Einführung in Datenbanken

Kapitel 14: Relationale Algebra

Prof. Dr. Stefan Brass

Martin-Luther-Universität Halle-Wittenberg

Wintersemester 2021/22

http://www.informatik.uni-halle.de/~brass/db21/

Lernziele

Nach diesem Kapitel sollten Sie Folgendes können:

- Die Bedeutung der relationalen Algebra erklären.
- Die fünf Basisoperationen der relationalen Algebra aufzählen und erklären.
- Die verschiedenen Arten von Verbunden (Joins) erklären.

  Insbesondere den natürlichen Verbund. Inklusive der Ableitung aus Basisoperationen. Auch (linker, rechter, voller) äußerer Verbund (Outer Join).
- Anfragen in relationaler Algebra formulieren.

  Insbesondere auch (aber nicht nur) mit dem Muster Projektion-Selektion-Join.
- Die syntaktische Korrektheit gegebener Anfragen prüfen.
- Relationale Algebra mit SQL vergleichen.

  Auch (einfache) Anfragen in beiden Richtungen übersetzen.

Inhalt

Beispiel-Datenbank

STUDENTEN

SID	VORNAME	NACHNAME	EMAIL
101	Lisa	Weiss	...
102	Michael	Grau	NULL
103	Daniel	Sommer	...
104	Iris	Winter	...

AUFGABEN

ATYP	ANR	THEMA	MAXPT
H	1	ER	10
H	2	SQL	10
Z	1	SQL	14

BEWERTUNGEN

SID	ATYP	ANR	PUNKTE
101	H	1	10
101	H	2	8
101	Z	1	12
102	H	1	9
102	H	2	9
102	Z	1	10
103	H	1	5
103	Z	1	7

Relationale Algebra: Bedeutung (1)

- Die relationale Algebra (RA) ist eine theoretische Anfragesprache für das relationale Modell.

  Als Maßstab für die Ausdrucksfähigkeit von Anfragesprachen und zum Beweis von Sätzen über Anfragen ist sie sehr nützlich (u.a. weil so einfach).

- Die relationale Algebra wird in keinem kommerziellen System an der Benutzerschnittstelle verwendet.
- Varianten der relationalen Algebra werden aber genutzt, um Auswertungspläne für Anfragen im DBMS darzustellen.

  Für „Performance Tuning“ von relationalen DBMS müssten Sie in der Lage sein, die Auswertungspläne des verwendeten DBMS wenigstens ungefähr zu verstehen.

- Kenntnis der relationalen Algebra hilft, SQL und relationale Datenbanksysteme besser zu verstehen.

  Z.B. sprach man schon immer bei SQL-Anfragen auch über „Joins“ (RA-Operation). Später wurde SQL dann um explizite Joins erweitert.

Relationale Algebra: Bedeutung (2)

- Es gibt Bibliotheken zur Analyse und Verarbeitung großer Datenmengen, die teilweise auf relationaler Algebra basieren:

  Das bedeutet aber nicht, dass sich die Operationen der relationalen Algebra dort 1:1 wiederfinden. Z.B. heißt die Join-Operation öfters `merge`, und Sortieren und Gruppieren sind auch wichtige Operationen. Relationale Algebra ist aber nützlich zum Verständnis dieser Bibliotheken.

  - pandas für Python

    [https://pandas.pydata.org/]

  - DataTable für R

    [https://www.machinelearningplus.com/data-manipulation/datatable-in-r-complete-guide/]

  - Dataset/DataFrame für Apache Spark

    Apache Spark ist ein Framework für „Big Data“ Verarbeitung auf Clustern.

    [https://spark.apache.org/docs/latest/sql-programming-guide.html]

Relationale Algebra (1)

- Eine Algebra ist eine Menge zusammen mit Operationen auf dieser Menge.
- Zum Beispiel bildet die Menge der ganzen Zahlen zusammen mit den Operationen $+$ und $*$ eine Algebra.

  Einen kommutativen Ring mit Einselement.
- Im Fall der relationalen Algebra ist die Menge die Menge aller endlichen Relationen.
- Eine Operation der relationalen Algebra ist $\cup$ (Vereinigung). Da Relationen Mengen (von Tupeln) sind, sind die üblichen Mengenoperationen in der RA enthalten.

  Also auch die Mengendifferenz $\setminus$ und der Schnitt $\cap$.

  Der Schnitt ist aber eine abgeleitete Operation, da man ihn durch eine doppelte Mengendifferenz ausdrücken kann: $R \cap S = R \setminus (R \setminus S)$.

Relationale Algebra (2)

- Eine weitere Operation der RA ist die Selektion.

  Im Gegensatz zu Operationen wie $+$ für Zahlen ist die Selektion σ durch eine einfache Bedingung parametrisiert (wird als Index notiert).

- Z.B. selektiert $\sigma_{\texttt{SID=101}}$ alle Tupel der Eingaberelation, die den Wert „101“ in der Spalte „SID“ haben:

$\sigma_{\texttt{SID=101}}$

BEWERTUNGEN

SID	ATYP	ANR	PUNKTE
101	H	1	10
101	H	2	8
101	Z	1	12
102	H	1	9
102	H	2	9
102	Z	1	10
103	H	1	5
103	Z	1	7

$=$

SID	ATYP	ANR	PUNKTE
101	H	1	10
101	H	2	8
101	Z	1	12

Relationale Algebra (3)

- Da die Ausgabe einer Operation der relationalen Algebra wieder eine Relation ist, kann sie Eingabe für eine andere Operation der Algebra sein.

  Usw., bis das Ergebnis der Anfrage berechnet wird (wieder eine Relation).
  Die relationale Algebra ist so einfach, da das relationale Modell nur ein Konstrukt enthält: die Relation.

- Eine Anfrage ist ein Term/Ausdruck in der Algebra.
- Zum Vergleich: arithmetischer Ausdruck $(x + 2) * y$.
- In relationaler Algebra verknüpft man Relationen:

$$\pi_{\texttt{NACHNAME}}\Big(\texttt{STUDENTEN} \bowtie \sigma_{\texttt{ATYP}=\texttt{'Z'}}(\texttt{BEWERTUNGEN})\Big).$$

Relationale Algebra (4)

Kleine Datenmodell-Unterschiede zu SQL:

- Nullwerte werden in der Definition der relationalen Algebra normalerweise ausgeschlossen, außer wenn Operationen wie äußerer Verbund definiert werden.

