Einführung in Datenbanken

Übung 11: Relationale Algebra, Outer Join in SQL

Prof. Dr. Stefan Brass
PD Dr. Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Wintersemester 2020/21

http://www.informatik.uni-halle.de/~brass/db20/

Inhalt

1. Hausaufgabe 9
2. Präsenzaufgabe 10
3. Relationale Algebra
4. Joins in SQL
5. Präsenzaufgabe 11

Hausaufgabe 9a (1)

- Geben Sie Name und Vorname von allen Komponisten aus, von denen es mindestens 5 Stücke in der Datenbank gibt.
- Geben Sie die Anzahl der Stücke mit aus, nennen Sie die Spalte `anz_stuecke` und sortieren Sie die Ausgabe danach absteigend (größte Anzahl zuerst).

  Stücke mit unbekanntem Komponisten (d.h. mit Nullwert in der Spalte `knr`) können Sie natürlich ignorieren. Sie können davon ausgehen, dass Name und Vorname zusammen einen Alternativschlüssel der Komponisten-Tabelle bilden.

name	vorname	anz_stuecke
Händel	Georg Friedrich	25
⋮	⋮	⋮
Beethoven	Ludwig van	5

10 Datensätze

Hausaufgabe 9a (2)

- KOMPONIST(KNR, NAME, VORNAME, GEBOREN, GESTORBEN°)
- STUECK(SNR, KNR→KOMPONIST, TITEL, TONART°, OPUS°)
- Lösung (Typisches Muster für Klausur-Aufgaben):

```
SELECT k.name, k.vorname, COUNT(*) anz_stuecke
FROM   komponist k, stueck s
WHERE  k.knr = s.knr
GROUP  BY k.knr, k.name, k.vorname
HAVING COUNT(*) >= 5
ORDER  BY anz_stuecke DESC
```

Stücke mit unbekanntem Komponisten werden durch den Join eliminiert.

Hausaufgabe 9b (1)

- Was ist die minimale, maximale und durchschnittliche Anzahl Stücke pro CD?

  Sie müssen also die Anzahl Stücke pro `cdnr` bestimmen, und jeweils durch `anz_cds` teilen (z.B. haben Doppel-CDs nur eine `cdnr`), und anschließend den Durchschnitt über diesen Anzahlen bilden. Runden Sie die Zahlen auf eine Nachkommestelle (geht mit der Funktion `ROUND` mit der Anzahl Nachkommastellen als zweitem Argument). Nennen Sie die Spalten „`Min`“, „`Max`“ und „`Durchschnitt`“ — auch mit dieser Groß-/Kleinschreibung.

- Die erwartete Antwort ist:

Min	Max	Durchschnitt
0.5	23.0	3.8

1 Datensatz

Hausaufgabe 9b (2)

- CD(CDNR, NAME, HERSTELLER, ANZ_CDS, GESAMTSPIELZEIT)
- AUFNAHME(CDNR→CD, SNR→STUECK, ORCHESTER°, LEITUNG°)
- Dies ist ein Beispiel für eine geschachtelte Aggregation:

```
WITH   stuecke_pro_cd AS
       (SELECT a.cdnr,
               COUNT(*)/c.anz_cds AS anz_stuecke
        FROM   aufnahme a, cd c
        WHERE  a.cdnr = c.cdnr
        GROUP  BY a.cdnr, c.anz_cds)
SELECT ROUND(MIN(anz_stuecke),1) AS "Min",
       ROUND(MAX(anz_stuecke),1) AS "Max",
       ROUND(AVG(anz_stuecke),1) AS "Durchschnitt"
FROM   stuecke_pro_cd
```

Hausaufgabe 9c (1)

- Geben Sie für jeden Komponisten, von dem mindestens drei Stücke mit bekannter Tonart (nicht Null) in der Datenbank stehen, aus,
  - wie viele Stücke es sind, und
  - was der Prozentsatz in Dur-Tonarten und
  - was der Prozentsatz in Moll-Tonarten ist.

