Datenbanken
Wintersemester 11/12
Prof. Dr. W. May

2. Übungsblatt: Algebra

Besprechung voraussichtlich am 18.11.2009

Aufgabe 1 (Division mit Basisoperationen) Beweisen Sie, daß die in der Vorlesung angebene Darstellung der Division durch relationale Basisoperatoren als

$$r \div s = \pi[Z](r) - \pi[Z]((\pi[Z](r) \bowtie s) - r)$$

mit $r(X)$, $s(Y)$ und $Z = X \setminus Y$ äquivalent zu der gegebenen Definition

$$r \div s = \{\mu \in \mathsf{Tup}(Z) \mid \{\mu\} \bowtie s \subseteq r\}$$

ist.

Veranschaulichen Sie sich Ihre Überlegungen anhand des Beispiels "Geben Sie die Namen derjenigen Organisationen an, die auf jedem Kontinent mindestens ein Mitglied haben".

Nach Definition der Division ist

$$r \div s = \{\mu \in \mathsf{Tup}(Z) \mid \{\mu\} \bowtie s \subseteq r\} \quad (*)$$

Betrachte

$$r \div s = \pi[Z](r) - \underbrace{\pi[Z]((\pi[Z](r) \bowtie s) - r)}_{(***)} \quad (**)$$

Im Beispiel ist r die Menge aller Paare $(org, cont)$ so dass Organisation o ein Mitglied auf Kontinent $cont$ hat. s ist die Menge der (Namen der) Kontinente. $r \div s$ ist das gesuchte Ergebnis.

Im allgemeinen Fall (z.B. "... auf allen Kontinenten mit mehr als 10.000.000 km^2 Fläche") kann r auch Tupel mit Y-Werten (Kontinenten) enthalten, die in $\{\mu\} \bowtie s$ nicht gefordert werden. Die Division prüft nur, ob die geforderten Kombinationen vorhanden sind.

Vorwärtsbeweis

(1) $(**) \subseteq (*)$: Korrektheit des Ausdrucks. Alle Ergebnisse erfüllen die Charakterisierung:
Sei $\mu \in (**)$.
μ ist also in $\pi[Z](r)$, also $\in \mathsf{Tup}(Z)$ – soweit nicht besonders interessant.
Interessanter ist der zweite Teil, der aussagt, dass μ nicht in $\pi[Z]((\pi[Z](r) \bowtie s) - r)$ ist. μ ist also kein Z-Tupel (im Beispiel: keine Organisation), so dass bei $(\{\mu\} \bowtie s) - r$ ein Rest übrigbleibt. D.h., $\{\mu\} \bowtie s \subseteq r$, und damit ist μ ein Tupel, das $(*)$ erfüllt.

(2) $(*) \subseteq (**)$: Vollständigkeit der Charakterisierung – Ergebnis enthält alle geforderten Werte:
Sei μ ein Tupel über Z (Beispiel: eine Organisation), $\mu \in (*)$, also $\{\mu\} \bowtie s \subseteq r$.
Damit offensichtlich $\mu \in \pi[Z](r)$; jetzt also noch zu zeigen, dass μ nicht in $\mu \notin (***)$ ist.
Aus $\{\mu\} \bowtie s \subseteq r$ folgt, dass $(\{\mu\} \bowtie s) - r$ leer ist. Damit ist auch μ nicht in $\pi[Z]((t \bowtie s) - r$, für beliebige Mengen $t \subseteq \mathsf{Tup}Z$, also nicht in $(***)$.

Hinweis: Man könnte anstelle $(**)$ auch

$$r \div s = \pi[Z](r) - \underbrace{\pi[Z]((\mathsf{Tup}(Z) \bowtie s) - r)}_{(***)} \quad (*)$$

schreiben. Damit müsste man allerdings eine unendliche Menge von Werten überprüfen. Durch die Einschränkung auf $\pi[Z](r)$ erhält man eine endliche Ausgangsmenge. (Im Beispiel entspräche $\mathsf{Tup}(Z)$ der Menge aller möglichen Zeichenketten als Namen von Organisationen, die nicht mit jedem Kontinent in r auftreten – natürlich kommen dabei als Antworten nur solche in Frage, die in $\pi[Z](r)$ überhaupt enthalten sind).

