauptstrasse 100* in *Göttingen* ab 1.5.2011 abgeschlossen.
   (b) Der oben genannte *Hans Dampf* hat einen am 1.4.2011 beginnenden Mietvertrag für die Wohnung Nr. 17 im Haus *Rheinstrasse 53* in *Köln* abgeschlossen.
6. Wenn ein Mieter auszieht wird die neue Adresse gespeichert (z.B. für die Nebenkostenabrechnung des Auszugsjahres).
   (a) Die neue Adresse von *Lieschen Müller* ist *Mühlengasse 24, Mühlhausen* in einem Haus, das nicht der Wohnungsgenossenschaft gehört.
7. Daten über alte Mietverträge werden nicht gelöscht.

Aufgabe 1 (ER-Modell [20 Punkte])
Entwickeln Sie ein ER-Modell für das Szenario. Geben Sie darin die Schlüsselattribute sowie die Beziehungskardinalitäten an.
[File Upload: pdf oder sowas, empfohlen: draw.io]

Lösung



Hinweise:

- *Stadt* bzw. *Person* nur als Attribute zu *Haus* und *Mietvertrag* geht auch.
- *Von* ist Schlüsselattribut von *Mietvertrag*, um den Fall dass eine Person zweimal zu unterschiedlichen Zeiträumen in einer Wohnung wohnt, abbilden zu können (-1/2P). Damit muß die *hat*-Beziehung nicht notwendigerweise als identifizierend ausgezeichnet werden, d.h. Wohnung + *von*-Datum reicht als Schlüssel aus. (Person und *von*-Datum würden nur ausreichen, wenn man annimmt, dass eine Person nicht gleichzeitig zwei Mietverträge beginnt).
- *AdrVorher* und *AdrNachher* können auch als Attribute zu *Mietvertrag* gemacht werden (man verliert aber die in Aufgabe (3e) wertvolle Assoziation, dass dies ja ggf. auf ein Haus der Genossenschaft verweisen kann). Sinnvoll ist auch dann eine Aufteilung in *Stadt* und *Adresse*.
- Anstatt der expliziten Verwendung des Entitätstyps *Mietvertrag* könnte man einfach eine Beziehung *mietet(e)* zwischen *Person* und *Wohnung* mit den Attributen *von* und *bis* verwenden.
  Dieses würde das Konzept des Mietvertrages abdecken, es wäre aber dann nicht möglich, damit die vorher/nachher-Adressen zu verbinden (diese sind dann also nur Attribute zu *mietet(e)*).
- *Ganz schlecht* ist es, *AdrVorher* und *AdrNachher* mit *Person* zu assoziieren, da man so keinen (zeitlichen) Zusammenhang mit den einzelnen möglicherweise aufeinanderfolgenden Mietverträgen speichern kann.

Aufgabe 2 (Transformation in das Rel. Modell (Textmode) [9 Punkte])
Bearbeiten eine der Aufgaben: Textmode (diese) oder grafisch (die nächste)
Geben Sie an, welche Tabellen (mit Attributen, Schlüsseln etc.) Ihre Datenbank enthält (keine SQL `CREATE TABLE`-Statements, sondern entsprechend "ASCII-Art" Pseudocode: (Empfehlung: editieren Sie es in einem lokalen File und mausen es am Ende ins Ilias, dann können Sie Ihr lokales File zur Bearbeitung der SQL-Aufgaben auch sehen).
Geben Sie die Tabellen mit jeweils mindestens zwei Beispieltupeln (z.B. denen, die sich aus dem Aufgabentext ergeben, und weiteren erfundenen) an.

- Tabellenschemata:

```
tabname(_attr1_,_attr2_,attr3,attr4)
-------------------------------------
         bsp11  bsp12   bsp13  bsp14
         bsp21  bsp22   bsp23  bsp24

oder analog
relname(attr1,attr2,attr3,attr4)  PK: (attr1,attr2)
```

- Angabe von Fremdschlüsselreferenzen:

```
rel1(A,B) -> rel2(X,Y)
bzw. bei einelementigen Fremdschluesseln einfach
rel1.A -> rel2.B
```

Lösung

- Tabellen für die Entitätstypen "Stadt" und "Person" werden nicht unbedingt benötigt, da sie nur eine Spalte "Name" enthalten würden.

Haus		
Adresse	Stadt	AnzWohnungen
Hauptstrasse 100	Göttingen	50
Rheinstrasse 53	Köln	83
:	:	:

(Strasse + Hausnummer in getrennten Spalten ist auch OK)

Wohnung					
Adresse	Stadt	WhgNr	Zimmer	Fläche	Preis
Hauptstrasse 100	Göttingen	42	3	80	500
Hauptstrasse 100	Göttingen	43	2	50	320
:	:	:	:	:	:
Rheinstrasse 53	Köln	17	1	36	400
:	:	:	:	:	:

Mietvertrag									
Adresse	Stadt	WhgNr	von	Person	Adr.Alt	StadtAlt	bis	Adr.Neu	Stad
Hauptstr.100	Göttingen	42	1.4.2008	Karl Napf	null	null	null	null	null
Hauptstr.100	Göttingen	43	1.8.2001	Hans Dampf	null	null	30.4.2011	Rheinstr.53	Köln
Rheinstr.53	Köln	17	1.3.1980	L.Müller	null	null	28.2.2011	Mühleng.24	Müh
Rheinstr.53	Köln	17	1.4.2011	Hans Dampf	Hauptstr.100	Göttingen	null	null	null
Hauptstr.100	Göttingen	43	1.5.2011	Nils Pferd	Im Stall 4	H'tupfing	null	null	null
:	:	:	:	:	:	:	:	:	:

