Klausur Datenbanken
Wintersemester 2008/2009
Prof. Dr. Wolfgang May
21. Januar 2009, 14-16 Uhr
Bearbeitungszeit: 90 Minuten

Vorname:

Nachname:

Matrikelnummer:

Studiengang:

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.

☐ 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/FlexNever oder beim zuständigen Prüfungsamt.

	Max. Punkte	Schätzung für "4"
Aufgabe 1 (ER-Modell)	20	15
Aufgabe 2 (Transformation ins Relationale Modell)	20	14
Aufgabe 3 (SQL und Relationale Algebra)	35	16
Aufgabe 4 (Indexierung und Auswertung)	15	5
Summe	90	50

Note:

Themenstellung

Alle Klausuraufgaben basieren auf einem gemeinsamen "Auftrag": Entwerfen Sie ein Bibliotheksinformationssystem, das Bestand und Ausleihstatus *aller* Bibliotheken der Universität beinhaltet. Es wird angenommen, dass jeder Fakultät genau eine Bibliothek zugeordnet ist.
Die folgenden Informationen sollen darin abgelegt sein:

- Welche Bücher gibt es in welchen Bibliotheken?
- Welche Person hat welche Buchexemplare ausgeliehen?

Aus den folgenden Hinweisen können Sie ableiten, welche Attribute benötigt werden. (Weitere Hinweise geben auch die zu beantwortenden Anfragen in Aufgabe 3.)

Beispiele für Benutzerausweise:

Name:	Potter
Vorname:	Harry
Benutzer-ID:	13
Statusgruppe:	Student

Name:	Einstein
Vorname:	Albert
Benutzer-ID:	5050
Statusgruppe:	Professor

In der Mathematik-Informatik-Bibliothek stehen mehrere Exemplare des folgenden Buches:

Buchdeckel:

A Guide to the SQL Standard
Christopher Date Hugh Darwen

Addison Wesley Publ.

mit Aufklebern auf dem Buchrücken:

MATHINF
DB123-1

. . .

MATHINF
DB123-18

(die Inventarnummern werden von jeder Bibliothek nach deren eigenem Prinzip vergeben - hier könnte es bedeuten, dass es ein Datenbankbuch ist, von dem (mindestens) 18 Exemplare vorhanden sind). Das gleiche Buch ist auch in der Geographie-Bibliothek vorhanden, wobei es dort den folgenden Aufkleber hat:

GEO
Inf42

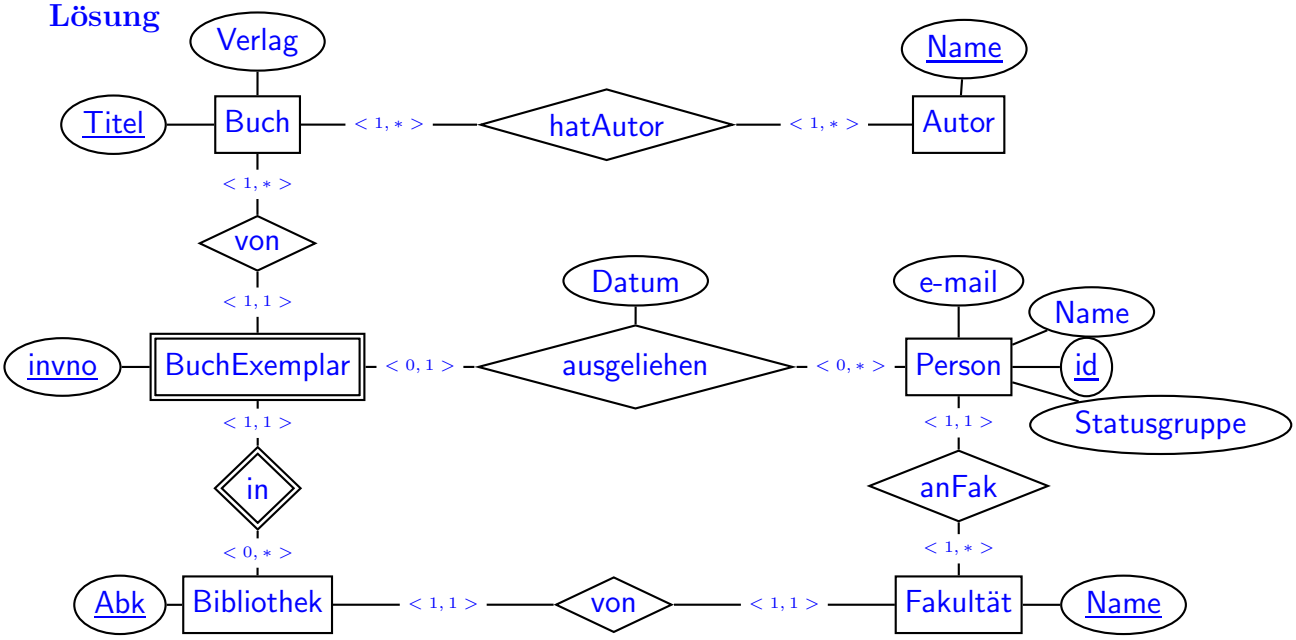
Ausleihzettel (jede Person darf in jeder Bibliothek ausleihen):