  Auch beim äußeren Verbund betrachtet man den Nullwert als einen zusätzlichen Wert zu jedem Datentyp. Die Entsprechung zu einer dreiwertigen Logik wie in SQL ist recht kompliziert.

- Die RA behandelt Relationen als Mengen, d.h. Duplikate werden automatisch eliminiert.

  In SQL sind Relationen Multimengen und können das gleiche Tupel mehrfach enthalten. Falls erforderlich, muss man die Duplikatelimination explizit verlangen (mit „`DISTINCT`“). In der relationalen Algabra muss man über Duplikate nicht nachdenken.

Relationale Algebra (5)

Bedeutung der RA für die DB-Theorie:

- Die relationale Algebra ist viel einfacher als SQL.
  Sie hat nur fünf Basisoperationen und kann vollständig auf einer Seite definiert werden.

  Deshalb kann man sich beim Beweis von Sätzen auf die eigentliche Aussage konzentrieren, während in SQL die vielen syntaktischen Varianten stören.

- Die Relationale Algebra ist auch ein Maßstab, um die Ausdruckskraft von Anfragesprachen zu messen.

- Z.B. kann jede Anfrage, die in relationaler Algebra formuliert ist, auch in SQL formuliert werden.

  D.h. SQL ist zumindest genauso mächtig wie die relationale Algebra.
  Umgekehrt kann jede Anfrage des SQL-Kerns (ohne Nullwerte, Aggregationen und Duplikate) auch in relationaler Algebra formuliert werden.

Relationale Algebra und Logik (1)

- Man kann die relationale Algebra auch so verstehen, dass sie mit Mengen von Variablenbelegungen rechnet.
- Sei ein Relationenschema $(A_1 : D_1, \ldots, A_n : D_n)$ gegeben.
- Man kann Tupel der Relation als Variablenbelegung auffassen, wobei die Attribute A_i die Variablen sind (der Sorte D_i).
- Damit ordnet jedes Tupel den Attributen einen Wert des entsprechenden Datentyps zu.

  Da Tupel mathematisch als Abbildung von $\{1, 2, \ldots, n\}$ auf die entsprechenden Komponentenwerte formalisiert sind, werden hier nur Attributpositionen durch Attributnamen ersetzt.
- Eine Relation ist dann eine Menge von Variablenbelegungen bezüglich der Variablendeklaration $\{A_1/D_1, \ldots, A_n/D_n\}$.

Relationale Algebra und Logik (2)

- Diese Formalisierung (Relation als endliche Menge von Variablenbelegungen) ist für die Relationenalgebra besonders praktisch weil:
  - Man so einen Teil der Definitionen aus der Logik mitbenutzen kann (z.B. Term, atomare Formel).
  - Die Relationenalgebra sich so auf andere Logik-basierte Datenmodelle verallgemeinern läßt.
  - Man so recht leicht einsehen kann, dass Relationenalgebra und bereichsbeschränkte Formeln gleich mächtig sind.

    Die Bereichsbeschränkung ist eine Eigenschaft von Formeln, die erlaubt, für jede Variable nur endlich viele Werte einzusetzen.
- Die relationale Algebra ist operationaler, die Logik deklarativer.

Inhalt

Selektion (1)

- Die Operation σ_F selektiert eine Teilmenge der Tupel einer Relation, nämlich die, die die Bedingung F erfüllen. Die Selektion wirkt wie ein Filter auf die Eingabemenge.

  σ ist der griechische Buchstabe sigma.

  In ASCII kann man z.B. `SEL[Bedingung](Relation)` schreiben.

  In der Übungssoftware „RA“ [https://users.cs.duke.edu/~junyang/radb/] schreibt man: „`\select_{Bedingung} Relation`“.

  In der Übungssoftware „RelaX“ [https://dbis-informatik.uibk.ac.at/relax]: „`sigma Bedingung (Relation)`“. Dies kann man im Web ausprobieren: [http://dbis-uibk.github.io/relax/calc/local/uibk/local/0].

- Beispiel:

$$\sigma_{A=1}\left(\begin{array}{|c|c|}\hline A & B \\ \hline 1 & 3 \\ 1 & 4 \\ 2 & 5 \\ \hline\end{array}\right) = \begin{array}{|c|c|}\hline A & B \\ \hline 1 & 3 \\ 1 & 4 \\ \hline\end{array}$$

Selektion (2)

- Die Selektionsbedingung F ist eine atomare Formel bezüglich
  - der Datentyp-Signatur $\Sigma_{\mathcal{D}}$, und
  - der Variablendeklaration ν, die für jedes Attribut der Eingaberelation eine Variable gleichen Namens und gleicher Sorte enthält.
- Typischerweise hat die Bedingung F die Form:

$$\langle \text{Term} \rangle \ \langle \text{Vergleichsoperator} \rangle \ \langle \text{Term} \rangle$$

Die Bedingung liefert einen Wahrheitswert (wahr oder falsch) für ein gegebenes Eingabetupel. Vergleichsoperatoren: $=$, $\neq$, $<$, $\leq$, $>$, $\geq$.

Selektion (3)

- Ein Term („Wertausdruck“) ist typischerweise
  - ein Attributname, oder
  - eine Datentypkonstante.

  Er kann für ein gegebenes Tupel zu einem Datentypelement ausgewertet werden.
- Im allgemeinen können aber alle in der Datentyp-Signatur $\Sigma_{\mathcal{D}}$ deklarierten Funktionen benutzt werden, um komplexe Terme zu bilden, z.B. $+$, $-$, $*$, $/$.
- Entsprechend können in der Selektionsbedingung alle Datentyp-Prädikate verwendet werden, z.B. `LIKE`.