- Man kann auch aufteilen:
  PersonEinzug: *Name,Adresse,Stadt*, *Datum*, *StadtAlt*, *AdresseAlt*
  PersonAuszug: *Name,Adresse,Stadt*, *Datum*, *StadtNeu*, *AdresseNeu*
  Wichtig ist, dass Name und Datum dabei sind, wenn jemand nacheinander mehrere Wohnungen mietet (oder auch dieselbe einige Zeit später nochmal).
- *StadtAlt*, *AdresseAlt* und *StadtNeu*, *AdresseNeu* sind keine Foreign Keys auf *Haus(StadtAdresse)*, da sie nur dann Referenzen sind, wenn die betreffenden Häuser auch im Besitz der Genossenschaft sind.

Aufgabe 3 (Transformation in das Rel. Modell (grafisch) [9 Punkte])
Bearbeiten eine der Aufgaben: grafisch (diese) oder Textmode (die vorhergehende)
Geben Sie an, welche Tabellen (mit Attributen, Schlüsseln etc.) Ihre Datenbank enthält (keine SQL `CREATE TABLE`-Statements, sondern einfach grafisch.
Markieren Sie dabei auch Schlüssel (durch unterstreichen – das ist in draw.io ganz einfach) und Fremdschlüsselreferenzen (durch überstreichen (frei malen)) oder durch Angabe von Fremdschlüsselreferenzen:

```
rel1(A,B) -> rel2(X,Y)
bzw. bei einelementigen Fremdschluesseln einfach
rel1.A -> rel2.B
```

Aufgabe 4 (Relationales Modell CREATE TABLE [7 Punkte])
Geben Sie das `CREATE TABLE`-Statement für diejenige Tabelle (bzw. die Tabellen), in der bei Ihnen die Daten über die Wohnungen abgespeichert sind, so vollständig wie möglich an.

Lösung

```
CREATE TABLE wohnung                                    Basis 4P
( adresse VARCHAR2(30),
  stadt   VARCHAR2(30),
  whgnr   NUMBER CHECK (whgnr > 0),
  zimmer  NUMBER NOT NULL CHECK (zimmer > 0),           1/2 Checks
  flaeche NUMBER NOT NULL CHECK (flaeche > 0),          1/2 NOT NULL
  preis   NUMBER NOT NULL CHECK (preis > 0),
CONSTRAINT whgkey PRIMARY KEY (adresse, stadt, whgnr),  1P PKEY
CONSTRAINT whghaus FOREIGN KEY (adresse, stadt)
                   REFERENCES haus(adresse,stadt) )     1P FKEY
```

Aufgabe 5 (SQL und Relationale Algebra (a1) [2 Punkte])
Verwenden Sie für diese und die folgenden Aufgaben die von Ihnen entworfene relationale Datenbasis. Keine der Antworten soll Duplikate enthalten.

Geben Sie **eine SQL-Anfrage** an, die die Namen aller Städte ausgibt, in denen die Genossenschaft mindestens eine 4-Zimmer-Wohnung besitzt.

Lösung

```
select distinct stadt
from wohnung
where zimmer = 4;
```

Aufgabe 6 (SQL und Relationale Algebra (a2) [2 Punkte])
Geben Sie **einen Algebra-Ausdruck** an, der die Namen aller Städte ausgibt, in denen die Genossenschaft mindestens eine 4-Zimmer-Wohnung besitzt.

Lösung

π[stadt]
|
σ[zimmer=4]
|
wohnung

Aufgabe 7 (SQL und Relationale Algebra (b1) [3 Punkte])
Geben Sie **eine SQL-Anfrage** an, die die Namen aller Personen ausgibt, die gegenwärtig eine 4-Zimmer-Wohnung mit mindestens 100qm für weniger als 600E gemietet haben.

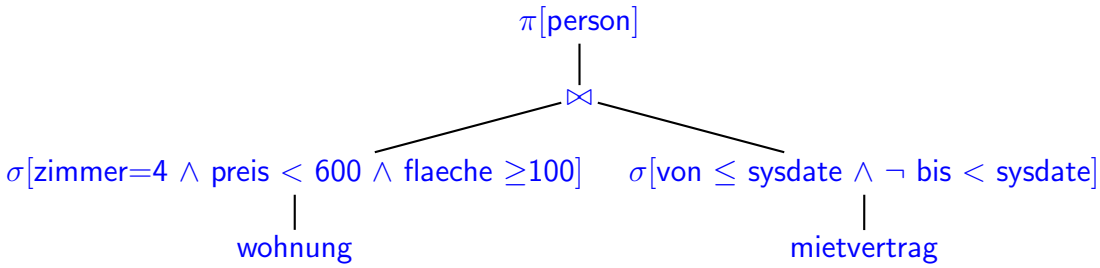
Lösung

```
select distinct person
  -- distinct (-1/2P) ist wichtig, sonst sind alle Personen, die
  -- gerade umziehen, mehrfach dabei
from mietvertrag mv, wohnung w
where mv.adresse = w.adresse
  and mv.stadt = w.stadt
  and mv.whgnr = w.whgnr
  and w.zimmer = 4 and w.preis < 600 and w.flaeche >= 100
  and mv.von <= sysdate and not bis < sysdate;

(beachten: bei gegenwaertigen Vertraegen ist "bis" NULL oder > dem
  aktuellen Datum, und "von" < dem aktuellen Datum (vgl.
  Vertrag mit Nils Pferd, der schon gespeichert ist, aber erst
  am 1.5. beginnt)    (-1/2 P wenn falsch)

select distinct person
from mietvertrag
where (adresse,stadt,whgnr) in
             (select adresse, stadt, whgnr
              from wohnung
              where zimmer = 4 and preis < 600 and flaeche >= 100)
```

```
  and von <= sysdate and not bis < sysdate;
  (oder "9.2.2011" direkt einsetzen)
-- geht auch ganz aehnlich mit  ... where exists (... SFW ...)
```