Bibliothek: MATHINF
InvNo: DB123-17
Titel: A Guide to the SQL Standard
Ausleiher: 5050
Datum: 15-01-2009

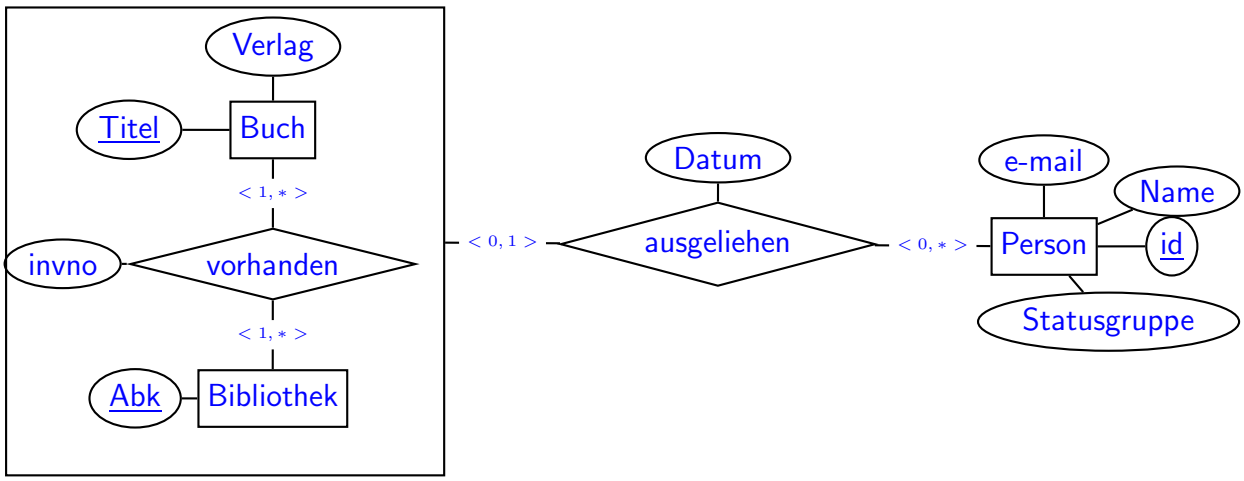
Weiterhin soll zu jeder Person die e-mail-Adresse und an welcher Fakultät sie studiert bzw. arbeitet, abgelegt werden.
Wenn ein Buch von einem Ausleiher zurückgegeben wird, werden die Leihdaten gelöscht.
Weitere Anforderung: Es soll später jederzeit möglich sein, weitere Bibliotheken, Fächer/Fakultäten und Statusgruppen hinzuzufügen.

Aufgabe 1 (ER-Modell [20 Punkte])
Entwickeln Sie ein ER-Modell für das Szenario. Geben Sie darin die Schlüsselattribute sowie die Beziehungskardinalitäten an.

Lösung



- "ausgeliehen" könnte man auf den ersten Blick auch als dreistellige Beziehung zwischen "Person", "BuchExemplar" und "Bibliothek" auffassen. Die Bibliothek ist aber bereits eindeutig durch das BuchExemplar (schwacher Entitätstyp) bestimmt.
- "Bibliothek" und "Fakultät" könnte man auch zusammenfassen (1:1-Entsprechung lt. Voraussetzung), und Attribute "Name" und "Abk" verwenden.
- "Autor" kann auch als mehrwertiges Attribut zu "Buch" modelliert werden.
- anstatt den Entity-Typ "BuchExemplar" kann man auch eine Beziehung "vorhanden" zwischen "Buch" und "Bibliothek" mit Attribut "invno" modellieren. Diese wird dann zu einem Entity-Typ *aggregiert* und nimmt an der Beziehung "ausgeliehen" teil, womit die linke Seite folgendermaßen aussieht:



Die oben genannten Alternativen haben keinen Einfluß auf das Ergebnis der Transformation in das relationale Modell – dieses ist in allen Fällen dasselbe (wenn man es richtig macht)!