Selektion (4)

- Beispiele für Bedingungen:
  - `NACHNAME = 'Weiss'`
  - `PUNKTE >= 8`
  - `PUNKTE = MAXPT`

  Die Eingaberelation muss beide Attribute enthalten.
- Natürlich müssen die Attribute aus der Selektionsbedingung auch in der Eingabetabelle vorkommen:

$$\sigma_{C=1}\left(\begin{array}{|c|c|}\hline A & B \\ \hline 1 & 3 \\ 2 & 4 \\ \hline\end{array}\right) = \textbf{Error}$$

Selektion (5)

- $\sigma_F(R)$ kann wie folgt implementiert werden:

```
(1)  Create new temporary relation T;
(2)  foreach tuple t from input relation R do
(3)      Evaluate condition F for tuple t;
(4)      if true then
(5)          insert t into T;
(6)      fi
(7)  od;
(8)  return T;
```

- Mit anderen Datenstrukturen (B-Baum Index) kann es möglich sein, $\sigma_F(R)$ zu berechnen, ohne jedes Tupel aus der Eingaberelation durchzugehen.

Erweiterte Selektion (1)

- In der grundlegenden Selektions-Operation sind nur atomare Formeln (meist Vergleiche) als Bedingungen möglich.
- Man kann aber auch die Kombination dieser atomaren Formeln durch die logischen Junktoren $\wedge$ (und), $\vee$ (oder), und $\neg$ (nicht) zulassen:

F_1	F_2	$F_1 \wedge F_2$	$F_1 \vee F_2$	$\neg F_1$
falsch	falsch	falsch	falsch	wahr
falsch	wahr	falsch	wahr	wahr
wahr	falsch	falsch	wahr	falsch
wahr	wahr	wahr	wahr	falsch

Erweiterte Selektion (2)

- Die Selektionsbedingung muss die Auswertung für jedes Eingabetupel getrennt/einzeln zulassen.

  Somit sind „es gibt“ ($\exists$) und „für alle“ ($\forall$) und verschachtelte Anfragen in Selektionsbedingungen nicht erlaubt.

- Die erweiterte Form der Selektion ist nicht notwendig, da man sie immer durch die Basisoperationen der relationalen Algebra ausgedrücken kann.

  Aber sie ist bequem.

- Z.B. ist $\sigma_{F_1 \wedge F_2}(R)$ äquivalent zu $\sigma_{F_1}\left(\sigma_{F_2}(R)\right)$.

  $\vee$ und $\neg$ benötigen $\cup$ (Union) und $\setminus$ (Mengendifferenz) die auch zu den Basisoperationen der relationalen Algebra gehören (s.u.): $\sigma_{F_1 \vee F_2}(R)$ ist äquivalent zu $\sigma_{F_1}(R) \cup \sigma_{F_2}(R)$ und $\sigma_{\neg F}(R)$ ist äquivalent zu $R \setminus \sigma_F(R)$.

Übung

Schreiben Sie folgende Anfragen in relationaler Algebra:

- Welche Aufgaben behandeln „SQL“?

  Geben Sie die ganze Zeile der Tabelle aus.

  Die Projektion (zur Eliminierung von Spalten) wird unten behandelt.

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

Dies bezieht sich auf das Schema auf Folie 4:

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

Inhalt

Projektion (1)

- Die Projektion π wird typischerweise benutzt, um Attribute (Spalten) aus der Eingaberelation zu eliminieren (so wie die Selektion Zeilen eliminiert).

  π ist der griechische Buchstabe „pi“.

  In Datenbanken bedeutet dies immer „Projektion“, und nicht 3.14....

  Einige Autoren verwenden ein großes Π zur Unterscheidung.

  In ASCII kann man z.B. schreiben: `PROJ[Spalten](Relation)`.

  In der „RA“ Übungssoftware: `\project_{Spalten} Relation`.

  In der „RelaX“ Übungssoftware: `pi Spalten (Relation)`.

- Beispiel:

$$\pi_{A,C}\left(\begin{array}{|c|c|c|}\hline A & B & C \\ \hline 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \\ \hline\end{array}\right) = \begin{array}{|c|c|}\hline A & C \\ \hline 1 & 7 \\ 2 & 8 \\ 3 & 9 \\ \hline\end{array}$$

Projektion (2)

- Im allgemeinen hat die Projektion die Form
$$\pi_{B_1 \leftarrow t_1, \ldots, B_k \leftarrow t_k}(R),$$
wobei die B_i Attributnamen sind (die nicht in der Eingabe vorzukommen brauchen), und die t_i Terme bezüglich
  - der Datentyp-Signatur $\Sigma_{\mathcal{D}}$, und
  - der Variablendeklaration ν, die für jedes Attribut der Eingaberelation eine Variable gleichen Namens und gleicher Sorte enthält.

  Doppelte Attribute sind nicht erlaubt, d.h. $B_i \neq B_j$ für $i \neq j$.
- Die Umbenennung von Attributen ist hier Teil der Projektion.

  Manche Autoren verwenden eine getrennte Umbenennungsoperation.

  Wenn man aber Berechnungen in der Projektion zulässt (beliebige Terme, nicht nur Attribute), muss man definieren, wie die Ergebnisspalten heissen sollen.

Projektion (3)

- Die Projektion erzeugt für jedes Eingabetupel ein Ausgabetupel.

  Falls jedoch zwei Eingabetupel zum gleichen Ausgabetupel führen, wird das Duplikat eliminiert.

- Sei $\mathcal{A}$ die Variablenbelegung, die einem Eingabetupel $(A_1 : d_1, \ldots, A_n : d_n)$ entspricht.

- Dann erzeugt die Projektion

  $$\pi_{B_1 \leftarrow t_1, \ldots, B_k \leftarrow t_k}(R),$$

  daraus das Tupel

  $$(B_1 : \langle \mathcal{I}_\mathcal{D}, \mathcal{A} \rangle[t_1], \ldots, B_k : \langle \mathcal{I}_\mathcal{D}, \mathcal{A} \rangle[t_k]).$$

Projektion (4)

- Die Projektion wendet eine Abbildung auf jedes Eingabetupel an und liefert die Menge der Funktionswerte (Ausgabetupel).

  Wie „`map`“ in funktionalen Sprachen, z.B. Lisp.

- Jedes Eingabetupel wird lokal auf ein Ausgabetupel abgebildet.