Aufgabe 8 (SQL und Relationale Algebra (b2) [3 Punkte])
Geben Sie **einen Algebra-Ausdruck** an, der die Namen aller Personen ausgibt, die gegenwärtig eine 4-Zimmer-Wohnung mit mindestens 100qm für weniger als 600E gemietet haben.

Lösung Der Algebra-Ausdruck entspricht der ersten Lösung der vorhergehenden Aufgabe:



Aufgabe 9 (SQL und Relationale Algebra (c) [2 Punkte])
Geben Sie **eine SQL-Anfrage** an, die für jede Stadt ausgibt, wieviel Miete die Genossenschaft insgesamt monatlich in dieser Stadt bekommen kann.

Lösung

```
select stadt, sum(miete)
from wohnung
group by stadt;
```

Aufgabe 10 (SQL und Relationale Algebra (d1) [3 Punkte])
Geben Sie **eine SQL-Anfrage** an, die die Namen aller Personen ausgibt, die irgendwann in einer der Wohnungen der Wohnungsgenossenschaft gewohnt haben, und die aber nie in Köln in einer Wohnung der Wohnungsgenossenschaft gewohnt haben.

Lösung

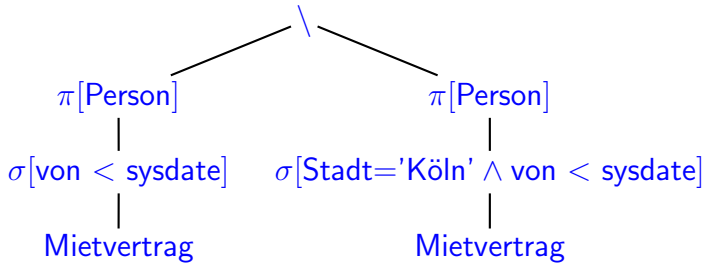
```
(select person
 from mietvertrag
 where von < sysdate)
minus
(select person
 from mietvertrag
 where Stadt = 'Koeln'
   and von < sysdate)
```

```
select person
from mietvertrag m1
where von < sysdate
  and not exists
   (select *
    from mietvertrag m2
    where von < sysdate
    and m1.person=m2.person
      and m2.stadt = 'Koeln')
```

```
                                -- analog auch mit
                                -- where person not in (...)

-- das "von < sysdate" hatte fast niemand, gab ggf. +1/2 Punkt.
```

Aufgabe 11 (SQL und Relationale Algebra (d2) [3 Punkte])
Geben Sie **einen Algebra-Ausdruck oder -Baum** an, der die Namen aller Personen ausgibt, die irgendwann in einer der Wohnungen der Wohnungsgenossenschaft gewohnt haben, und die aber nie in Köln in einer Wohnung der Wohnungsgenossenschaft gewohnt haben.

Lösung



Aufgabe 12 (SQL und Relationale Algebra (e) [7 Punkte])
Geben Sie eine Aufgabenstellung (Textaufgabe) an, deren Lösung eine relationale Division erfordert. (Wenn ihnen nichts sinnvolles einfällt, können Sie dazu auch eine weiteres Attribut zu einer der Tabellen, bzw. eine neue Tabelle hinzufügen).
Geben Sie dazu eine SQL-Anfrage **oder** einen Algebra-Ausdruck an.

Lösung Ein einfaches Beispiel ist "diejenigen Personen, die schon in allen Städten eine Wohnung gemietet hatten, in denen Lieschen Müller eine Wohnung gemietet hatte".

π[Person,Stadt](Mietvertrag) div (π[stadt](σ[name='L.M.'](Mietvertrag)))

```
select x.Person
from (select distinct person from mietvertrag) x
where not exists (select *
                  from mietvertrag m
                  where name='Lieschen Mueller'
                    and not (x.Person, m.stadt) IN
                         (select Person,stadt
                          from mietvertrag))
```

Oder: Der Entitätstyp bzw. die Tabelle Stadt (die man bei dieser Lösung noch hinzufügen muss) bekommt ein Attribut "Population", und man fragt dann an "welche Personen hatten in jeder Stadt mit mehr als einer Million Einwohnern einen Mietvertrag bei dieser Wohnungsgenossenschaft":

π[Person,Stadt](Mietvertrag) div ρ[name$\rightarrow$Stadt](π[name](σ[pop>1000000](Stadt)))

```
select x.Person
from (select distinct person from mietvertrag) x
where not exists (select *
```