Aus dem Aufgabentext ("die Inventarnummern werden von jeder Bibliothek nach deren eigenem Prinzip vergeben") geht ziemlich eindeutig hervor, dass das Paar (invNo,BibAbk) gemeinsam den Schlüssel eines Buchexemplars bilden (so könnte die Inventarnummen "Inf42" z.B. auch noch in der WiWi-Bibliothek existieren).
Lösungen, in denen invNo alleine den Schlüssel bildet, wurden nicht als falsch bewertet (die "Strafe" dafür ist allerdings, dass einige der Anfrage in Aufgabe 2 dadurch auf mehr Tabellen zugreifen müssen).

Aufgabe 2 (Transformation ins Relationale Modell [20 Punkte])

a) **Lösen Sie diesen Aufgabenteil auf dem *letzten* Blatt und trennen dieses ab (und geben es am Ende mit ab!).** Dann haben Sie dieses Blatt separat zugreifbar um später damit die Aufgaben 2b, 3 und 4 (SQL, Relationale Algebra+SQL, interne Auswertung) zu lösen.

Geben Sie an, welche Tabellen (mit Attributen, Schlüsseln etc.) Ihre Datenbank enthält (keine SQL `CREATE TABLE`-Statements, sondern einfach grafisch). (14 P)

Markieren Sie dabei auch Schlüssel (durch unterstreichen) und Fremdschlüssel (durch überstreichen).

Geben Sie die Tabellen mit jeweils mindestens zwei Beispieltupeln (z.B. denen, die sich aus dem Aufgabentext ergeben, und weiteren erfundenen) an.

Lösung

Fakultät	
Name	Abk
Math. und Informatik	MATHINF
Geographie und -logie	GEO
:	:

Person				
Name	id	Statusgr.	Fak	e-mail
Harry Potter	13	Student	PHIL	harry@potter.org
Albert Einstein	5050	Professor	MATHINF	einst@uni-math.goe.de
:	:	:	:	:

Buch	
Titel	Verlag
A Guide to SQL	Addison Wesley Publ.
:	:

hatAutor	
Titel	Autor
A Guide to SQL	C. Date
A Guide to SQL	H. Darwen
:	:

Exemplar		
Titel	Bib	InvNo
A Guide to SQL	MATHINF	DB123-1
A Guide to SQL	MATHINF	DB123-18
A Guide to SQL	GEO	Inf42
Das Göttinger Minischwein	AGR	VIEHZ13
:	:	:

ausgeliehen			
Bib	InvNo	Person	Datum
MATHINF	DB123-17	5050	15-01-2009
AGR	VIEHZ13	815	20-01-2009
:	:	:	:

Alternativ kann man "Exemplar" und "ausgeliehen" zusammenfassen (<0,1>-Kardinalität):

Exemplar				
Titel	Bib	InvNo	Person	Datum
A Guide to SQL	MATHINF	DB123-17	5050	15-01-2009
A Guide to SQL	GEO	Inf42	null	null
Das Göttinger Minischwein	AGR	VIEHZ13	815	20-01-2009
:	:	:	:	:

Anmerkungen:

- bei Person und bei Exemplar jeweils das Kürzel der Fak/Bib als Fremdschlüssel.
- Bei "Exemplar" ist "Titel" kein Schlüsselattribut. Das BuchExemplar ist durch (Bib + Inventarnummer) eindeutig identifiziert.
- Bei "ausgeliehen" ist "Person" ebenfalls kein Schlüssel: jedes Buchexemplar kann an eine Person ausgeliehen sein.

b) Geben Sie das `CREATE TABLE`-Statement für die Tabelle, in der die Ausleihdaten gespeichert sind, so vollständig wie möglich an (6 P).

Lösung

```
CREATE TABLE ausgeliehen                                  Basis: 2.5P
( Person NUMBER NOT NULL REFERENCES Person.id,              1/2 + 1/2
  Bib    VARCHAR2(10),
  Buch   VARCHAR2(10),
  Datum  DATE NOT NULL,                                          1/2
  CONSTRAINT leihe-key PRIMARY KEY (Bib,Buch)                      1
  CONSTRAINT leihe-refs-book
    FOREIGN KEY (Bib,Buch) REFERENCES Exemplar(Bib,invno));      1
```

"NOT NULL" für "Bib" und "Buch" ergibt sich daraus, dass die beiden Attribute den Primary Key bilden (es ist aber nicht falsch, es explizit nochmal anzugeben).