Alternative: Widerspruchsbeweis (verwendet im Prinzip natürlich dieselbe Argumentation):

(1) $(**) \subseteq (*)$: Korrektheit des Ausdrucks. Alle Ergebnisse erfüllen die Charakterisierung:
Sei $\mu \in (**)$.
Annahme: $\mu \notin (*)$, d.h. $\{\mu\} \bowtie s \not\subseteq r$.
Dann gibt es also ein $\nu \in s$, so dass $\mu\nu \notin r$.
$\mu \in \pi[Z](r)$ nach Konstruktion von $(**)$
Schauen wir uns wieder $(***)$ an: Da $\mu \in \pi[Z](r)$ ist, ist $\mu\nu \in \pi[Z](r) \bowtie s$ und ja $\notin r$, also $\mu\nu \in (\pi[Z](r) \bowtie s) - r$ und damit $\mu = \pi[Z](diesem) = (***)$.
Womit μ nicht in $(*)$ ist. Widerspruch zur Voraussetzung.

(2) $(*) \subseteq (**)$: Vollständigkeit der Charakterisierung – Ergebnis enthält alle geforderten Werte:
Sei $\mu \in (*)$, also $\{\mu\} \bowtie s \subseteq r$. Für alle $\nu \in s$ ist also $\mu\nu \in r$.
Damit also erstmal $\mu \in \pi[Z](r)$.
Zu zeigen: $\mu \notin (***)$.
Annahme: $\mu \in (***)$ – also gäbe es ein ν', so dass $\mu\nu' \in (\pi[Z](r) \bowtie s) - r$ ist. Der Anteil $\pi[Z](\mu\nu')$ ist gerade μ, also müsste $\nu' \in s$ sein, damit dies erfüllt ist, und $\mu\nu' \notin r$ – im Widerspruch zu oben.
Also $\mu \in (**)$.

Aufgabe 2 (Äquivalenzen: Join, Division, Differenz) Seien $R(X), S(Y)$ Relations-Schemata. Zeigen oder widerlegen Sie:

(a) Sei $X \cap Y = \emptyset$.

$$(R \bowtie S) \div S \equiv R.$$

(b) Sei $X = Y$ und $Z \subseteq X$.

$$\pi[Z](R - S) \equiv \pi[Z]R - \pi[Z]S.$$

(zu a) Nach Voraussetzung sind $R(X)$ und $S(Y)$ Relationen, die kein gemeinsames Attribut haben. Damit gilt $(X \cup Y) - Y = X$.

Behauptung: dann gilt: $(R \bowtie S) \div S = R$

Definition der Division: Sei $Y \subset X$, $Z = X - Y$.

$$r \div s = \{\mu \in \mathsf{Tup}(Z) \mid \{\mu\} \times s \subseteq r\} = \pi[Z](r) - \pi[Z]((\pi[Z](r) \times s) - r).$$

$$
\begin{aligned}
&(R \bowtie S) \div S && \text{keine gemeinsamen Attribute}\\
&= (R \times S) \div S && \text{Definition der Division}\\
&= \pi[X](R \times S) - \pi[X]((\pi[X](R \times S) \times S) - R \times S)\\
&= R - \pi[X]((R \times S) - (R \times S))\\
&= R - \pi[X](\emptyset)\\
&= R
\end{aligned}
$$

(zu b) $\pi[Z](R - S) \neq \pi[Z](R) - \pi[Z](S)$

Bsp:

Sei $X = Y = \{A, B\}$ und $Z \subseteq X = \{A\}$

R	
A	B
1	2

S	
A	B
1	3

$\pi[A](R - S) = \pi[A](R) = \{(A : 1)\}$ (da $R - S = R$ ist), und $\pi[A](R) - \pi[A](S) = \{(A : 1)\} - \{(A : 1)\} = \emptyset$

Aufgabe 3 (Korrektheit der Join-Algorithmen) Die Definition des relationalen Joins-Operators, $\bowtie$, auf den Folien ist nicht konstruktiv, sondern beschreibt "nur" *deklarativ*, aus welchen Tupeln das Ergebnis bestehen soll.

Beweisen Sie, dass die beiden folgenden Join-Algorithmen (a) und (b) korrekt sind, d.h. diese Menge von Tupeln liefern.