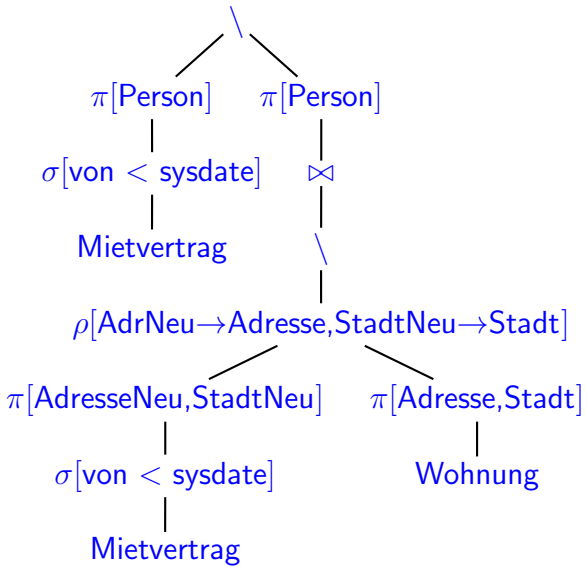
```
from stadt
where population > 1000000
  and not (x.Person, stadt.name) IN
       (select Person,stadt
        from mietvertrag))
```

Aufgabe 13 (SQL und Relationale Algebra (f) [6 Punkte])
Gibt es eine Möglichkeit, mit SQL aus der Datenbasis alle Personen herauszufinden, die seit ihrem erstmaligen Einzug ausschliesslich, ggf. aber beliebig oft, von einer der Genossenschaft gehörenden Wohnung direkt in eine andere, auch der Genossenschaft gehörende Wohnung gezogen sind?

Falls ja, geben Sie eine solche Anfrage (SQL oder Algebra) an. Falls nein, begründen Sie Ihre Antwort.

Lösung Ja. Alle, die nie in eine nicht der Genossenschaft gehoerende Wohnung umgezogen sind (also einschliesslich derer, die bisher noch nie ausgezogen sind).

```
SELECT DISTINCT person
FROM mietvertrag m1  -- um alle bekannten Personen zu betrachten
WHERE m1.von < sysdate
WHERE NOT EXISTS -- Mietvertrag dieser Person, aus der diese Person
                 -- in eine nicht der G. gehoerende Whg gezogen ist
 (SELECT *
  FROM mietvertrag m2
  WHERE m1.Person = m2.Person
    AND m2.von < sysdate
    AND bis IS NOT NULL
    AND -- Nachfolgewohnung gehoert nicht der Genossenschaft
        (AdrNeu, StadtNeu) NOT IN
          (SELECT Adresse, Stadt
           FROM Wohnung))

(SELECT person
 FROM mietvertrag
 WHERE m1.von < sysdate)
MINUS
(SELECT person
 FROM mietvertrag
 WHERE von < sysdate
   AND bis IS NOT NULL
   AND -- Nachfolgewohnung gehoert nicht der Genossenschaft
       (AdrNeu, StadtNeu) NOT IN
         (SELECT Adresse, Stadt
          FROM Wohnung))
```

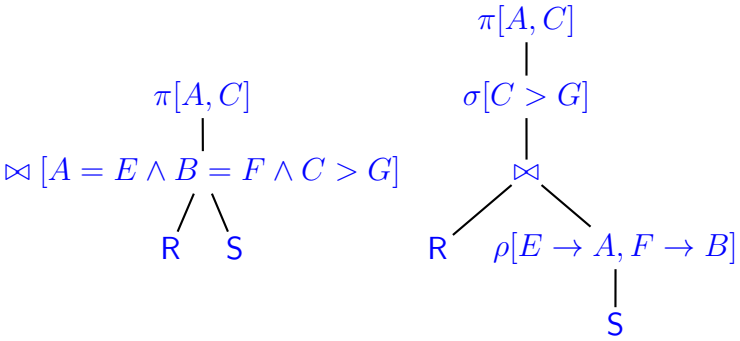


Aufgabe 14 (SQL und Relationale Algebra (f2-upload) [6 Punkte])
Falls Sie zu der vorhergehenden Aufgabe (f) einen Algebra-Ausdruck oder -Baum als Grafik-pdf hochladen wollen, können Sie dies hier tun:

Aufgabe 15 (Abstrakt: SQL und Algebra [5 Punkte])
Gegeben sind zwei Relationen $R(A, B, C, D)$ und $S(E, F, G, H)$. Die Attribute $R.C$ und $S.G$ sind numerisch, alle anderen sind Strings.
Geben Sie einen **Algebra-Ausdruck oder -Baum** an, der äquivalent zu der folgenden SQL-Anfrage ist:

```
SELECT A,C
FROM R
WHERE (A,B) IN (SELECT E,F
                FROM S
                WHERE G < C)
```

Lösung Alle Subqueries werden in Joins umgewandelt. Hier:



Aufgabe 16 (Theorie-Aufgabe [5 Punkte])
Seien R und S zwei Relationen.

Geben Sie an, ob, und ggf. unter welchen Bedingungen bzgl. R und S die beiden untenstehenden Ausdrücke der relationalen Algebra äquivalent sind.
Begründen Sie Ihre Aussage.
Beschreiben Sie ggf. auch, bzw. geben Sie ein Gegenbeispiel an, wo die beiden Ausdrücke nicht äquivalent sind.