  D.h. es sind nur Funktionen erlaubt, deren Ausgabe nur von jeweils einem Eingabetupel abhängt. Man kann also mit der Projektion nicht Werte verschiedener Eingabetupel zu einem Ausgabetupel zusammenfassen (siehe aber kartesisches Produkt unten). Ansonsten sind sehr allgemeine Tupel-zu-Tupel Funktionen erlaubt.

Projektion (5)

- Meistens gibt es ein Ausgabetupel für jedes Eingabetupel. Falls jedoch zwei Eingabetupel zu dem gleichen Ausgabetupel führen, wird das Duplikat eliminiert.

  DBMS verwenden eine explizite Ausgabeeliminierung, wenn benötigt.

  Aber in der Theorie sind Relationen Mengen.

- Beispiel:

$$\pi_B \left(\begin{array}{|c|c|} \hline A & B \\ \hline 1 & 4 \\ 2 & 5 \\ 3 & 4 \\ \hline \end{array} \right) = \begin{array}{|c|} \hline B \\ \hline 4 \\ 5 \\ \hline \end{array}$$

Projektion (6)

- $\pi_{B_1 \leftarrow t_1, \ldots, B_k \leftarrow t_k}(R)$:

```
(1)  Create new temporary relation T;
(2)  foreach (A_1: d_1, ..., A_n: d_n) in R do
(3)      for i = 1 to k do
(4)          Let d'_i be the value of t_i for the
(5)              input tuple (A_1: d_1, ..., A_n: d_n)
(6)              /* e.g. if t_i = A_j then d'_i = d_j */
(7)      insert (B_1: d'_1, ..., B_k: d'_k) into T;
(8)  od;
(9)  return T;
```

Diese Programmskizze setzt voraus, dass „**insert**“ Duplikate eliminiert.

Projektion (7)

- Falls ein Ergebnisterm selbst ein Attributname ist, und man diesen Attributnamen auch im Ausgabetupel verwenden will, ist es nicht nötig, explizit einen neuen Namen anzugeben.
- $\pi_{A_{i_1},\ldots,A_{i_k}}(R)$ ist also eine Abkürzung für
$$\pi_{A_{i_1}\leftarrow A_{i_1},\ldots,A_{i_k}\leftarrow A_{i_k}}(R).$$
- Da es die typische Anwendung der Projektion ist, überflüssige Attribute/Spalten „wegzuprojizieren“, ist das eine sehr nützliche Abkürzung.

Projektion (8)

Weitere Beispiele:

- Umbenennung einer Spalte (von SID in NR):

$$\pi_{\texttt{NR} \leftarrow \texttt{SID},\ \texttt{VORNAME},\ \texttt{NACHNAME}}(\texttt{STUDENTEN}).$$

- Berechnung eines neuen Spaltenwertes durch eine Datentyp-Operation (String-Kontatenation):

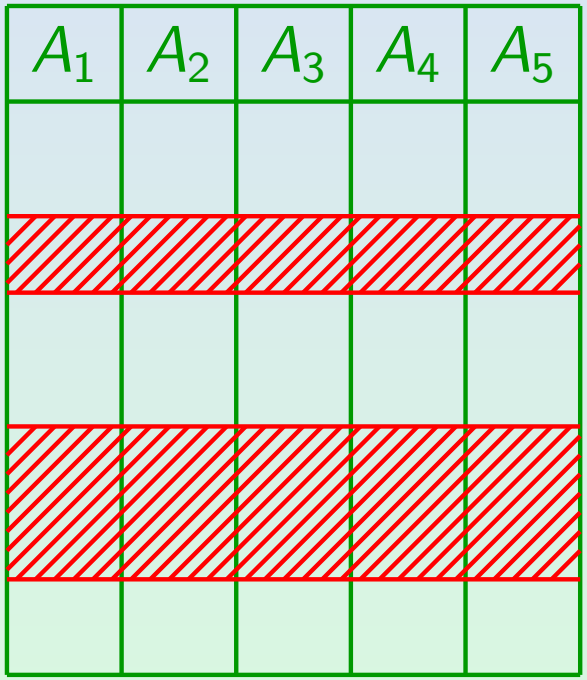
$$\pi_{\texttt{SID},\ \texttt{NAME} \leftarrow \texttt{VORNAME} \,||\, '\ ' \,||\, \texttt{NACHNAME}}(\texttt{STUDENTEN}).$$

- Erstellung einer neuen Spalte mit konstantem Wert (nützlich als Eingabe für eine Vereinigung mit einer Teilrelation mit einem anderen Wert):

$$\pi_{\texttt{SID},\ \texttt{VORNAME},\ \texttt{NACHNAME},\ \texttt{NOTE} \leftarrow 1}(\texttt{STUDENTEN}).$$

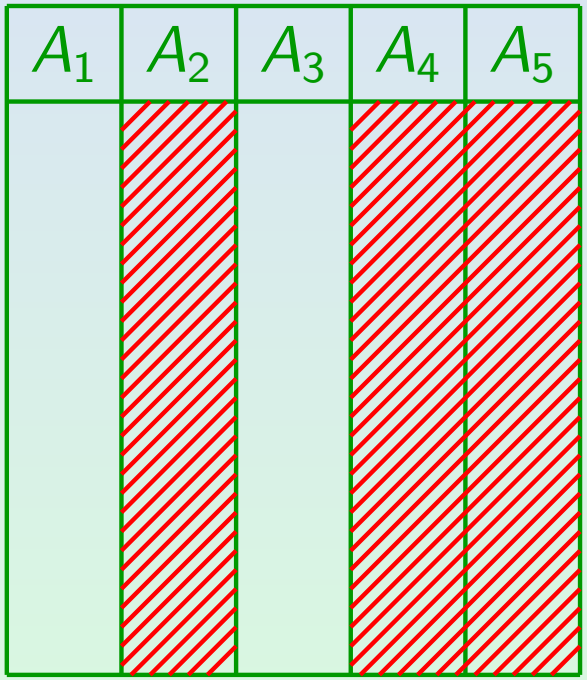
Zusammenfassung

Selektion σ



(Filtert Eingabezeilen)

Projektion π



(wendet Funktion auf jede Zeile an)

Operationen schachteln (1)