  Den Komponisten identifizieren Sie durch Name und Vorname und ordnen die Ausgabe nach dem Namen, bei gleichem Namen nach dem Vornamen. Dur-Tonarten sind solche, die in „`-dur`“ enden, und Moll-Tonarten solche, die in „`-moll`“ enden. Bei dieser Aufgabe sollen nur Stücke mit definierter Tonart betrachtet werden (auch in den Prozent-Berechnungen). Geben Sie die Prozente gerundet ohne Nachkommastellen aus (nutzen Sie `ROUND`, und zwar die Variante mit nur einem Parameter). Denken Sie an mögliche Probleme mit der Integer-Division!

Hausaufgabe 9c (2)

- Die erwartete Antwort ist:

name	vorname	anz	prz_dur	prz_moll
Bach	Johann Sebastian	10	70	30
Händel	Georg Friedrich	21	62	38
Mozart	Leopold	6	100	0
Mozart	Wolfgang Amadeus	5	80	20
Prokofiev	Serge	4	75	25
Schubert	Franz	3	67	33
Telemann	Georg Philipp	15	87	13
Vivaldi	Antonio	8	50	50
Wolf-Ferrari	Ermanno	4	100	0

9 Datensätze

Die beiden letzten Spalten sollen eigentlich `prozent_dur` und `prozent_moll` heißen.

Hausaufgabe 9c (3)

```
SELECT k.name, k.vorname, COUNT(*) anz,
       ROUND(SUM(CASE WHEN s.tonart LIKE '%-dur'
                      THEN 1 ELSE 0 END
                     ) * 100.0 / COUNT(*)
             ) AS prozent_dur,
       ROUND(SUM(CASE WHEN s.tonart LIKE '%-moll'
                      THEN 1 ELSE 0 END
                     ) * 100.0 / COUNT(*)
             ) AS prozent_moll
FROM   komponist k, stueck s
WHERE  k.knr = s.knr AND s.tonart IS NOT NULL
GROUP  BY k.knr, k.name, k.vorname
HAVING COUNT(*) >= 3
ORDER  BY k.name, k.vorname
```

Hausaufgabe 9c (4)

- Statt `SUM(CASE ... THEN 1 ELSE 0 END)` geht auch:

```
COUNT(CASE WHEN s.tonart LIKE '%-dur'
THEN s.snr ELSE NULL END)
```

  Statt `s.tonart` kann man irgendetwas schreiben, was nicht `NULL` ist, z.B. auch `1`.

- Man beachte, dass PostgreSQL die Integer-Division verwendet, wenn beide Argumente von einem Integer-Typ sind.
- Die Aggregationsfunktion `COUNT` liefert den Typ `bigint`.
- Z.B. funktioniert Folgendes nicht (liefert meist 0, sonst 100):

```
(COUNT(CASE ... END) / COUNT(*)) * 100
```

  Wenn jemand z.B. 4 Stücke in einer Dur-Tonart hat, und 5 Stücke insgesamt, berechnet PostgreSQL 4/5 als 0. Mal 100 bleibt 0.

Hausaufgabe 9c (5)

```
SELECT k.name, k.vorname, COUNT(*) anz,
       ROUND((SELECT COUNT(*) FROM stueck dur
              WHERE  dur.knr = k.knr
              AND    dur.tonart like '%-dur'
             )*100.0/COUNT(*)) AS prozent_dur,
       ROUND((SELECT COUNT(*) FROM stueck moll
              WHERE  moll.knr = k.knr
              AND    moll.tonart LIKE '%-moll'
             )*100.0/COUNT(*)) AS prozent_moll
FROM   komponist k, stueck s
WHERE  k.knr = s.knr and s.tonart is not null
GROUP  BY k.knr, k.name, k.vorname
HAVING COUNT(*) >= 3
ORDER  BY k.name, k.vorname
```