Gegeben seien Relationen $R(\bar{X})$ und $S(\bar{Y})$; berechnet werden soll $R \bowtie Y$.

a) Nested-Loop-Join:

```
let Z̄ := X̄ ∩ Ȳ;
let T := ∅;
for r ∈ R do
    begin
      for s ∈ S do
        if r[Z̄] = s[Z̄] then
          begin
            μ := ein neues Tupel so dass μ(x) = r(x) für alle x ∈ X̄ und μ(y) = s(y) für alle y ∈ Ȳ \ X̄;
            T := T ∪ {μ};
          end;
    end;
return T;
```

b) Loop-Index-Join: Hierbei ist ein Baumindex über den Join-Attributen $\bar{Z} := \bar{X} \cap \bar{Y}$ von S in der Datenbank vorhanden.
Nehmen Sie an, dass $\bar{Z}$ Schlüssel von S ist (dann erstellt ein Datenbanksystem diesen Index automatisch). Nehmen Sie weiter an, dass $\bar{Z}$ nur ein Attribut enthält, welches numerisch ist, und stellen Sie sich als Index einen Binärbaum wie (hoffentlich) in Info I besprochen vor. In den jeweiligen Baumknoten ist nicht nur der Wert, sondern auch eine Referenz auf das Tupel, das diesen Wert hat, enthalten.

```
let Z̄ := X̄ ∩ Ȳ;
let B := der Baumindex über S.Z̄;
let T := ∅;
for r ∈ R do
    begin
      if Suche nach r[Z̄] in B erfolgreich
        begin
          s = das vom Baumknoten referenzierte Tupel in S;
          μ := ein neues Tupel so dass μ(x) = r(x) für alle x ∈ X̄ und μ(y) = s(y) für alle y ∈ Ȳ \ X̄;
          T := T ∪ {μ};
          end;
    end;
return T;
```

c) Wie ist die Komplexität der beiden Algorithmen?

---

a) Nested Loop. Die äußere Schleife iteriert über alle Tupel $r \in R$, in der inneren Schleife wird jedes Tupel $s \in S$ getestet, ob es "passt" (d.h. $r[\bar{Z}] = s[\bar{Z}]$). Wenn ja, wird das kombinierte Tupel rs zum Ergebnis dazugenommen.

R enthält m Tupel, S enthält n Tupel, geschachtelte Schleife, also Komplexität $O(n \cdot m)$.

Korrektheitsbeweis: Zu zeigen ist

$$\{\mu \in \mathrm{Tup}(\overline{XY}) : \text{Ergebnis des Algorithmus enthält } \mu\} = \{\mu \in \mathrm{Tup}(\overline{XY}) \mid \mu[\bar{X}] \in R \text{ und } \mu[\bar{Y}] \in S\} =: R \bowtie S$$

Zu zeigen sind zwei Richtungen: "$\subseteq$" (Korrektheit: alle ausgegebenen Ergebnisse sind richtig), und "$\supseteq$" (Vollständigkeit: der Algo findet alle Ergebnisse). Üblicherweise ist die Korrektheit einfacher zu zeigen (da sie die Entwicklung des Algorithmus widerspiegelt), als die Vollständigkeit (diese ist aber wichtiger, da man evtl. etwas übersehen hat).

"$\subseteq$" Wenn μ ein Ergebnistupel ist, dann gibt es Tupel $r \in R$ und $s \in S$ so dass $r[\bar{Z}] = s[\bar{Z}]$ ist (die in dem "if"-Statement an diese Variablen gebundenen Tupel), nach denen μ so erzeugt wird, dass $\mu(x) = r(x)$ für alle $x \in \bar{X}$ und $\mu(y) = s(y)$ für alle $y \in \bar{Y} \setminus \bar{X}$, also $\mu = rs$ gilt. Offensichtlich ist dann genau $r = \mu[\bar{X}]$ und $s = \mu[\bar{Y}]$, und damit $\mu \in R \bowtie S$.

"$\supseteq$" Wenn $\mu \in R \bowtie S$, ist also $r' := \mu[\bar{X}] \in R$ und $s := \mu[\bar{Y}] \in S$. Die Variable r der äußeren Schleife enthält irgendwann dieses Tupel r', und im inneren Durchlauf enthält die Variable s irgendwann das Tupel s'. Da $Z = \bar{X} \cup \bar{Y}$ ist $r'[\bar{Z}] = s'[\bar{Z}]$, womit die "if"-Bedingung zu "wahr" ausgewertet wird, und das entsprechende Ergebnistupel erzeugt wird.

b) Loop-Index-Join. Die äußere Schleife iteriert über alle Tupel $r \in R$. Für jedes r wird über den Index auf alle Tupel $s \in S$ zugegriffen, für die $s[\bar{Z}] = r[\bar{Z}]$) ist, und das kombinierte Tupel rs zum Ergebnis dazugenommen.

Komplexität: Komplexität: n Durchläufe der äußeren Schleife. Der Index-Lookup benötigt $O(\log n)$ (für ausgeglichene Binärbäume $O(\log_2 n)$, für die in Datenbanken üblichen B- bzw. B*-Bäume (siehe späteres Kapitel der Vorlesung) z.B. $O(\log_{200} n)$).