Aufgabe 3 (SQL und Relationale Algebra [35 Punkte])
Verwenden Sie für diese Aufgabe die von Ihnen entworfene relationale Datenbasis. Keine der Antworten soll Duplikate enthalten.

a) Geben Sie **eine SQL-Anfrage und einen Algebra-Ausdruck** an, die folgendes berechnen:
Welche Bücher, an denen *Hugh Darwen* mitgearbeitet hat, hat die Universität, und zu welchen Bibliotheken gehören sie? (2+2 P)

Lösung

```
select distinct titel, bib
from Exemplar, hatAutor
where Exemplar.titel = hatAutor.Titel
  and hatAutor.name = 'Hugh Darwen';
```

π[titel,bib](Exemplar $\bowtie$ σ[name="Hugh Darwen"](hatAutor))

Es genügt auch ein Semijoin:
π[titel,bib](Exemplar $\ltimes$ σ[name="Hugh Darwen"](hatAutor))

b) Geben Sie **eine SQL-Anfrage** an, die angibt, wieviele Exemplare von "A Guide to the SQL Standard" es an dieser Uni gibt. (2 P)

Lösung

```
select count(*)
from Exemplar
where Titel='A Guide to the SQL Standard'
```

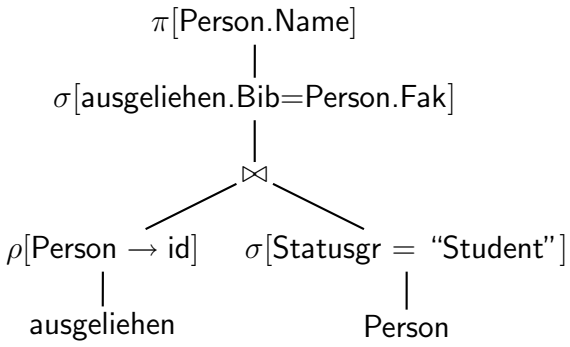
c) Geben Sie **eine SQL-Anfrage oder einen Algebra-Ausdruck** an, die/der folgendes berechnet:
Die Namen derjenigen Studenten, die ein Buch aus der Bibliothek der Fakultät an der sie studieren, ausgeliehen haben. (3 P)

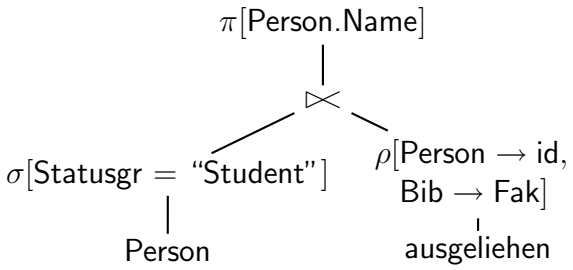
Lösung

```
select distinct Person.Name
from Person, ausgeliehen
where Person.Statusgr = "Student"
  and Person.id = ausgeliehen.Person
  and ausgeliehen.Bib = Person.Fak

select Person.Name
from Person
where Person.Statusgr = "Student"
  and exists
   (select *
```

```
from ausgeliehen
where ausgeliehen.Person = Person.id
  and ausgeliehen.Bib = Person.Fak)
```

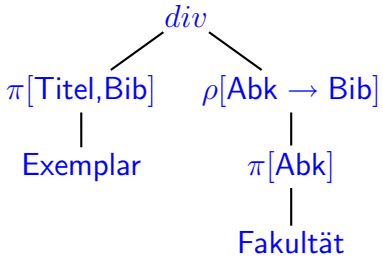




d) Geben Sie **eine SQL-Anfrage oder einen Algebra-Ausdruck** an, die folgendes berechnen:
Welche Buchtitel sind in jeder Bibliothek vorhanden? (3 P)

Lösung

```
select title from buch
where not exists
  (select *
   from Fakultaet
   where not exists
     (select * from Exemplar
      where Bib = Fakultaet.Abk
        and Titel = Buch.Titel))
```



Man kann es in diesem Fall auch mit einer COUNT-basierten Anfrage machen (diese sind allerdings im allgemeinen weniger effizient, und in der Entwicklung fehleranfällig wenn man nur mit einer durch eine Subquery bestimmten Teilmenge vergleichen will):

```
select title from Buch
where
 (select count(distinct Bib)     %% in wievielen Bibs steht dieses Buch?
  from exemplar
  where exemplar.Titel = Buch.Titel)
 =
 (select count(*)          %% sind das alle?
  from Fakultaet)
```