Hausaufgabe 9c (6)

```
WITH
gesamt AS (SELECT knr, count(*) AS anz
           FROM   stueck
           WHERE  TONART IS NOT NULL
           GROUP  BY knr),
dur AS     (SELECT knr, count(*) AS anz
           FROM   stueck
           WHERE  TONART LIKE '%-dur'
           GROUP  BY knr),
moll AS    (SELECT knr, count(*) AS anz
           FROM   stueck
           WHERE  TONART LIKE '%-moll'
           GROUP  BY knr)
SELECT     -- siehe nächste Folie
```

Hausaufgabe 9c (7)

```
SELECT k.name, k.vorname, g.anz,
       ROUND(COALESCE(d.anz,0) * 100.0 / g.anz)
            AS prozent_dur,
       ROUND(COALESCE(m.anz,0) * 100.0 / g.anz)
            AS prozent_moll
FROM   komponist k
       JOIN gesamt g    ON k.knr = g.knr
       LEFT JOIN dur d  ON k.knr = d.knr
       LEFT JOIN moll m ON k.knr = m.knr
WHERE  g.anz >= 3
ORDER  BY k.name, k.vorname
```

Der äußere Verbund („`LEFT JOIN`“) ist nötig, weil es auch Komponisten gibt, die keine Moll-Stücke haben (und es auch welche ohne Dur-Stücke geben könnte). Wenn ein Komponist in der Hilfstabelle `moll` nicht auftaucht, würde er bei einem normalen Verbund eliminiert. Der äußere Verbund wird unten noch geübt.

Hausaufgabe 9c (8)

- Wenn man den Outer Join vermeiden will, muss man bei Komponisten ohne Moll-Stücke einen 0-Eintrag erzeugen:

```
moll AS (SELECT knr, COUNT(*) AS anz
         FROM   stueck
         WHERE  tonart LIKE '%-moll'
         GROUP  BY knr
         UNION  ALL
         SELECT knr, 0 AS anz
         FROM   komponist k
         WHERE  NOT EXISTS
                (SELECT *
                 FROM   stueck s
                 WHERE  s.knr = k.knr
                 AND    s.tonart LIKE '%-moll'))
```

Hausaufgabe 9c (9)

- Auch mit einer Unteranfrage unter SELECT kann man die Zahl 0 für Komponisten ohne Moll-Stück erzeugen:

```
moll AS (SELECT k.knr,
                (SELECT COUNT(*)
                 FROM   stueck s
                 WHERE  s.knr = k.knr
                 AND    s.tonart LIKE '%-moll')
                 AS anz
         FROM   komponist k)
```

Obwohl im Beispielzustand zufällig alle Komponisten, bei denen es überhaupt Stücke mit definierter Tonart haben, auch mindestens ein Dur-Stück haben, kann man sich darauf natürlich nicht verlassen. D.h. auch bei den Dur-Anzahlen muss man wie hier dafür sorgen, dass ggf. auch die Anzahl 0 erzeugt wird.

Hausaufgabe 9c (10)

- War diese Hausaufgabe zu schwer?
  - A. Viel zu schwer. Frustrierend.
  - B. Herausfordernd, ich bin daran gewachsen.
  - C. Ok.
  - D. War doch einfach.
- Sollte es mehr als drei Aufgaben geben (die meisten natürlich eher einfach)?
  - A. Ja. So wie es ist, fehlt mir Übungsmaterial.
  - B. Ja, das wäre hilfreich.
  - C. Es ist ok wie es ist.
  - D. Auf keinen Fall.

Inhalt

1 Hausaufgabe 9

2 Präsenzaufgabe 10

3 Relationale Algebra

4 Joins in SQL

5 Präsenzaufgabe 11

Präsenzaufgabe: Erste RA-Anfragen (1)

Schreiben Sie folgende Anfragen in relationaler Algebra:

- Geben Sie alle Bewertungen für Hausaufgabe 1 aus, bei denen weniger als 10 Punkte erzielt wurden.