Hat man exakt ein Tupel pro Indexeintrag zu verarbeiten, ist die Komplexität somit $O(n \log m)$.

Sind es mehrere (keine bis viele) Tupel (beim allgemeinen Join, wenn $\bar{Z}$ nicht Schlüssel von S ist) ist immerhin garantiert, dass pro Zugriff ein Ergebnis geliefert wird. Wenn man die Anzahl der Ergebnisse nicht (anhand Wissens über die Anwendung) abschätzen kann, kann man nur $O(\max(n \log m, \#\text{Ergebnisse}))$ als Komplexitätsabschätzung angeben.

Beweis: Der Beweis funktioniert wie oben:

"$\subseteq$" wie oben. Man könnte auch einfach argumentieren, dass das Ergebnis eine Teilmenge dessen aus (a) ist.

"$\supseteq$" Diese Richtung ist etwas interessanter. Wenn $\mu \in R \bowtie S$, ist also $r' := \mu[\bar{X}] \in R$ und $s := \mu[\bar{Y}] \in S$. Die Variable r der äußeren Schleife enthält irgendwann dieses Tupel r'. Da $Z = \bar{X} \cup \bar{Y}$ ist, ist $s'[\bar{Z}] = r'[\bar{Z}]$, womit der Indexeintrag zu diesem Wert $r'[\bar{Z}]$ auf s' verweisen muss (im Prinzip reduziert man das Problem auf die Vollständigkeit des Indexes). Damit wird für r' und s' das entsprechende Ergebnistupel erzeugt.

c) siehe (a) und (b).

Aufgabe 4 (Algebra: Minimale- und Maximale Anzahl von Tupeln) Die Relationen $R(\bar{X})$ und $S(\bar{Y})$ enthalten n bzw. m Tupel. Wie groß ist die maximale und minimale Anzahl von Tupeln, die das Ergebnis folgender Operationen (bei geeigneten $\bar{X}, \bar{Y}$) enthalten kann?

a) $R \cup S$

b) $R \bowtie S$

c) $\sigma[C](R) \times S$, für eine Bedingung C

d) $\pi[Y](R) - S$

e) $R \div S$

---

a) Nur zulässig wenn $\bar{X} = \bar{Y}$.
   max: $n + m$ (keine gemeinsamen Tupel)
   min: $max\{n, m\}$ (Teilmengenbeziehung)

b) max: $n * m$, wenn entweder $\bar{X} \cap \bar{Y} = \emptyset$ oder im natürlichen Join jedes Tupel R mit jedem Tupel aus S auf den gemeinsamen Attributen $\bar{X} \cap \bar{Y}$ zusammenpasst (d.h., diese Attribute haben in allen Tupeln beider Relationen den selben Wert).
   min: 0 (natürliches Join mit $\bar{X} \cap \bar{Y} \supsetneq \emptyset$, bei dem aber nichts übrigbleibt, da die Einträge der gemeinsamen Spalte(n) nicht zusammenpassen.
   Hinweis: man sieht, dass ein Join also die Ergebnismenge sowohl deutlich vergrößern, als auch einschränkend wirken kann.
   - Verhältnis zwischen $|R \bowtie S|$ und $|R|$ bezeichnet man als *Selektivität* des Joins (bzgl. R).
   - ist $\bar{X} \cap \bar{Y}$ z.B. Schlüssel von S und Fremdschlüssel in R, so hat man $|R \bowtie S| = |R|$.

c) max: $n * m$ (alle Tupel in R erfüllen C)
   min: 0 (kein Tupel in R erfüllt C)

d) Nur sinnvoll wenn $\bar{Y} \subseteq \bar{X}$.
   max: n, wenn $S = \emptyset$, oder zumindest $S \cap \pi[Y](R) = \emptyset$, und bei der Projektion keine Duplikate (die wegfallen würden) entstehen.
   min: 0, wenn $\pi(Y)(R) \subseteq S$.

e) max: n/m, wenn alle Werte $\pi[\bar{X} \setminus \bar{Y}](R)$ mit jedem der Werte aus S in R vorkommen.
   min: 0, wenn kein Wert die durch die Division bzgl. S gestellte "für alle"-Bedingung erfüllt.