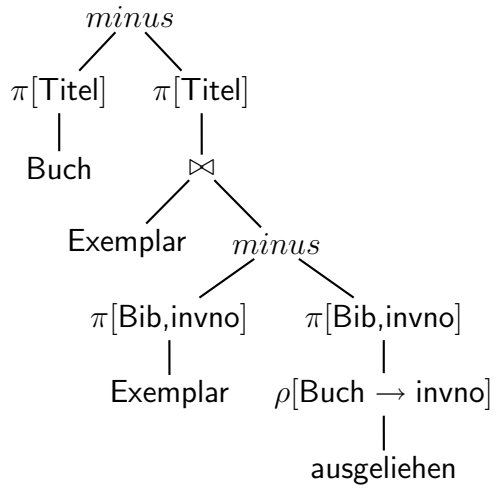
e) Geben Sie **eine SQL-Anfrage oder einen Algebra-Ausdruck** an, die/der alle Buchtitel ausgibt, von denen momentan alle Exemplare verliehen sind. (5 P)

Lösung

```
select titel
from Buch b
where not exists
((select bib, invno
  from Exemplar
  where titel = b.titel)
 minus
 (select bib, Buch as invno
  from ausgeliehen
  where titel = b.titel))

-- man kann es auch abzaehlen: sind soviele Exemplare verliehen, wie es gibt
select titel
from Buch b
where (select count(*)
       from Exemplar e
       where e.Titel = b.Titel)
      =
      (select count(*)
       from Exemplar e, ausgeliehen a
       where e.Titel = b.Titel
         and e.Bib = a.Bib
         and e.invno = a.Buch)

select titel
from Exemplar e
group by titel
having count(*)   -- Eintraege (=invnos) dieser Gruppe/dieses Titels
   = (select count(*)   -- ausgeliehene
      from ausgeliehen, Exemplar
      where ausgeliehen.invno = Exemplar.invno
        and ausgeliegen.bib = Exemplar.bib
        and Exemplar.titel = e.titel);
```



Mit einer einfachen Division geht es hier nicht, da man eine *korrelierte* Division benötigt, deren "rechte" Seite von der linken abhängig ist (nämlich dasselbe Buch betrifft).

f) Geben Sie **eine SQL-Anfrage und einen Algebra-Ausdruck** an, die die Namen aller Paare von (Student, Professor) ausgeben, die an derselben Fakultät studieren bzw. lehren, und denselben Buchtitel ausgeliehen haben. (4+4 P).

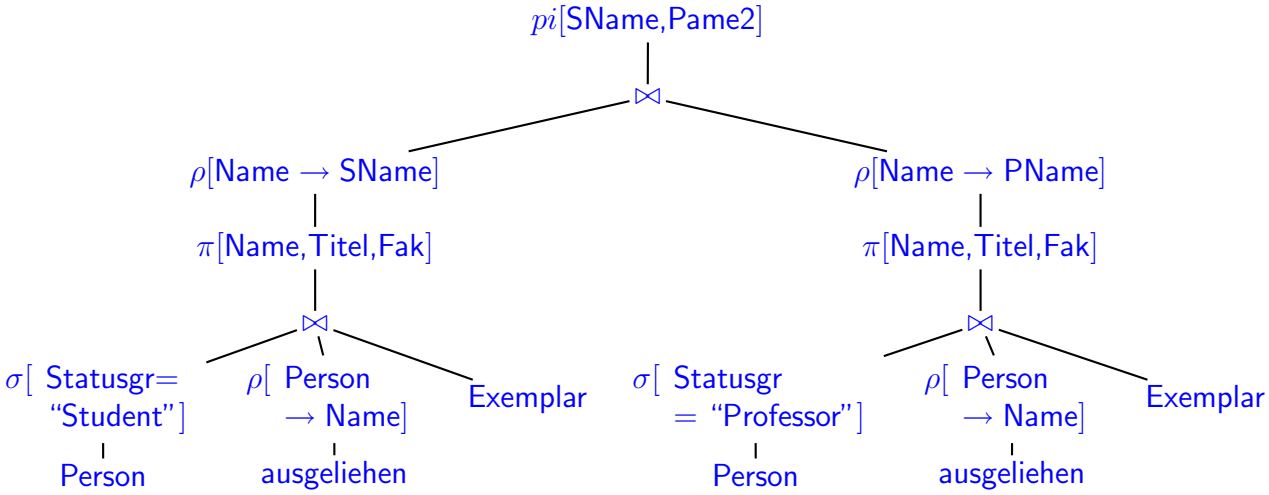
Lösung

```
select distinct p1.name, p2.name
from person p1, ausgeliehen a1, exemplar e1,
     person p2, ausgeliehen a2, exemplar e2
where p1.Statusgruppe = "Student"
  and p1.id = ausgeliehen1.Person
  and a1.Buch = e1.invno and a1.Bib = e1.invno
  and e1.Titel = e2.Titel
  and a2.Buch = e2.invno and a2.Bib = e2.invno
  and p2.id = ausgeliehen2.Person
  and p2.Statusgruppe = "Professor"
  and p1.Fakultaet = p2.Fakultaet

aehnliche Loesungen mit

select p1.name, p2.name
from person p1, person p2
where (p1.id, p2.id) in (select ... from ... where ...)

select p1.name, p2.name
from person p1, person p2
where exists (select ... from ... where ...)
```