  Drucken Sie die komplette Zeilen der Tabelle `BEWERTUNGEN`.

- Geben Sie die E-Mail-Adresse von Lisa Weiss aus.

  Schreiben Sie die Anfragen in ASCII, so dass RelaX sie versteht.
  [http://dbis-uibk.github.io/relax/calc/gist/8dc2652578ee12ae756a234c4cf21b3f]

Dies bezieht sich auf das bekannte Schema:

- STUDENTEN(SID, VORNAME, NACHNAME, EMAIL^o)
- AUFGABEN(ATYP, ANR, THEMA, MAXPT)
- BEWERTUNGEN(SID$\rightarrow$STUDENTEN, (ATYP, ANR)$\rightarrow$AUFGABEN, PUNKTE)

Präsenzaufgabe: Erste RA-Anfragen (2)

- Geben Sie alle Bewertungen für Hausaufgabe 1 aus, bei denen weniger als 10 Punkte erzielt wurden.

  Drucken Sie die komplette Zeilen der Tabelle `BEWERTUNGEN`.

- Lösung 1:

  $\sigma_{\texttt{PUNKTE} < 10 \ \wedge \ \texttt{ATYP} = \texttt{'H'} \ \wedge \ \texttt{ANR} = 1}(\texttt{BEWERTUNGEN})$

  Dies ist schon die erweiterte Selektion (mit $\wedge$, $\vee$, $\neg$). Bei der Basisoperation sind nur atomare Formeln erlaubt (wie in Lösung 2).

- Lösung 2:

  $\sigma_{\texttt{PUNKTE} < 10}(\sigma_{\texttt{ATYP} = \texttt{'H'}}(\sigma_{\texttt{ANR} = 1}(\texttt{BEWERTUNGEN})))$

- Zum Vergleich:

```
SELECT *
FROM   BEWERTUNGEN
WHERE  PUNKTE < 10 AND ATYP = 'H' AND ANR = 1
```

Präsenzaufgabe: Erste RA-Anfragen (3)

- Geben Sie die E-Mail-Adresse von Lisa Weiss aus.
- Lösung:
  $\pi_{\texttt{EMAIL}}\left(\sigma_{\texttt{VORNAME='Lisa'} \wedge \texttt{NACHNAME='Weiss'}}(\texttt{STUDENTEN})\right)$
- Zum Vergleich:

```
SELECT EMAIL
FROM   STUDENTEN
WHERE  VORNAME = 'Lisa' AND NACHNAME = 'Weiss'
```

- Also:
  - Selektion σ entspricht der `WHERE`-Klausel.
  - Projektion π entspricht der `SELECT`-Klausel.