- Da das Ergebnis einer Operation der relationalen Algebra wieder eine Relation ist, kann man es als Eingabe für eine weitere Operation verwenden.
- Zum Beispiel, wenn man die abgegebenen Übungen von Student 102 ausgeben möchte:

$$\pi_{\texttt{ATYP, ANR}}\left(\sigma_{\texttt{SID}=102}(\texttt{BEWERTUNGEN})\right)$$

- Man kann ein Zwischenergebnis in einer temporären Relation speichern (andere Sicht: Makro-Definition).

$$\texttt{S102} := \sigma_{\texttt{SID}=102}(\texttt{BEWERTUNGEN});$$

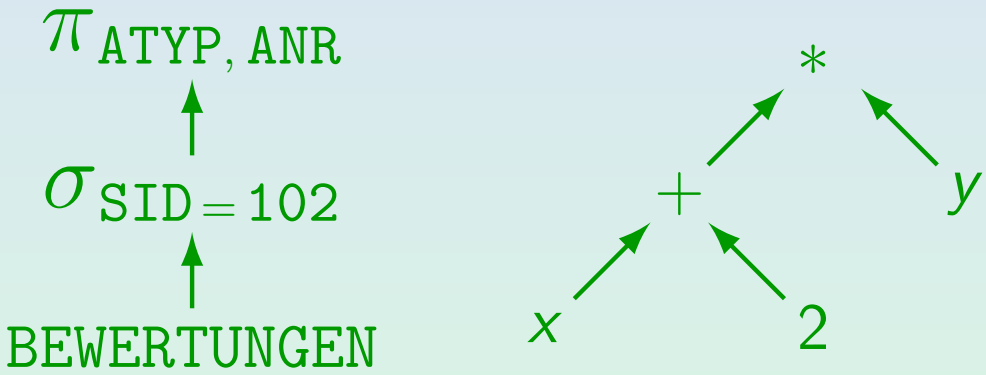
$$\pi_{\texttt{ATYP, ANR}}(\texttt{S102})$$

In RelaX schreibt man = statt := und läßt das „;“ weg (man kann mehrere Hilfsrelationen definieren und dann die eigentliche Anfrage schreiben).

In RA schreibt man :- statt :=, und das „;“ ist nötig (auch nach der Anfrage).

Operationen schachteln (2)

- Ausdrücke der relationalen Algebra können klarer werden, wenn man sie in Operatorbäumen darstellt.



- Zum Vergleich ist rechts der Operatorbaum des arithmetischen Ausdrucks $(x + 2) * y$ dargestellt.

Zwischenergebnisse fließen entlang der Linien von unten nach oben.

Basis-Operanden

- Die Blätter eines Operatorbaumes sind
  - Namen von Datenbankrelationen
  - konstante Relationen (explizite Tupelaufzählung).
- Ein Relationsname R ist ein legaler Ausdruck der relationalen Algebra. Sein Wert ist die gesamte, unter diesem Namen gespeicherte, Relation.

  Die relationale Algebra fordert nicht, dass jede Anfrage eine Projektion enthalten muss. Eine Projektion auf alle Attribute ist überflüssig.
- Eine konstante Relation mit einem Tupel wird in folgener Form geschrieben: $\{(A_1 : c_1, \ldots, A_n : c_n)\}$.

  Für mehrere Tupel verwende man $\cup$. Die c_i sind Konstanten.

Übungen (1)

Schreiben Sie folgende Anfrage in relationaler Algebra:

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

  Schreiben Sie die Anfrage als Baum und verschachtelt mit Klammern.

Dies bezieht sich auf das Schema auf Folie 4:

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

Übungen (2)

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

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

Inhalt

Kartesisches Produkt (1)

- Häufig müssen Ausgabetupel berechnet werden, die aus mehreren Eingabetupeln abgeleitet sind.

  Für σ und π gilt, dass jedes Ausgabetupel von einem einzigen Eingabetupel abgeleitet ist. Da π Duplikate eliminiert, ist es möglich, dass das gleiche Ausgabetupel von zwei verschiedenen Eingabetupeln abgeleitet wird, aber dann ist eins der beiden überflüssig.

- Dies leistet das „kartesische Produkt“ $\times$.

  Vgl. „kartesische Koordinaten“. Man nennt es auch „Kreuzprodukt“.

- $R \times S$ verbindet („klebt zusammen“) jedes Tupel von R mit jedem Tupel von S.

  In ASCII schreibt man „`R PRODUCT S`“ oder „`R x S`“ für $R \times S$. Falls nötig, setzt man `(...)` um die Eingaberelationen.
  In der RA Übungssoftware heisst es „`R \cross S`“. In RelaX: „`R x S`.

Kartesisches Produkt (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.
- Das ist keine echte Einschränkung.
  Man kann die Attribute notfalls erst umbenennen (π).

  In der Literatur ist dagegen eine automatische Umbenennung üblicher.

  Siehe aber die nächste Folie.

Kartesisches Produkt (3)

- Einige Autoren definieren $\times$ so, dass doppelte Attribute automatisch umbenannt werden:
  - Z.B. für Relationen $R(A, B)$ und $S(B, C)$ hat das Produkt $R \times S$ die Attribute $(R.A, R.B, S.B, S.C)$.
  - Sie erlauben außerdem, den Präfix wegzulassen, wenn der Rest eindeutig ist, z.B. A und C.
- **In dieser Vorlesung ist das nicht gestattet!**

  Die formale Definition ist einfacher: Was passiert z.B. mit $((R \cup S) \times T)$ und mit $R \times R$? Normalerweise geben Autoren, die solche Namen zulassen, keine formale Definition. In dieser Vorlesung kann man den Umbenennungsoperator (siehe unten) verwenden, um Attributnamen der Form $R.A$ einzuführen, aber dann kann A allein nicht verwendet werden: Jedes Attribut hat genau einen Namen.