g) Geben Sie **eine SQL-Anfrage** an, die den Titel desjenigen Buches angibt, von dem momentan am meisten Exemplare verliehen sind (bzw. mehrere Titel, falls es mehrere solche Bücher gibt)? (5 P)

Lösung

```
select titel
from ausgeliehen
group by titel
having count(*)
= (select max(count(*))
   from ausgeliehen, Exemplar
   where ausgeliehen.bib = Exemplar.bib
     and ausgeliehen.Buch = Exemplar.invno
   group by titel)
```

h) Ein bisschen Theorie: Gegeben sei eine Relation R mit Format (=Attributnamen) $R(A, B, C, D)$ sowie eine Relation $S(B, C)$.

R enthalte 64.000 Tupel. S sei leer.

Welche Aussagen können Sie über die Ergebnisrelation T des Ausdrucks R *div* S machen (*div*: relationale Division)? Beweisen oder begründen Sie Ihr Ergebnis. (5 P)

Lösung T hat das Format $T(A, D)$ [richtiges Format: 1P] und enthält die Tupel $R = \pi[A, D](R)$.

Formaler Beweis:

$R \div S := \{\mu \in \mathsf{Tup}(A, D) \mid \{\mu\} \times S \subseteq R\}$.

Da S leer ist, ist $\{\mu\} \times S = \emptyset$, was wiederum trivialerweise eine Teilmenge von R ist.

Intuitive Begründung: alle (a, d)-Wertepaare, die man in $\pi[A, D](R)$ hat, und die zusammen mit *jedem* der (b, c)-Paare aus S kombiniert, also als $(a, b, c, d) \in R$ sind. Da S leer ist, ist letztere Anforderung trivial, d.h., für jedes Paar (a, d) erfüllt.

(der übliche Fall eines Allquantors über eine leere Menge: die Aussage "alle Einhörner sind blau" ist logisch wahr, da es keine Einhörner gibt).

Das Ergebnis enthält nicht unbedingt 64.000 Tupel, da die Duplikate bei der Projektion $\pi(A, D)$ entfernt werden.

Aufgabe 4 (Indexierung und Auswertung [15 Punkte])
Gegeben ist wieder das Szenario aus den Aufgaben 1-3. An alle *Studierenden*, die Bücher aus der Bibliothek der *Geographie* ausgeliehen haben, soll eine e-Mail verschickt werden. Geben Sie eine SQL- oder Algebra-Anfrage an, die die benötigten e-Mail-Adressen ausgibt. Beschreiben Sie eine sinnvolle Möglichkeit, wie diese Anfrage intern ausgewertet werden kann, und welche Indexe die Datenbank dafür enthalten sollte.
Gehen Sie bei Ihren Größenabschätzungen von folgenden Daten aus:

- 30.000 Benutzer, davon 24.000 Studenten,
- derzeit 12 Bibliotheken,
- 12.000.000 Buchexemplare,
- davon sind 360.000 Buchexemplare ausgeliehen,
- Eine Speicherseite umfasst 4096 Bytes,
- Nehmen Sie an, dass im Hauptspeichercache Indexseiten behalten werden, während normale Tabellen-Speicherungs-Seiten wenn man sie nicht iterativ durchgeht, gleich wieder ausgelagert werden.

Machen Sie bzgl. Tupelgröße plausible Annahmen basierend auf Ihren jeweiligen Tabellenschemata. Begründen Sie *kurz* und nachvollziehbar, wie Sie auf Ihre Zahlen kommen. Nehmen Sie bezüglich Werteverteilung Gleichverteilung an. Geben Sie die Anzahl der Seitenzugriffe (grobe Überschlagsrechnungen) an.