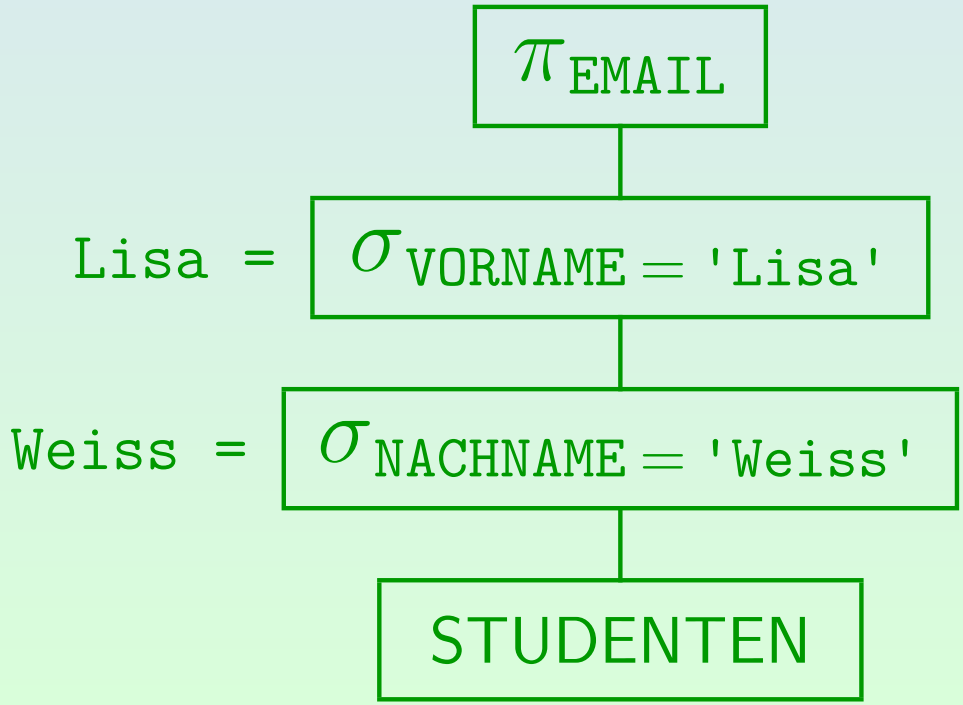
    Wenn man auf alle Spalten projizieren will, wendet man gar keine Projektion an.

Präsenzaufgabe: Erste RA-Anfragen (4)

- In RelaX auch möglich mit Zwischenrelationen:

```
WEISS = sigma NACHNAME = 'Weiss' (STUDENTEN)
LISA  = sigma VORNAME  = 'Lisa'  (WEISS)
pi EMAIL (LISA)
```

Im Skript sind Zwischenrelationen/Sichten auch vorgesehen, dort wird „:=“ für die Zuweisung verwendet, und am Ende jeder Zuweisung steht „;“.



Inhalt

1 Hausaufgabe 9

2 Präsenzaufgabe 10

3 Relationale Algebra

4 Joins in SQL

5 Präsenzaufgabe 11

Aufgabe (von letzter Woche)

- Welche der folgenden Ausdrücke der relationalen Algebra sind syntaktisch korrekt?
  Was bedeuten sie?

  - ☐ $\texttt{STUDENTEN}$.
  - ☐ $\sigma_{\texttt{MAXPT} \neq 10}(\texttt{AUFGABEN})$.
  - ☐ $\pi_{\texttt{VORNAME}}(\pi_{\texttt{NACHNAME}}(\texttt{STUDENTEN}))$.
  - ☐ $\sigma_{\texttt{PUNKTE} \leq 5}(\sigma_{\texttt{PUNKTE} \geq 1}(\texttt{BEWERTUNGEN}))$.
  - ☐ $\sigma_{\texttt{PUNKTE}}(\pi_{\texttt{PUNKTE} = 10}(\texttt{BEWERTUNGEN}))$.

Kartesisches Produkt und Spalten-Namen (1)

- Natürlich muss man auch mehrere Relationen verknüpfen können.
- Die Basis-Operation dazu ist das kartesische Produkt $\times$.
- Das kartesische Produkt entspricht der `FROM`-Klausel:
  Es betrachtet sämtliche Kombinationen von Tupeln aus den beiden Relationen und „klebt“ die beiden Tupel aus jedem Paar zu einem Tupel zusammen.

  Die Tupelkalkül-Sichtweise auf die `FROM`-Klausel ist etwas anders: Dort werden Variablenbelegungen betrachtet, die die Variablen jeweils auf Tupel abbilden. Dort findet das „Zusammenkleben“ nicht statt, aber es werden auch alle Kombinationen von Tupeln betrachtet.
- Wenn R die Attribute A und B hat, und S die Attribute C und D, so hat $R \times S$ die Attribute A, B, C, D.