Kartesisches Produkt (4)

- Ist $t = (A_1 : a_1, \ldots, A_n : a_n)$, $u = (B_1 : b_1, \ldots, B_m : b_m)$, so sei $t \circ u = (A_1 : a_1, \ldots, A_n : a_n, B_1 : b_1, \ldots, B_m : b_m)$.
- Das kartesische Produkt $R \times S$ kann mit zwei verschachtelten Schleifen berechnet werden:

```
(1)  Create new temporary relation T;
(2)  foreach tuple t in R do
(3)      foreach tuple u in S do
(4)          insert t ∘ u into T;
(5)      od;
(6)  od;
(7)  return T;
```

Kartesisches Produkt (5)

- Die Relation R enthalte n Tupel, und die Relation S enthalte m Tupel, dann enthält $R \times S$ $n * m$ Tupel.
- Das kartesische Produkt allein ist selten sinnvoll, da es zu einem „Aufblähen“ der Relationsgröße führt.
- Das Problem ist, dass $R \times S$ jedes Tupel von R mit jedem Tupel von S verbindet. Normalerweise will man aber nur bestimmte Paare von Tupeln verknüpfen.
- Somit ist das kartesische Produkt nur als Eingabe für eine folgende Selektion sinnvoll ($\rightarrow$ Verbund).

Umbenennung

- Ein Operator $\rho_R(S)$, der „$R.$“ vor alle Attributnamen stellt, ist manchmal nützlich:

$$\rho_R\left(\begin{array}{|c|c|}\hline A & B \\ \hline 1 & 2 \\ 3 & 4 \\ \hline\end{array}\right) = \begin{array}{|c|c|}\hline R.A & R.B \\ \hline 1 & 2 \\ 3 & 4 \\ \hline\end{array}$$

- Dies ist nur eine Abkürzung für: $\pi_{R.A \leftarrow A,\, R.B \leftarrow B}(S)$.
- Sonst enthalten Attributnamen in der relationalen Algebra nicht automatisch den Relationsnamen.

  Das ist in RA und RelaX anders: Beide Programme erlauben, Attribute als „Relation.Attribut“ anzusprechen. RA hat Operationen `\rename_{A,B,...}` zum Umbennen von Spalten, und `\rename_{R:*}` zum Umbenennen der Relation. In RelaX ändert man den Relations-Präfix mit `rho X (R)`, und ausgewählte Spalten können mit `rho a<-b, c<-d (R)` umbenannt werden (man kann auch `b->a` schreiben). Die übrigen Spalten bleiben dabei erhalten.

Verbund/Join (1)

- Da die Kombination von kartesischem Produkt und Selektion so verbreitet ist, wurde dafür ein spezielles Symbol eingeführt:

  $R \underset{A=B}{\bowtie} S$ ist eine Abkürzung für $\sigma_{A=B}(R \times S)$.

- Diese Operation nennt man „Verbund“ (Join):
  Sie wird verwendet, um zwei Tabellen (d.h. ihre Tupel) zu verbinden.

  In ASCII schreibt man z.B. „`R JOIN[A=B] S`“.

  In der Übungssoftware RA ist es „`R \join_{A=B} S`“. In der Übungssoftware RelaX: „`R join A=B S`“.

- Der Verbund ist eine der wichtigsten und nützlichsten Operationen der relationalen Algebra.

  Gleich nach der Selektion.

Verbund/Join (2)

STUDENTEN ⋈ BEWERTUNGEN						
SID	VORNAME	NACHNAME	EMAIL	ATYP	ANR	PUNKTE
101	Lisa	Weiss	...	H	1	10
101	Lisa	Weiss	...	H	2	8
101	Lisa	Weiss	...	Z	1	12
102	Michael	Grau	NULL	H	1	9
102	Michael	Grau	NULL	H	2	9
102	Michael	Grau	NULL	Z	1	10
103	Daniel	Sommer	...	H	1	5
103	Daniel	Sommer	...	Z	1	7

- Die Studentin Iris Winter taucht nicht auf, da sie keine Hausaufgabe abgegeben und nicht an einer Klausur teilgenommen hat.
- Oben ist der natürliche Verbund der beiden Tabellen gezeigt. Im folgenden wird aber zunächst der Standard-Verbund erklärt.

Verbund/Join (3)

- $R \underset{A=B}{\bowtie} S$ kann ausgewertet werden wie $\sigma_{A=B}(R \times S)$:

```
(1)  Create new temporary relation T;
(2)  foreach tuple t in R do
(3)      foreach tuple u in S do
(4)          if t.A = u.B then
(5)              insert t ∘ u into T;
(6)          fi;
(7)      od;
(8)  od;
(9)  return T;
```

Verbund/Join (4)

- Die obige Prozedur nennt man „nested loop join“.
- Man beachte, dass das Zwischenergebnis von $R \times S$ nicht explizit gespeichert wird.

  Tatsächlich versucht ein DBMS auch sonst, keine Zwischenergebnisse zu materialisieren (soweit möglich). Jeder Algebra-Operator berechnet nur jeweils ein Tupel („Pipeline-Auswertung“). Dadurch ist der Nested Loop Join ohnehin das Gleiche wie $\times$ gefolgt von σ.

- Es wurden viele verschiedene Algorithmen zur Berechnung des Verbunds entwickelt.

  Weil der Verbund eine wichtige und relativ teure Operation ist.
  Z.B. „merge join“, „index join“, „hash join“. Siehe Vorlesung „DB II B“.

Verbund/Join (5)

- Die Verbundbedingung muss nicht die Form $A = B$ haben (obwohl dies am häufigsten vorkommt). Es kann eine beliebige Bedingung sein, z.B. auch $A < B$.

  Ein Verbund mit einer Bedingung der Form $A = B$ (oder $A_1 = B_1 \wedge \cdots \wedge A_n = B_n$) nennt man „equijoin“.

  Verbunde mit allgemeinen Bedingungen werden auch „Theta-Join“ genannt, weil man das Symbol θ für einen beliebigen Vergleichsoperator benutzt.

- Eine typische Anwendung ist es, Tupel entsprechend einem Fremdschlüssel zu verknüpfen, z.B.

$$\texttt{BEWERTUNGEN} \underset{\texttt{SID}=\texttt{SID}'}{\bowtie} \pi_{\texttt{SID}' \leftarrow \texttt{SID}, \texttt{NACHNAME}, \texttt{EMAIL}}(\texttt{STUDENTEN})$$

  Die Umbenennung des Attributs „SID“ ist notwendig, da das kartesische Produkt disjunkte Attributnamen verlangt. Aber der unten erklärte natürliche Verbund macht dies wieder überflüssig.