Lösung

```
select email
from person
where statusgruppe = "Student"
  and id in (select person
             from ausgeliehen
             where bib = "GEO")

select distinct email
from person, ausgeliehen
where person.statusgruppe = "Student"
  and person.id = ausgeliehen.Person
  and ausgeliehen.bib = "GEO";
```

Vorgehen: man sollte sich *erst* überlegen, wie man auswerten will, und dann die geeigneten Indexe und Berechnungen betrachten.
Auswertung: Grundsätzlich hat man zwei offensichtliche Möglichkeiten, wie man vorgehen kann:

- Alle Studierenden in der Tabelle "Person" betrachten (Iteration, oder Index – da die meisten Benutzer Studierende sind, bringt hier ein Index wenig), per Index auf ausgeliehen.Person zugreifen, und schauen, ob man unter den Einträgen zu dieser Person ein Tupel findet, das auf die "GEO"-Bibliothek verweist.

  Iteration über Personen: jedes Person-Tupel umfasst einen Name (40 Zeichen/80Bytes in UTF-8), ID (4 Bytes), 10 Bytes (ASCII) Statusgruppe, 10 Bytes Fakultät, 40 Bytes e-mail (zusammen 150 Bytes). Also passen ca. 25 Einträge auf jede Seite. Es werden

also für 30000 Personen 1200 Seiten benötigt. Diese geht man linear durch (1200 Seitenzugriffe und 30000 Cache-interne Tests) und findet 24000 Studierende. Für diese will man nun möglichst effizient prüfen, was sie ausgeliehen haben.

B*-Baum-Index auf "ausgeliehen.Person": 360000 id-Einträge (NUMBER, 4 Bytes) zu indexieren. Jeder Blatt-Eintrag ist ein Paar (ID, referenz) und benötigt (Ref auf Tupel ebenfalls mit 4 Bytes annehmen) 8 Bytes, Blätter durchschnittlich zu etwa 75% gefüllt, sind 375, nehmen wir 400 Einträge pro Blatt. Also 900 Blätter.

Innere Knoten: ref-id-ref-id-...-ref-Struktur, also kann man wieder ca. 400 Einträge pro Blatt annehmen. Damit sind hat man 3 innere Knoten auf unterer Ebene, und noch einen als Wurzel. Insgesamt 904 Indexseiten, die passen in den Hauptspeichercache. (man sieht, dass es jetzt keine wirkliche Rolle spielt, ob man 350, 375, oder 400 ansetzt).

Auswertung also: Index in den Cache holen, 900 Hintergrundspeicher-Zugriffe. Für jeden der 24.000 Studierenden einmal den Index durchlaufen ($24.000 \cdot 3 = 72.000$ interne Vergleiche). Man findet durchschnittlich 12 Referenzen auf "ausgeliehen"-Tupel. Man muss die jeweilige Seite in den Cache holen, auf das Tupel zugreifen und dieses auf "GEO" testen: $12 \cdot 24.000 = 288.000$ Seitenzugriffe und auch 288.000 Cache-interne Vergleiche.

Summe: 1200 + 900 + 288.000 Seitenzugriffe.

Hierbei kann man die weitere Suche für einen Studenten abbrechen, wenn man für einen Studenten einen "GEO"-Eintrag gefunden hat. Das spart z.B. für Geo-Studenten einige Zugriffe. Man kann annehmen dass bei $1/12$ der Studierenden (=Geo) der Abbruch schnell erfolgt, und man somit statt 288.000 nur 265000 Vergleiche wirklich machen muss.

Stattdessen mit Hash-Index: Jeder Eintrag auf einer Hash-Kachel ist ebenfalls ein Paar (ID, referenz) und benötigt wie oben 8 Bytes, und man hat insgesamt ca. 900 Kachelseiten. Damit ist prinzipiell alles wie oben, nur hat man statt 72.000 Vergleiche im Index nur 24000 Anwendungen der Hash-Funktion.

- Oder, man läuft über alle Tupel in "ausgeliehen", prüft sie auf "GEO" "ausgeliehen.Bib='GEO"' (dieses Attribut muss es dort geben, da es Teil des Schlüssels des Buches ist) und greift für alle erhaltenen Werte ausgeliehen.Person dieser Tupel über den Index Person.id auf die Person zu und gibt den gefundenen Namen aus. Hierbei muss man dann aber noch die Duplikate entfernen!