Aufgabe 5 (Äquivalenz von Ausdrücken) Gegeben seien folgende Relationen:

- R(A,B,C)
- S(A,E,F)
- T(A,H)

Die Wertebereiche aller nicht namensgleichen Attribute seien voneinander verschieden. Gegeben sei nun folgender relationaler Ausdruck:

$$\pi[E,H](\sigma[B=10]((R \bowtie T) \bowtie S))$$

Sind die folgenden Ausdrücke äquivalent zu obigem Ausdruck? Begründen Sie Ihre Antwort.

a) $\pi[E,H]((\sigma[B=10](R)) \bowtie (\pi[A,E](S)) \bowtie T)$

b) $\pi[E,H](\sigma[B=10](((\pi[B](R)) \bowtie (\pi[A,E](S))) \bowtie (\pi[A,H](T))))$

c) $\pi[E,H]((\pi[A,B](\sigma[B=10](R))) \bowtie ((\pi[A](S)) \bowtie T))$

---

a) ist äquivalent:
Der ursprüngliche Ausdruck:

$$\pi[E,H](\sigma[B=10]((R \bowtie T) \bowtie S))$$

Join ist assoziativ:

$$\pi[E,H](\sigma[B=10](R \bowtie (T \bowtie S)))$$

Attribut B existiert nur in Relation R, daher kann Selektion vor Join ausgeführt werden:

$$\pi[E,H](\sigma[B=10]R \bowtie (T \bowtie S))$$

Von S werden nur die Attribute A (im Join mit T) und E (in der abschließenden Projektion) benötigt, man kann die Projektion also auch gleich auf S ausführen:

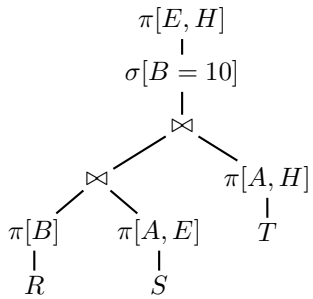
$$\pi[E,H](\sigma[B=10]R \bowtie (T \bowtie (\pi[A,E]S)))$$

Join ist auch kommutativ:

$$\pi[E,H](\sigma[B=10]R \bowtie ((\pi[A,E]S) \bowtie T))$$

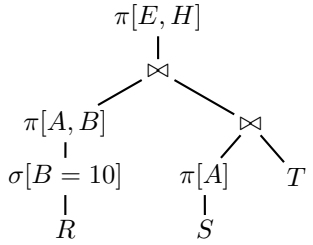
Der letzte Ausdruck zeigt dass die Äquivalenz gilt.
Hinweis: in diesem Ausdruck werden alle π und σ so früh wie möglich ausgeführt.

b) nicht äquivalent:



$\pi[B](R)$ eliminiert das für den Join mit S und T wichtige Attribut A, hier wird stattdessen das Kreuzprodukt ausgeführt.
$\pi[A,E](S)$ ist wie in a) gezeigt zulässig – F wird nicht gebraucht.
$\pi[A,H](T)$ ist nutzlos (Identität), stört aber keinen.
Der Ausdruck ist syntaktisch in Ordnung, hat aber ein anderes Ergebnis (eine Obermenge, da ein Join-Kriterium wegfällt).

c) nicht äquivalent:



$\pi[A](S)$ eliminiert das Attribut E, welches in der Ausgabe enthalten sein soll. der Ausdruck ist damit syntaktisch nicht zulässig.

Aufgabe 6 (Relationale Anfragen an Mondial: Bedingungen) Geben Sie Ausdrücke der relationalen Algebra für die folgenden Anfragen an die Mondial-Datenbank an:

a) Die Namen aller Städte, die mehr als 1.000.000 Einwohner haben.

b) Die Namen aller Städte, die mehr Einwohner als Neuseeland haben.

c) Die Namen aller Städte, in denen mehr als 25% der Bevölkerung des jeweiligen Landes leben.

Für spätere Übungsblätter:

- Geben Sie dieselben Anfragen in SQL an.
- Geben Sie dieselben Anfragen im relationalen Kalkül an.