1. $R \cup S$

2. $R \; \overline{\bowtie} \; S$

Lösung (1) ist erlaubt, wenn R und S dasselbe Format, d.h. dieselben Attribute haben. Unter diesen Bedingungen ist $R \bowtie S = R \cap S$ und $R \setminus (R \ltimes S) = R \setminus (R \cap S) = R \setminus S$ (analog auch symmetrisch).

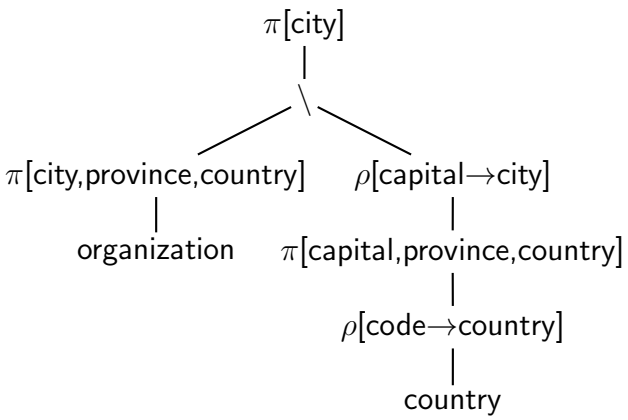
Und $R \; \overline{\bowtie} \; S = R \bowtie S \cup ((R \setminus (R \ltimes S)) \times \{\}) \cup ((S \setminus (S \ltimes R)) \times \{\}) = R \cap S \cup (R \setminus S) \cup (S \setminus R) = R \cup S$,
wobei $\{\}$ hier das null-spaltige Tupel mit Nullwerten für die (nicht vorhandenen) Attribute in R aber nicht in S (und umgekehrt) beschreibt.
Wenn R und S nicht dasselbe Format haben, ist (1) nicht erlaubt, aber (2) durchaus und (2) ist ein sinnvoller Ausdruck, der dann eben nicht zu (1) äquivalent ist.

Aufgabe 17 (Algebra-Anfragebäume/Multiple-Choice [8 Punkte])
Diese Aufgabe benutzt das Mondial-Schema:

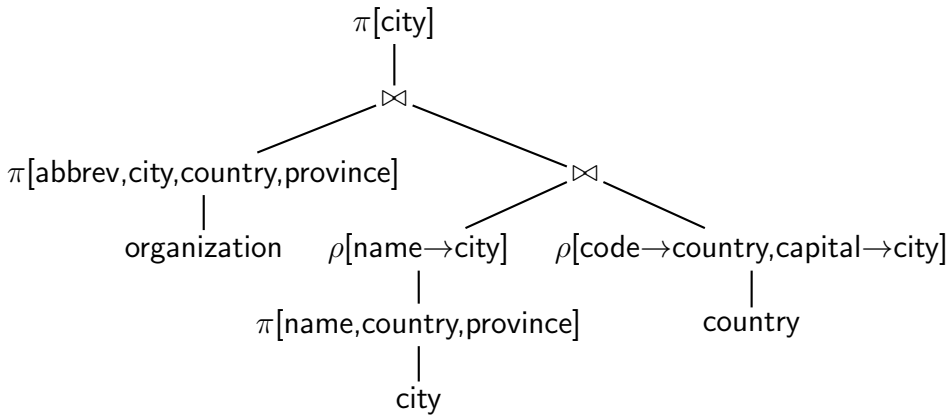
```
country(name,code,capital,province,area,population)
    (code,capital,province) geben den Landescode+Stadtname+Provinz der Hauptstad
organization(abbreviation,name,city,country,province,established)
    (city,country,province) geben die Stadt an, in der die Organisation ihren Si
city(name,country,province,population,latitude,longitude,elevation)
ismember(country,organization,type)
```

Welche der folgenden Algebra-Bäume beantworten die Anfrage "die Namen aller Städte, die die Hauptstadt eines Landes sind, und in denen eine Organisation ihren Sitz hat" korrekt?
(es können mehrere Bäume richtig sein)

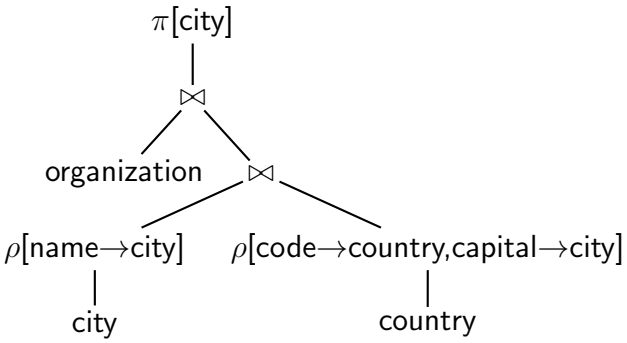
1. ☐



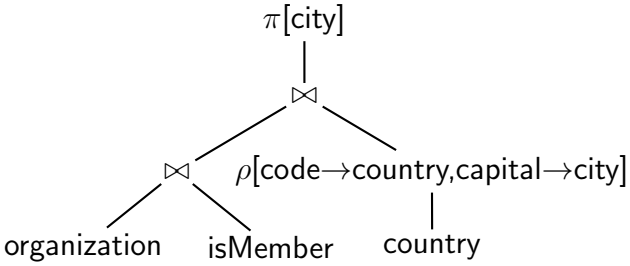
2. ☑


π[city]
⋈
π[abbrev,city,country,province]
⋈
organization
ρ[name→city]
ρ[code→country,capital→city]
π[name,country,province]
country
city