Kartesisches Produkt und Spalten-Namen (2)

- Beispiel:

A	B
1	2
3	4

$\times$

C	D
6	7
8	9

$=$

A	B	C	D
1	2	6	7
1	2	8	9
3	4	6	7
3	4	8	9

- Da Attributnamen innerhalb eines Tupels eindeutig sein müssen, kann man das kartesische Produkt nur anwenden, wenn R und S keine gemeinsamen Attribute haben.

  Da man Spalten umbenennen kann, ist das keine echte Einschränkung.

- Das kartesische Produkt `STUDENTEN` $\times$ `BEWERTUNGEN` wäre also ein Syntaxfehler, weil es zwei Spalten mit Namen `SID` geben müsste.

Kartesisches Produkt und Spalten-Namen (3)

- RelaX akzeptiert dagegen `STUDENTEN x BEWERTUNGEN`.

  [http://dbis-uibk.github.io/relax/calc/gist/8dc2652578ee12ae756a234c4cf21b3f]

- Dort besteht der Name $R.A$ jeder Spalte aus zwei Teilen:
  - Dem Relationen-Präfix R und
  - dem eigentlichen Spalten-Name A.
- Falls der eigentliche Spalten-Name eindeutig ist, reicht der, um die Spalte zu benennen.

  Das ist so ähnlich wie bei SQL, wo der Präfix eine Tupelvariable ist.

- Bei `STUDENTEN x BEWERTUNGEN` muss man die `SID`-Spalten also mit `STUDENTEN.SID` und `BEWERTUNGEN.SID` ansprechen.
- Dagegen hört die `PUNKTE`-Spalte sowohl auf den Namen „`PUNKTE`“ wie auf den Namen „`BEWERTUNGEN.PUNKTE`“.

Kartesisches Produkt und Spalten-Namen (4)

- Im Prinzip versteht RelaX also z.B.

```
sigma STUDENTEN.SID = BEWERTUNGEN.SID
      (STUDENTEN x BEWERTUNGEN)
```

- Damit gibt es jetzt aber das Problem, dass das nicht mit dem Skript kompatibel wäre.

  Sie müssen damit rechnen, dass es einen kleinen Punktabzug gibt, wenn Sie in Klausur oder Hausaufgaben schreiben, die zwar in RelaX funktionieren, aber nach dem Skript illegal wären.

- Glücklicherweise gibt es eine Lösung, die in RelaX funktioniert UND mit dem Skript kompatibel ist:

```
sigma S.SID = B.SID
      (rho S (STUDENTEN) x rho B (BEWERTUNGEN))
```

  Mit dem rho-Operator kann man einen Präfix explizit einführen. Wenn Sie das so machen, sind sie auf der sicheren Seite.

Kartesisches Produkt und Spalten-Namen (5)

- Nach dem Skript hat jede Spalte nur einen Namen. Wenn Sie mit dem ρ_X-Operator einen Präfix X für alle Attribute eingeführt haben, müssen Sie den auch immer schreiben.
- Die Projektion erlaubt bei Bedarf auch einzelne Spalten-Umbenennungen, auch in RelaX:

```
pi NAME <- NACHNAME, EMAIL (STUDENTEN)
```

- Auch Berechnungen sind in der Projektion möglich (wie im Skript):

```
pi NAME <- concat(VORNAME, concat(' ', NACHNAME))
   (STUDENTEN)
```

- RelaX nimmt bei $\cup$ die Spalten-Namen der linken Tabelle. Damit gilt in RelaX nicht: $R \cup S = S \cup R$.

  Nach den Definitionen im Skript gilt es dagegen.

Joins/Verbunde

- Wenn der natürliche Verbund funktioniert, erspart er Ihnen die Umbenennung:

```
pi VORNAME, NACHNAME, PUNKTE
    (sigma ATYP = 'H' and ANR = 1
        (STUDENTEN join BEWERTUNGEN))
```

Er funktioniert, wenn die Join-Spalten gleiche Namen haben und die einzigen gleich benannten Spalten sind.