---

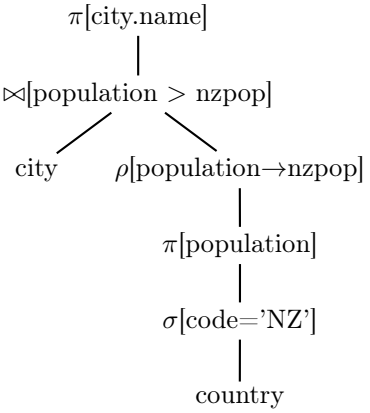
a) Einfache Selektion:

$$\pi[\text{name}](\sigma[\text{population} > 1000000](\text{city}))$$

b) Die Bevölkerungszahl von Neuseeland bekommt man als

$$\pi[\text{population}](\sigma[\text{code='NZ'}](\text{country}) \ .$$

In der Selektionsbedingung in Teil (1) ist aber nur eine Konstante erlaubt. Man benötigt ein Join, in dem man jedes Land mit dem Ergebnis der Subquery joint und den Vergleich durchführt:



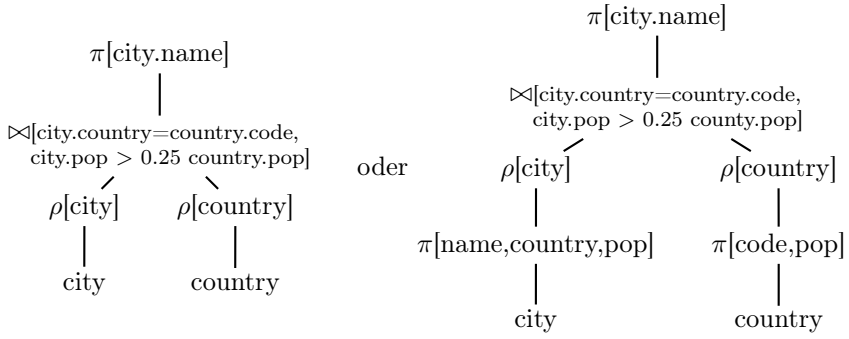
Hinweis: in SQL kann man die Subquery in die WHERE-Klausel einbauen:

```
SELECT name
FROM city
WHERE population > (SELECT population FROM country WHERE code='NZ')
```

oder als Join formulieren:

```
SELECT name
FROM city,  (SELECT population as nzpop FROM country WHERE code='NZ')
WHERE population > nzpop
```

c) Korreliertes Join mit Zusatzbedingung: Sei allgemein $\rho[alias]$ die Umbenennung, die alle Attribute mit dem Präfix *alias* qualifiziert (z.B. für city.name).



Aufgabe 7 (Relationale Anfragen an Mondial: Organisationen) Geben Sie Ausdrücke der relationalen Algebra für die folgenden Anfragen an die Mondial-Datenbank an:

a) Die Namen aller Organisationen, die auf jedem Kontinent mindestens ein Mitgliedsland haben.

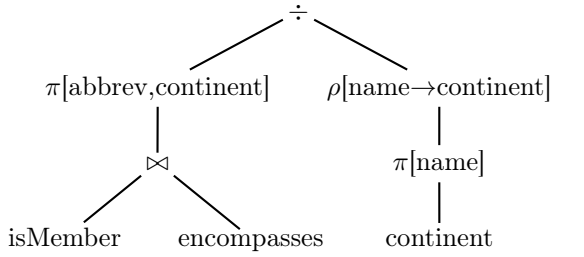
b) Die Namen aller Organisationen, in denen alle Staaten mit mehr als 50.000.000 Einwohnern Mitglied (unabhängig von der Art der Mitgliedschaft) sind.

Für spätere Übungsblätter:

- Geben Sie dieselben Anfragen in SQL an.
- Geben Sie dieselben Anfragen im relationalen Kalkül an.

---

a) Division:



isMember(org,country,type) und encompasses(country,continent,percent) werden über country gejoint. Davon wird nur (organisation,continent) (*) benötigt. Das wird dann durch die Menge der Namen der Kontinente geteilt, womit alle Organisationen übrig bleiben, die in (*) mit jedem Kontinent genannt werden.

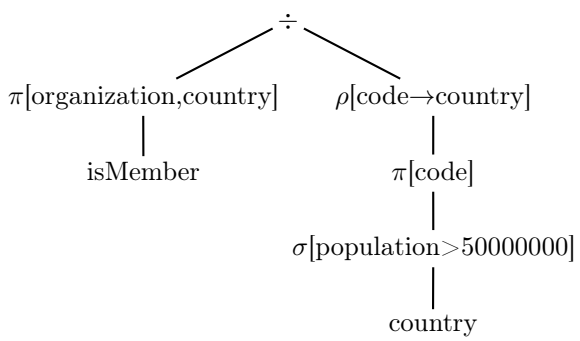
```
select abbreviation from organization o
where not exists
(select *
 from continent c
 where not exists (select *
                   from ismember i, encompasses e
                   where i.country = e.country
                     and i.organization = o.abbreviation
                     and e.continent = c.name ))
```