Verbund/Join (6)

- Der Verbund kombiniert Tupel und agiert als Filter:
  Er eliminiert Tupel ohne Verbundpartner.
  (Fremdschlüssel sichern Existenz von Verbundpartnern!)

A	B
1	2
3	4

$\underset{B=C}{\bowtie}$

C	D
4	5
6	7

$=$

A	B	C	D
3	4	4	5

- Ein „Semijoin“ ($\ltimes$, $\rtimes$) agiert nur als Filter.

  Er entspricht einem Verbund und anschließender Projektion auf die Attribute der linken (linker SJ) oder rechten Relation (rechter SJ).
  RelaX hat die Operationen „`left semi join`“ und „`right semi join`“.

- Ein „äußerer Verbund“ (vgl. Ende dieses Kapitels) agiert nicht als Filter: Er erhält alle Eingabetupel.

Natürlicher Verbund (1)

- Eine weitere nützliche Abkürzung ist der „natürliche Verbund“ (natural join) $\bowtie$.

  Anstelle dieses Symbols kann man auch „*“ verwenden.

  In RA schreibt man `R \join S`. RelaX interpretiert „`join`“ ohne Bedingung als natürlichen Verbund, man kann aber auch „`natural join`“ schreiben.

- Er verknüpft Tupel, die die gleichen Werte bei Attributen mit gleichem Namen haben.

  Während kartesisches Produkt und allgemeiner Verbund verlangen, dass die Attribute beider Relationen verschiedene Namen haben, verwendet der natürliche Verbund die gleichen Namen für die Bedingung.

A	B
1	2
3	4

$\bowtie$

B	C
4	5
4	8
6	7

$=$

A	B	C
3	4	5
3	4	8

Natürlicher Verbund (2)

- Der natürliche Verbund der beiden Relationen
  - $R(A_1, \ldots, A_n, B_1, \ldots, B_k)$ und
  - $S(B_1, \ldots, B_k, C_1, \ldots, C_m)$

  produziert im DB-Zustand $\mathcal{I}$ alle Tupel der Form

  $$(a_1, \ldots, a_n, b_1, \ldots, b_k, c_1, \ldots, c_m),$$

  wobei
  - $(a_1, \ldots, a_n, b_1, \ldots, b_k) \in \mathcal{I}[R]$ und
  - $(b_1, \ldots, b_k, c_1, \ldots, c_m) \in \mathcal{I}[S]$.

Natürlicher Verbund (3)

- Der natürliche Verbund entspricht nicht nur dem kartesischen Produkt gefolgt von einer Selektion, sondern
  - benennt jeweils eine Kopie der gemeinsamen Attribute vor dem kartesischen Produkt um, und
  - verwendet eine Projektion, um diese doppelten Attribute am Schluß zu eliminieren.
- Seien z.B. $R(A, B)$ und $S(B, C)$ gegeben.
  Dann ist $R \bowtie S$ eine Abkürzung für

$$\pi_{A,B,C}\Big(\sigma_{B=B'}(R \times \pi_{B' \leftarrow B,\, C}(S))\Big).$$

Algebraische Gesetze (1)

- Definition: Zwei RA-Ausdrücke Q_1 und Q_2 heißen äquivalent genau dann, wenn ihr Ergebnisschema gleich ist, und für alle DB-Zustände $\mathcal{I}$ gilt:

$$\mathcal{I}[Q_1] = \mathcal{I}[Q_2].$$

  Man schreibt dann $Q_1 = Q_2$.

  Es gibt tatsächlich zwei Begriffe von Äquivalenz, abhängig davon, ob man alle strukturell möglichen Zustände betrachtet, oder nur die, die den Integritätsbedingungen genügen. Die erste Alternative ist eine stärkere Forderung. Bei der zweiten Alternative erhält man mehr äquivalente Anfragen. Die folgenden Aussagen gelten in beiden Fällen.

- Äquivalente Anfragen liefern also immer das gleiche Ergebnis.

Algebraische Gesetze (2)

- Der Verbund erfüllt das Assoziativitätsgesetz:
$$(R \bowtie S) \bowtie T \;=\; R \bowtie (S \bowtie T).$$
- Somit werden die Klammern nicht benötigt:
$$R \bowtie S \bowtie T.$$
- Der Verbund ist nicht ganz kommutativ:
Die Reihenfolge der Spalten ist unterschiedlich.

  Fasst man Tupel als Variablenbelegung auf, gibt es keine Reihenfolge.
- Falls jedoch danach eine Projektion folgt, ist das egal (man kann π auch für diesen Zweck einführen):
$$\pi_{\ldots}(R \bowtie S) \;=\; \pi_{\ldots}(S \bowtie R).$$

Algebraische Gesetze (3)

- Vom Anfrageoptimierer eines relationalen DBMS werden weitere algebraische Gesetze verwendet.
- Z.B. wenn sich F nur auf S bezieht, dann gilt

$$\sigma_F(R \bowtie S) = R \bowtie \sigma_F(S).$$

  Die rechte Seite kann meist effizienter ausgewertet werden (abhängig von Relationsgröße, Indexen).
- Für diese Vorlesung ist Effizienz nicht so wichtig.

  Der Anfrageoptimierer formt eine gegebene Anfrage automatisch in eine effizientere Variante um, so dass sich der Nutzer darum nicht kümmern muss. Für eine Lösung, die immer das richtige Ergebnis liefert, und die nicht unnötig kompliziert ist (z.B. π auf alle Spalten), wird es die volle Punktzahl geben.

Algebraische Gesetze (4)

- Obige Äquivalenzen gelten auch für das kartesische Produkt anstelle des Verbundes:
  - Das kartesische Produkt ist assoziativ:
    $$(Q_1 \times Q_2) \times Q_3 = Q_1 \times (Q_2 \times Q_3).$$
  - Ist A ein Attribut im Schema von Q_1, dann ist
    $$\sigma_{A=d}(Q_1 \times Q_2) = (\sigma_{A=d}(Q_1)) \times Q_2.$$
- Die Reihenfolge aufeinander folgender Selektionen ist egal:
  $$\sigma_{F_1}(\sigma_{F_2}(Q)) = \sigma_{F_2}(\sigma_{F_1}(Q)).$$
- Zwei Projektionen direkt nacheinander sind überflüssig:
  $$\pi_{A_1,\ldots,A_n}(\pi_{B_1,\ldots,B_m}(Q)) = \pi_{A_1,\ldots,A_n}(Q).$$

Algebraische Gesetze (5)

- Satz: Die Äquivalenz von RA-Ausdrücken ist unentscheidbar.