- Ein Join mit expliziter Bedingung ist aber auch möglich:

```
pi S.VORNAME, S.NACHNAME, B.PUNKTE
    (sigma B.ATYP = 'H' and B.ANR = 1
        ((rho S STUDENTEN) join S.SID = B.SID
            (rho B BEWERTUNGEN)))
```

Um mit dem Skript kompatibel zu sein, muss man jetzt z.B. auch `S.VORNAME` schreiben. Bei RelaX würde `VORNAME` reichen.

Mengendifferenz

- Die Mengendifferenz kann man in RelaX „`-`“ oder „`\`“ oder „`except`“ schreiben.

  Weitere Mengenoperationen: `union` ($\cup$) und `intersect` ($\cap$).

- Z.B. Studenten ohne Hausaufgaben:

```
STUDENTEN join
    (pi SID (STUDENTEN) -
        pi SID (sigma ATYP = 'H' (BEWERTUNGEN)))
```

  Der natürliche Verbund dient hier nur als Filter: Es werden die Studenten ausgewählt, deren `SID` im Ergebnis der Mengendifferenz vorkommt.

- RelaX hat auch den im Skript erwähnten „Anti-Join“ als Abkürzung (nur das Symbol ist anders):

```
STUDENTEN anti join
    pi SID (sigma ATYP = 'H' (BEWERTUNGEN))
```

Inhalt

1 Hausaufgabe 9

2 Präsenzaufgabe 10

3 Relationale Algebra

4 Joins in SQL

5 Präsenzaufgabe 11

Joins in SQL (1)

- Geben Sie Vorname und Nachname von allen Studierenden aus, die Hausaufgabe 1 bearbeitet haben.
- Klassische Lösung:

```
SELECT S.VORNAME, S.NACHNAME
FROM   STUDENTEN S, BEWERTUNGEN B
WHERE  S.SID = B.SID
AND    B.ATYP = 'H' AND B.ANR = 1
```

- Lösung mit explizitem Join (mit `ON`):

```
SELECT S.VORNAME, S.NACHNAME
FROM   STUDENTEN S JOIN BEWERTUNGEN B
       ON S.SID = B.SID
WHERE  B.ATYP = 'H' AND B.ANR = 1
```

Joins in SQL (2)

- Lösung mit explizitem Join (mit USING):

```
SELECT S.VORNAME, S.NACHNAME
FROM   STUDENTEN S JOIN BEWERTUNGEN B
       USING (SID)
WHERE  B.ATYP = 'H' AND B.ANR = 1
```

- Lösung mit explizitem Join (mit NATURAL):

```
SELECT S.VORNAME, S.NACHNAME
FROM   STUDENTEN S NATURAL JOIN BEWERTUNGEN B
WHERE  B.ATYP = 'H' AND B.ANR = 1
```

- In beiden Fällen werden die Spalten S.SID und B.SID verschmolzen, man kann nicht mehr auf sie zugreifen, sondern nur auf SID.

Joins in SQL (3)

- Bei PostgreSQL und MySQL funktioniert Folgendes, nach dem Standard darf man nur `SID` schreiben:

```
SELECT S.SID, S.VORNAME, S.NACHNAME
FROM   STUDENTEN S NATURAL JOIN BEWERTUNGEN B
WHERE  B.ATYP = 'H' AND B.ANR = 1
```

- Man darf in PostgreSQL und MySQL (MariaDB 5.5.68) aber auch einfach `SID` schreiben (standard-konform).

  Auch `B.SID` wird akzeptiert.

- Bei Join mit `ON` wäre es dagegen ein Fehler, nur `SID` zu schreiben (`column reference "sid" is ambiguous`).