Hinweis: man sieht in SQL immer wieder Lösungen, die die Division auf die Mächtigkeit der Menge zurückführen; hier würde man einfach zählen, welche Organisation auf sovielen Kontinenten wie es gibt, vertreten ist. In diesem Fall wäre das richtig. Die gilt jedoch nicht für Teil (2) der Aufgabe:

```
select abbreviation from organization o
where
(select count(distinct continent)
 from ismember i, encompasses e
 where i.country = e.country
   and i.organization = o.abbreviation
 group by i.organization)
```

```
      = (select count (*) from continent)
```

b)




```
select name from organization o
where not exists
(select *
 from country
 where population > 50000000
   and code not in (select country
                    from ismember
                    where organization = o.abbreviation))
```

Hier würde ein Zählen der Mitgliedsländer nichts bringen, da es viele Länder gibt, die die zusätzliche Bedingung > 1000000 nicht erfüllen. Man könnte allenfalls die Mitgliedsländer zählen, die mehr als 1000000 Einwohner haben - womit die Anfrage aber auch nicht einfacher würde.

Aufgabe 8 (Relationale Anfragen an Mondial: Schweizer Sprachen) Geben Sie Ausdrücke der relationalen Algebra für die folgenden Anfragen an die Mondial-Datenbank an:

a) Alle Landescodes von Ländern, in denen eine Sprache gesprochen wird, die auch in der Schweiz gesprochen wird.

b) Alle Landescodes von Ländern, in denen ausschliesslich Sprachen gesprochen werden, die in der Schweiz nicht gesprochen werden.

c) Alle Landescodes von Ländern, in denen nur Sprachen gesprochen werden, die auch in der Schweiz gesprochen werden.

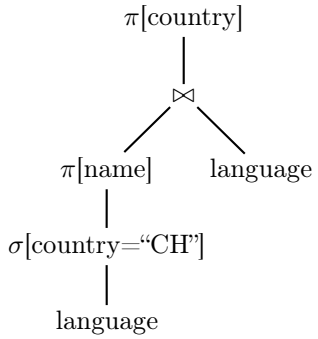
d) Alle Landescodes von Ländern, in denen alle Sprachen gesprochen werden, die in der Schweiz gesprochen werden.

Für spätere Übungsblätter:

- Geben Sie dieselben Anfragen in SQL an.
- Geben Sie dieselben Anfragen im relationalen Kalkül an.

Sei allgemein $\rho[alias]$ die Umbenennung, die alle Attribute mit dem Präfix *alias* qualifiziert (z.B. für city.name).

a) Join zur Formulierung einer "Auswahlbedingung". Verwende Tabelle language(country,name,percent), z.B. (CH, "german", 65).

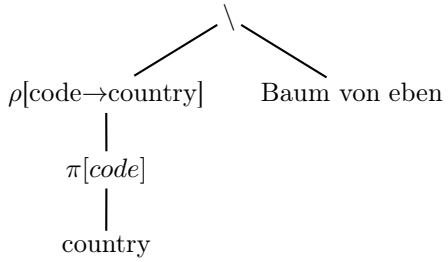


Der linke Ast ergibt die Tabelle

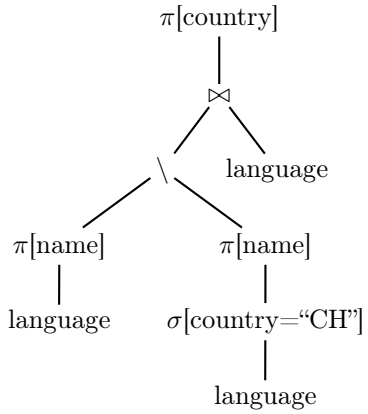
language
name
german
french
italian
romansch

Diese wird mit language gejoint (Join-Attribut "name"), womit genau die Einträge aus language übrigbleiben, die eine der aufgeführten Sprachen enthalten.

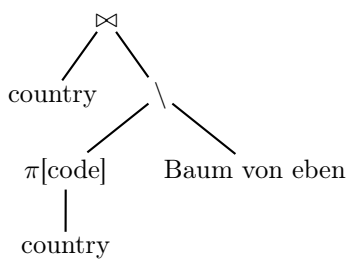
b) Genau alle Länder, die eben nicht aufgezählt wurden:



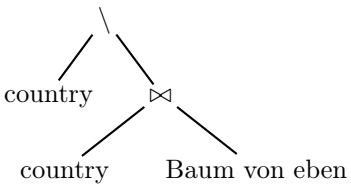
c) Löst man in mehreren Schritten. Erst alle Sprachen bestimmen, die nicht in CH gesprochen werden. Das Ergebnis wird mit language gejoint, und übrig bleiben die Einträge, die nicht-schweizer Sprachen beschreiben. Die Länder (bzw. Landescodes), in denen irgendeine nicht-schweizer Sprache gesprochen wird, bekommt man also folgendermassen:



Bleibt nun, die Komplementmenge, eben die Länder in denen nur schweizer Sprachen gesprochen werden, zu bestimmen:

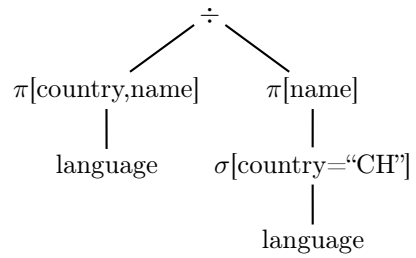


oder äquivalent



... schönes Beispiel dafür, wie man Subqueries in der WHERE-Klausel in der Algebra durch Joins realisiert.

d) "alle" – also mal wieder Division. $\pi[country, name](language)$ durch die Menge der schweizer Sprachen dividieren. Übrig bleiben die Länder die mit allen diesen Sprachen genannt werden:



Aufgabe 9 (Transitive Hülle) Gegeben sei eine Relation R(A,B). Skizzieren Sie einen Algorithmus, der, bestehend aus Operationen der relationalen Algebra und einer while-Schleife, die transitive Hülle der Relation R berechnet.

Hinweis: Die transitive Hülle einer Relation R, bezeichnet als R^*, ergibt sich wie folgt: betrachte z.B. eine Relation $R(von, nach)$ von Flugverbindungen. R^2 ist dann die Menge aller Verbindungen, die über eine Zwischenlandung zustandekommen, etc; R^n sind also diejenigen, Verbindungen, die sich aus n Teilverbindungen zusammensetzen. Die unendliche Vereinigung $R \cup R^2 \cup R^3 \cup ...$ für $R \to \infty$ wird dann als R^* bezeichnet. In einer endlichen Datenbasis benötigt man nur endlich viele Schritte um diese zu berechnen. Ein anderes beliebtes Beispiel ist die aus $Kind(x, y)$ berechnete Vorfahren-Relation.

Problem der Algebra und SQL: jedes R^n kann ausgedrückt werden - aber die beliebig große Vereinigung nicht. Man weiß nicht, wo man abbrechen soll.

Über $R(A, B)$ soll die transitive Hülle gebildet werden. T und S sind binäre Relationen über A und B.

$S := \emptyset$
$T := R$
while $T - S \neq \emptyset$ do
  begin
    $S := T$;
    $T := T \cup \pi[T.A, R.B](\sigma[T.B = R.A](T \times R))$;
  end.

In diesem Programm enthält S jeweils den Wert von T aus der letzten Iteration der Schleife. Die Berechnung endet, wenn S und T übereinstimmen, d.h. während der zuletzt ausgeführten Iteration wurden keine weiteren Tupel mehr zu T hinzugefügt.

```
% ?- river(N,R,L,S,A,B,C,D,E,F,G).
% ?- river(N,R,L,S,A,B,C,D,E,F,G), S \= null.
% ?- lake(N,A,B,C,D,R,E,F), R \= null.
toSea(N,S) :- river(N,_R,_L,S,_,_,_,_,_,_,_), S \= null.
toSea(N,S) :- river(N,R,_L,_S,_,_,_,_,_,_,_), R \= null, toSea(R,S).
toSeaLake(N,S) :- lake(N,_,_,_,_,R,_,_), R \= null, toSea(R,S).
toSea(N,S) :- river(N,_R,L,_,_,_,_,_,_,_,_), L \= null, toSeaLake(L,S).
```

Aufgabe 10 (Relationale Vollständigkeit) Was versteht man unter *Relationaler Vollständigkeit*? Wenn eine Sprache relational vollständig ist, ist es dann möglich jede beliebige Anfrage in dieser Sprache zu formulieren?

Relationale Vollständigkeit bedeutet, dass eine Sprache so ausdruckskräftig wie die relationale Algebra ist, d.h., dass man alle Probleme, die man im relationalen Modell codieren und in der Algebra lösen kann, auch in der gegebene Sprache codieren und lösen kann.

Nein, z.B. können relational vollständige Sprachen Probleme wie die "transitive Hülle" nicht notwendigerweise lösen (Hinweis: es gibt natürlich relational vollständige Sprachen, die das können – jede Turing-vollständige Sprache ist auch relational vollständig).