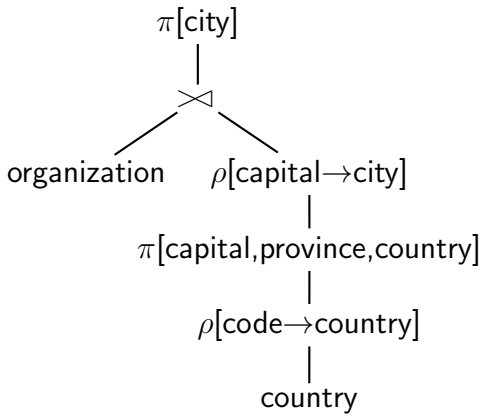
3. ☐


π[city]
⋈
organization
⋈
ρ[name→city]
ρ[code→country,capital→city]
city
country

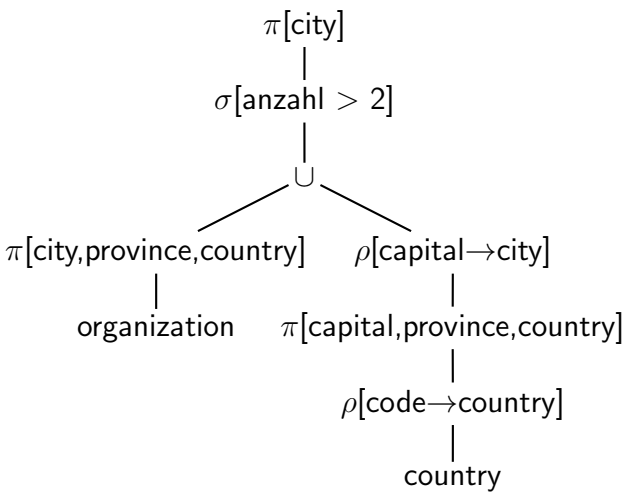
4. ☐


π[city]
⋈
⋈
ρ[code→country,capital→city]
organization
isMember
country

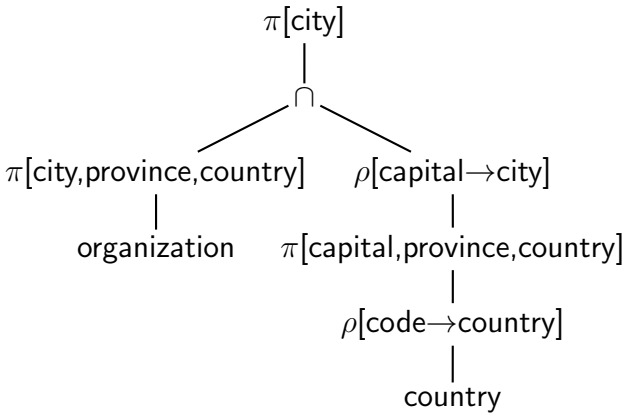
5. ☑


π[city]
⋉
organization
ρ[capital→city]
π[capital,province,country]
ρ[code→country]
country

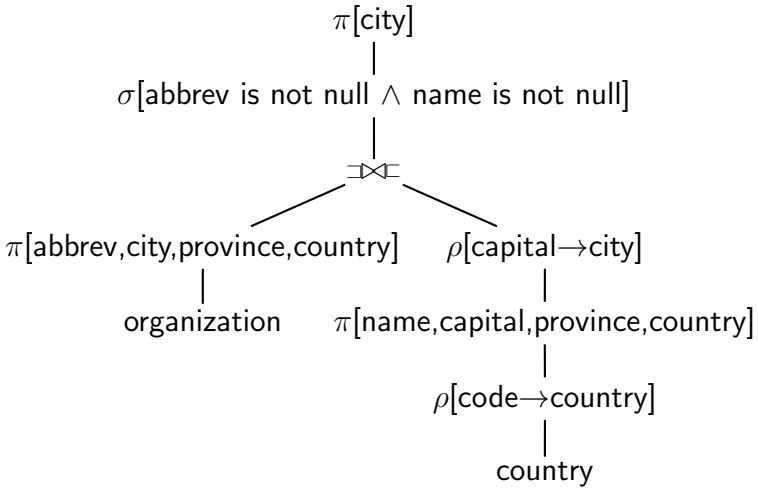
6. $\boxed{-}$



7. $\boxed{\checkmark}$



8. $\boxed{\checkmark}$



Lösung

1. falsch: Das gäbe alle Städte, in denen eine Organisation ihren Sitz hat, die aber keine Hauptstadt sind.

2. richtig: join, wobei die Tabelle city nicht wirklich benötigt wird

3. falsch: join ist OK; city ist unnötig (aber nicht falsch), aber es joint über org.name=country.name und city.pop=country.pop!

4. falsch: joint über org.country(hq)=ismember.country=ismember.country=country.code(→country) , was nicht erwartet wird. Der Sitz darf in einem nicht-Mitgliedsland sein. Und country.name=org.name ist auch noch implizit dabei, was es wirklich richtig falsch macht.