  Beim Zugriff auf "ausgeliehen.Bib" bringt ein Index wenig, da $1/12$ aller Einträge GEO sind, und man sowieso alle Seiten lesen muss (es kann sogar kontraproduktiv sein, wenn man über den Index durcheinander zugreift).

  Iteration über "ausgeliehen": jedes "ausgeliehen"-Tupel umfasst 10 Bytes Fakultät, 10 Bytes Inventarcode, Person-ID (4 Bytes), Datum (Oracle: 7 Bytes), also ca. 30 Bytes. Man hat ca. 133 Einträge pro Seite, also werden für 360.000 Einträge ca. 2700 Seiten benötigt. Man iteriert über alle Seiten (2700 Seitenzugriffe, 360.000 Vergleiche) und findet ca. $360.000/12 = 30.000$ Geo-Ausleihungen.

  Index auf "Person.id": Blätter (oder Hash-Kacheln) enthalten wie oben Paare (id,referenz), also auch wieder 400 Einträge pro Blatt. Bei 30.000 Personen also ca. 75 Blätter und noch eine Wurzel. Ein Hash-Index hätte genauso ca. 75 Kachelseiten.

  Für jede der 30.000 Geo-Ausleihungen greift man auf den Index und das referenzierte Person-Tupel zu, und schaut ob es ein Student ist. Da man Duplikate entfernen muss, kann man die e-mail-Adresse nicht sofort ausgeben, sondern muss auf das DISTINCT

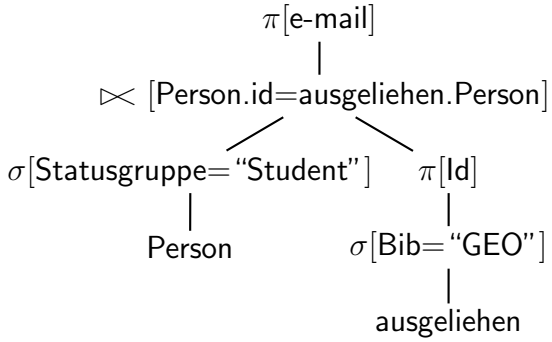
warten. Die erhaltenen Ergebnisse werden daher intern zwischengespeichert (in einem Baum, um gleich beim Einfügen die Duplikate wegzulassen). Dieser Baum ist prinzipiell eine Indexstruktur, bleibt also auch im Cache.

Eine plausible Größenordnung wären ca. 5000 Ergebnisse (die 3000 Geo-Studierenden und einige weitere), also ca. 200.000 Bytes, was in ca. 50 Blattknoten (und einer Wurzel) resultiert.

Summe: 2700 + 75 + 30.000 Seitenzugriffe.

Wie man feststellt, greift man bei beiden Möglichkeiten über den Index "durcheinander" zu, was mehr Seitenzugriffe erfordert, als notwendig. Z.B. benötigt man im 2. Fall fur jede der 30.000 Ausleihungen einen Seitenzugriff, ob die ausleihende Person ein Student ist.

- Eine dritte Möglichkeit entspricht dem Algebra-Baum



In einem ersten(linearen) Durchgang durch alle Seiten von "ausgeliehen" (2700 Seitenzugriffe, s.o.) werden alle IDs der Personen, die in "GEO" etwas ausgeliehen haben, (in einem B-Baum als Zwischenergebnis) gespeichert. Dies sind 30.000 Geo-Ausleihungen. Wenn man den "Worst-Case" annimmt wären es 30.000 verschiedene IDs, d.h. es hätte jede der 30.000 Personen ein Geo-Buch ausgeliehen (in Wirklichkeit sind es erheblich weniger).

Dennoch, für 30.000 IDs à 4 Bytes hätte der B-Baum (der keine Referenzen, sondern nur die Werte enthält) 10 Blätter (3000 Einträge pro Blatt bei 75% Füllgrad) und eine Wurzel, also nur 11 Seiten; kann im Cache gehalten werden.

Dann wird linear einmal über "Person" (1200 Seiten) iteriert, und für jeden Eintrag getestet ob Statusgruppe="Student" ist, und, falls ja, alle Studierenden geschaut, ob die ID in dem Baum enthalten ist. Hierbei sind wie eben gesehen keine weiteren Seitenzugriffe notwendig. Da man die Personen nacheinander durchgeht, müssen auch keine Duplikate eliminiert werden – jedes Ergebnis kann direkt ausgegeben werden.

Anzahl Speicherzugriffe: 2700 + 1200 = 39000.

[Trennen Sie dieses Blatt am besten vor Beginn der Bearbeitung ab]
Lösen Sie hier Aufgabe 2a