Joins in SQL (4)

- Bei `NATURAL JOIN` ist das Risiko
  - dass es Spalten gibt, an die man nicht gedacht hat, die zufällig auch in beiden Tabellen vorkommen, die aber nicht als Verbundspalten gedacht sind.
  - dass es durch später hinzugefügte Spalten zu einer Veränderung der Join-Bedingung kommt.
- Daher empfehlen wir, `USING` zu verwenden (oder `ON`).

  `ON` war zumindest in der Übergangszeit deutlich portabler als `USING`.
- In der relationalen Algebra dürfen Sie dagegen den natürlichen Verbund gern einsetzen.

  Dort gibt es die Variante mit `USING` nicht, und die Anfragen sind nur Übungsaufgaben, stehen also nicht in Programmen, die Jahre/Jahrzehnte benutzt werden.

Äußerer Verbund in SQL (1)

- Test-Daten:

R	
X	Y
1	4
2	4
3	5

S	
Y	Z
4	10
5	11
6	12

- Was liefert folgende Anfrage?

```
SELECT * FROM R LEFT JOIN S USING (Y)
```

Option A		
X	Y	Z
1	4	10
2	4	10
3	5	11

Option B		
Y	X	Z
4	1	10
4	2	10
5	3	11

Option C		
X	Y	Z
1	4	10
2	4	10
3	5	11
NULL	6	12

Äußerer Verbund in SQL (2)

- Test-Daten:

R	
X	Y
1	4
2	4
3	5

S	
Y	Z
4	10
5	11
6	12

- Was liefert folgende Anfrage?

```
SELECT * FROM R RIGHT JOIN S USING (Y)
```

Option A		
Y	X	Z
4	1	10
4	2	10
5	3	11

Option B		
Y	X	Z
4	1	10
4	2	10
5	3	11
6	NULL	12

Äußerer Verbund in SQL (3)

- Test-Daten:

R

X	Y
1	4
2	4
3	5

S

Y	Z
4	10
5	11
6	12

- Was liefert folgende Anfrage?

```
SELECT * FROM R RIGHT JOIN S USING (Y)
    WHERE X < 3
```

Option A

Y	X	Z
4	1	10
4	2	10

Option B

Y	X	Z
4	1	10
4	2	10
6	NULL	12

Option C

Y	X	Z
4	1	10
4	2	10
5	NULL	11
6	NULL	12

Äußerer Verbund in SQL (4)

- Bedingungen für die linke Tabelle machen in einem linken äußeren Verbund wenig Sinn.
- Z.B. betrachte man diese Anfrage:

```
SELECT A.ATYP, A.ANR, B.SID, B.PUNKTE
FROM   AUFGABEN A LEFT OUTER JOIN BEWERTUNGEN B
       ON A.ATYP = 'H' AND B.ATYP = 'H'
          AND A.ANR = B.ANR
```

- Aufgabe:
  Wird `A.ATYP = 'Z'` in der Ausgabe auftauchen?

  ☐ ja ☐ nein

Inhalt

1 Hausaufgabe 9

2 Präsenzaufgabe 10

3 Relationale Algebra

4 Joins in SQL

5 Präsenzaufgabe 11

Präsenzaufgabe: UNION oder Outer Join

Schreiben Sie folgende Anfrage in SQL (3 Punkte):

- Geben Sie alle Bundesstaaten aus zusammen mit der Anzahl Präsidenten, die in dem jeweiligen Staat geboren wurden.
- Dabei sollen auch Staaten mit 0 Präsidenten gelistet werden.

  Nennen Sie die Spalte mit der Anzahl `num_pres` und sortieren Sie die Ausgabe absteigend nach dieser Anzahl, bei gleicher Anzahl nach dem Staat-Namen.

state_name	num_pres
Virginia	8
⋮	⋮
Wyoming	0

Bitte keine Lösungen mit Unteranfragen unter SELECT!

- president(pres_name, birth_year, years_serv, death_age, party, state_born→state)
- state(state_name, admin_entered, year_entered)