5. richtig: Die Hauptstädte werden im rechten Baum berechnet, scharf auf die benötigten Attribute projiziert, passend renamed, und per Semijoin geschaut, ob es Sitze von Organisationen sind.
   Man könnte das Semijoin auch umdrehen, hätte dasselbe Ergebnis.

6. falsch union+count geht so in der Algebra nicht. Sie hat kein group-by, "anzahl" würde als Spaltenname interpretiert (deutsch könnte sie sowieso nicht). Die Idee ist auch insofern falsch, dass wenn eine Nicht-Hauptstadt Sitz zweier Organisationen wäre, sie auch reinkäme.
   In SQL wäre folgendes korrekt, aber übelster Code:

```
select DISTINCT city
FROM
  ((select DISTINCT city, country, province
    from organization)
   UNION ALL
   (select capital as city, code as country, province
    from country)
GROUP BY city, country, province
HAVING count(*) >= 2
-- UNION ALL: do not remove duplicates
```

7. richtig: Schnittmenge aus headquarters und capitals.
   Im Prinzip das, was die o.g. Semijoin-Lösung auch tut.

8. richtig: outer join, und alle Paare, die eine Organisation und ein Land "haben", ergeben eine Lösung.

Aufgabe 18 (Schemaentwurf/Multiple-Choice [4 Punkte])
Gegeben seien Relationen

```
R(_a1_, a2),
S(_b1_, b2),
T(_c1_, c2), und
X(a,b,c)
```

X ist offensichtlich eine dreistellige Beziehung. Für X seien sowohl A, als auch B, als auch C (einzeln!) Schlüsselkandidaten.
(Überlegen Sie sich, was man daraus schließen kann)
Es gelten die Fremdschlüsselbeziehungen `X.a->R.a1, X.b->S.b1, X.c->T.c1`.
Welche der Aussagen ist/sind zutreffend (offensichtliche sich aus den Namen ergebende Fremdschlüsselbedingungen jeweils nicht explizit angegeben):

☑ Das angegebene Schema ist redundanzfrei (d.h. formal in 3. Normalform).

☑ Das Schema
`R(_a1_, a2, b, c), S(_b1_, b2), T(_c1_, c2)`
ist äquivalent, redundanzfrei und verlustfrei.

☐ Das Schema `R(_a1_, a2, b, c), S(_b1_, b2, a, c), T(_c1_, c2, a, b)`
ist äquivalent, redundanzfrei und verlustfrei.

☐ Das Schema
`R(_a1_, a2), S(_b1_, b2), T(_c1_, c2), X(_a_,b), Y(_b_,c), Z(_c_,a)`
ist äquivalent, redundanzfrei und verlustfrei.

☐ Das Schema
`R(_a1_, a2), S(_b1_, b2), T(_c1_, c2), X(_a_,b), Z(_c_,a)`
ist äquivalent, redundanzfrei und verlustfrei.

☐ Das Schema
`R(_a1_, a2, b), S(_b1_, b2, c), T(_c1_, c2, a, b)`
ist äquivalent, redundanzfrei und verlustfrei.

☐ Es gibt kein anderes Schema, das äquivalent zu dem gegebenen Schema, redundanzfrei und verlustfrei ist.

☑ Es gibt noch weitere Schemata, die äquivalent zu dem gegebenen Schema, redundanzfrei und verlustfrei sind.

Lösung Dass alle drei Attribute Schlüsselattribute sind, bedeutet, dass jedes a, b, c höchstens einer Kombination der beiden anderen Attribute zugeordnet wird (wobei dann beide ungleich *null* sein müssen, da sie ja auch Schlüsselkandidaten sind). Es kann aber auch a_1, b_1 und c_1-Werte in R, S bzw. T geben, die keine solche Zuordnung haben.
(Intutiv: man hat drei Körbe, und kann jeweils a-b-c-Tripel "rausnehmen" und eintüten.)

1. Das Schema ist redundanzfrei: Eine Zerlegung in untergeordnete Zusammenhänge wäre höchstens bei der Tabelle X möglich. Da aber jede Spalte Schlüssel ist, kann es keine Redundanzen geben.
2. Ja. Da a Schlüssel in X ist, kann man die Einträge für b und c mit nach R reinziehen.
3. ist redundant (und damit Fehleintrags-anfällig). Einmal reinziehen genügt.
4. ist redundant/riskant. Man könnte $X(1,2)$, $Y(2,3)$, $Z(3,4)$ bilden und damit Widersprüche erzeugen. (Siehe auch nächste Variante)
5. Auch das nicht: Es ist immerhin widerspruchsfrei: man legt in X ab, welches b zum a gehört, und in Z, welches c zu beiden. Man könnte aber $X(1,2)$ haben, ohne einen Eintrag in Z, der Z dazu, womit es kein c dazu gäbe.
6. Auch das nicht: kombiniert die Probleme aus (4) und (5).
7. Doch, offensichtlich gibt es (mindestens) eines.
8. Ja. Analog zu (2) könnte man die Beziehung auch in *eine* der Tabellen S oder R reinziehen. Man würde sie ggf. in diejenige Tabelle reinziehen, die die wenigsten Tupel enthält. Evtl. gäbe es in einer realen Anwendung eine, aus der alle Elemente zugeordnet werden.

Man kann also die Zordnung (a,b,c) *zusammen* in *eine* der Tabellen reinziehen. Wenn man sie aufspaltet oder mehrmals reinzieht, geht jeweils irgendetwas kaputt.

Aufgabe 19 (Ausdruckskraft - (Umordnungs-Beispiel) [4 Punkte])
Ordnen Sie die folgenden Klassen von Sprachen nach ihrer Ausdruckskraft (stärkste zuerst):

- Konjunktive Anfragen (Selektion/Projektion/Join/Renaming) "SPJR-Algebra"
- Turing-vollständige Programmiersprachen
- Relationale Algebra
- SQL-Anfragen
- nachweisbar terminierende Java-Programme

Lösung Von oben nach unten:

- Turing-vollständige Programmiersprachen: "(fast) alles"
- nachweisbar terminierende Java-Programme: es gibt Einschränkungen gängiger Programmiersprachen, mit denen partielle (=Programm terminiert) und totale (=Programm erfüllt logische Zusicherungen) -meistens mit interaktiven Beweisern- nachgewiesen werden kann.
- SQL-Anfragen: Relationale Algebra plus Datentypen und GROUP BY und Aggregationsoperatoren.
- Relationale Algebra.
- Konjunktive Anfragen (Selektion/Projektion/Join/Renaming) "SPJR-Algebra" – ein wichtiges Fragment der relationalen Algebra aus Sicht der effizienten Auswertbarkeit.
- eine weitere wichtige Sprachfamilie ist Datalog (Vorlesung "Deductive Databases") mit Well-Founded und Stable Semantics. Liegt über der Relationalen Algebra (z.B. ermöglichen schon einfache rekursive Datalog-Regeln die Berechnung der transitiven Hülle; Well-founded und Stable Semantics gehen darüber weit hinaus, Stable Semantics kann z.B. Sudokus lösen). Datalog als Theorie-Sprache liegt ist aber unvergleichbar zu SQL, da es keine Datentypen und kein GROUP BY hat. Implementierungen (z.B. XSB) enthalten das aber auch und liegen damit oberhalb von SQL-Anfragen.

Aufgabe 20 (Transaktionen [6 Punkte])
1) Was bedeutet die Eigenschaft *Durability (Dauerhaftigkeit)* von Transaktionen?

2) Welche Maßnahmen werden in kommerziell eingesetzten Datenbanksystemen eingesetzt, um dies zu gewährleisten?

Lösung

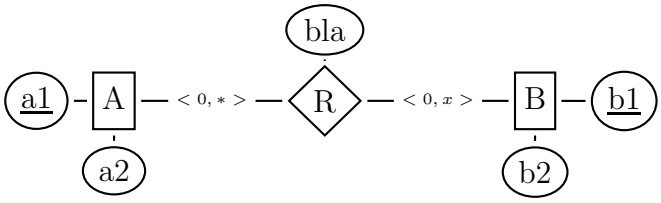
1) Der Effekt einer *erfolgreich* beendeten Transaktion (z.B. die Bestätigung, dass eine Überweisung durchgeführt wurde) ist dauerhaft, d.h., (1) es wird nicht mehr durch ein Rollback oder ähnliches ungeschehen gemacht, und (2) auch im Fall eines Software- oder Hardware-Absturzes kann es nicht mehr verloren gehen.

2) Redundanz (wenn eine der Instanzen crasht, bleibt die andere bestehen), regelmäßige Backup-Kopien (Snapshots sichern), logfiles der ausgeführten Transaktionen und Aktionen (um auf einem Snapshot bzw. dem Absturzzustand aufbauend wieder einen konsistenten Zustand zu erreichen).

Aufgabe 21 (Abb. ER-Modell auf Relationales Modell (a) [2 Punkte])
Nehmen Sie an, dass Sie ein gutes ER-Diagramm entwickelt haben, und dabei sind, dies in ein relationales Modell umzusetzen.
In welchen Situationen benötigt man für einen Entitätstyp mehrere Relationen, um seine Informationen abzulegen?

Lösung Wenn er ein oder mehrere mehrwertige Attribute hat.
(Antworten, die auf die Definitionen der 2. und 3. Normalform eingehen sind auch richtig, allerdings kann man diese Situationen durch ein wirklich gut durchdachtes ER-Modell in den allermeisten Fällen verhindern).

Aufgabe 22 (Abb. ER-Modell auf Relationales Modell (b) [6 Punkte])
Gegeben sei folgendes ER-Diagramm:



Geben Sie das Tabellenschema der Tabelle zur Speicherung von R im relationalen Modell an. (2 P)
Welche Attribute sind (u.a. in Abhängigkeit von x in der Kardinalität von B bzgl. R) Schlüsselattribute dieser Relation? (4 P)

Lösung Die Relation R hat die Attribute a_1, b_1, bla (oder als A, B, bla benannt).
$x = 1$: nur b_1.
Wenn $x > 1$ und jedes B zu jedem A nur einmal in dieser Beziehung stehen kann, bilden a_1 und b_1 den Schlüssel.
Wenn $x > 1$ und ein B zu einem A auch mehrmals mit verschiedenen Werten von bla in Beziehung stehen kann, bilden a_1, b_1 und bla den Schlüssel.
[Bewertung: eine Antwort gab 2P, jede weitere 1.5P; insgesamt konnten also 5P erreicht werden]