  D.h. man kann kein Programm schreiben, das zwei beliebige RA-Ausdrücke Q_1 und Q_2 einliest, und „ja“ oder „nein“ ausgibt, abhängig davon, ob Q_1 und Q_2 äquivalent sind, und das garantiert nach endlicher Rechenzeit anhält. Für spezielle Paare von Q_1 und Q_2 ist es natürlich häufig möglich, festzustellen, ob sie äquivalent sind, oder nicht. Es wird aber immer Fälle geben, die ein Programm nicht behandeln kann.
  Dies ist auch eine Aussage über die Mächtigkeit der Relationenalgebra.
  Ein Anfrageoptimierer verwendet die bekannten Äquivalenzen (u.a. Umordnung von Verbunden, Aufspalten und Verschieben von Selektionen), um eine Reihe von äquivalenten Anfragen zu erzeugen, und schätzt dann die „Kosten“ (Ausführungsdauer). Er kann aber niemals alle äquivalenten Formulierungen erzeugen, das Ergebnis muss also nicht optimal sein.

Häufiges Anfragemuster (1)

- Folgende Anfragestruktur ist sehr häufig:

$$\pi_{A_1,\ldots,A_k}\Big(\sigma_F(R_1 \bowtie \cdots \bowtie R_n)\Big).$$

  - Erst verknüpft man alle Tabellen, die für die Anfrage benötigt werden, mit einem Verbund.

    Wenn die Attribute der Relationen gut benannt sind, reicht häufig ein natürlicher Verbund. Sonst muss man ggf. umbenennen und explizite Join-Bedingungen verwenden.

  - Dann selektiert man die relevanten Tupel.

    Die Selektionsbedingung kann sich auf die Attribute von allen Tabellen beziehen, die im ersten Schritt verknüpft wurden.

  - Als drittes projiziert man auf die Attribute, die ausgeben werden sollen.

Häufiges Anfragemuster (2)

- Muster sind oft nützlich.

  Damit können typische Anfragen schnell formuliert werden.

- Aber die Operationen der relationalen Algebra können auf jede Weise kombiniert werden. Man muss sich nicht notwendig an dieses Muster halten.

  Z.B. wenn keine Projektion oder Selektion benötigt wird, wäre es falsch, die Anfrage zu verkomplizieren, indem man eine Projektion auf alle Attribute oder eine Selektion mit einer stets wahren Bedingung verwendet (beides entspricht einfach einer Identitätsabbildung).

- Dagegen sind in SQL die Schlüsselworte `SELECT` und `FROM` Pflicht, und die Reihenfolge muss immer sein:

  `SELECT ... FROM ... WHERE ...`

Häufiges Anfragemuster (3)

- Um eine Anfrage zu formulieren, sollte man zunächst über die benötigten Tabellen nachdenken:
  - Normalerweise nennt die in natürlicher Sprache gestellte Anfrage gewisse Attribute.

    Es ist auch möglich, dass Datenwerte genannt werden, die gewissen Attributen entsprechen (z.B. Studentennamen).
  - Jedes solche Attribut benötigt zumindest eine Relation, die dieses Attribut enthält.

    So dass das Attribut in der Selektionsbedingung oder der Projektionsliste verwendet werden kann.

Häufiges Anfragemuster (4)

- Anfrageformulierung, fortgesetzt:
  - Schließlich benötigt man manchmal Zwischenrelationen, um einen Verbund sinnvoll zu machen.
  - Z.B. seien $R(A, B)$, $S(B, C)$, $T(C, D)$ gegeben und die Attribute A und D werden gebraucht. Dann wäre $R \bowtie T$ nicht korrekt. Warum?
  - Korrekt ist hier der Verbund $R \bowtie S \bowtie T$.
  - Eine Zeichnung der Fremdschlüsselbeziehungen zwischen den Tabellen ist oft hilfreich (entspricht typischen Verknüpfungen mittels Verbund).

Übung

Schreiben Sie folgende Anfragen in relationaler Algebra:

- Geben Sie alle Hausaufgabenergebnisse von Lisa Weiss aus (Übungsnummer und Punkte).
- Wer hat auf eine Hausaufgabe volle Punktzahl (Vorname, Nachname und Aufgabennummer)?

Dies bezieht sich auf das Schema auf Folie 4:

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

Selbstverbund (1)

- Manchmal ist es notwendig, sich auf mehr als ein Tupel einer Relation gleichzeitig zu beziehen.
- Z.B. Wer hat auf irgendeine Übung mindestens so viele Punkte wie die Studentin 101?
- In diesem Fall benötigt man zwei Tupel der Relation `BEWERTUNGEN`, um ein Ergebnistupel zu berechnen:
  - Ein Tupel für die Studentin 101.
  - Ein Tupel der gleichen Übung, bei dem `PUNKTE` mindestens so groß wie in dem ersten Tupel ist.

Selbstverbund (2)

- Dies erfordert eine Verallgemeinerung des obigen Anfragemusters,
  in dem zwei Kopien einer Relation verknüpft werden (mindestens eins muss vorher umbenannt werden).

$$S := \rho_{\texttt{X}}(\texttt{BEWERTUNGEN}) \bowtie_{\substack{\texttt{X.ATYP} = \texttt{Y.ATYP} \\ \wedge\, \texttt{X.ANR} = \texttt{Y.ANR}}} \rho_{\texttt{Y}}(\texttt{BEWERTUNGEN});$$

$$\pi_{\texttt{X.SID}}\left(\sigma_{\texttt{X.PUNKTE} \geq \texttt{Y.PUNKTE} \wedge \texttt{Y.SID} = 101}(S)\right)$$

- Diese Verbunde einer Tabelle mit sich selbst werden „Selbstverbund“ („self join“) genannt.

Inhalt

Mengenoperationen (1)

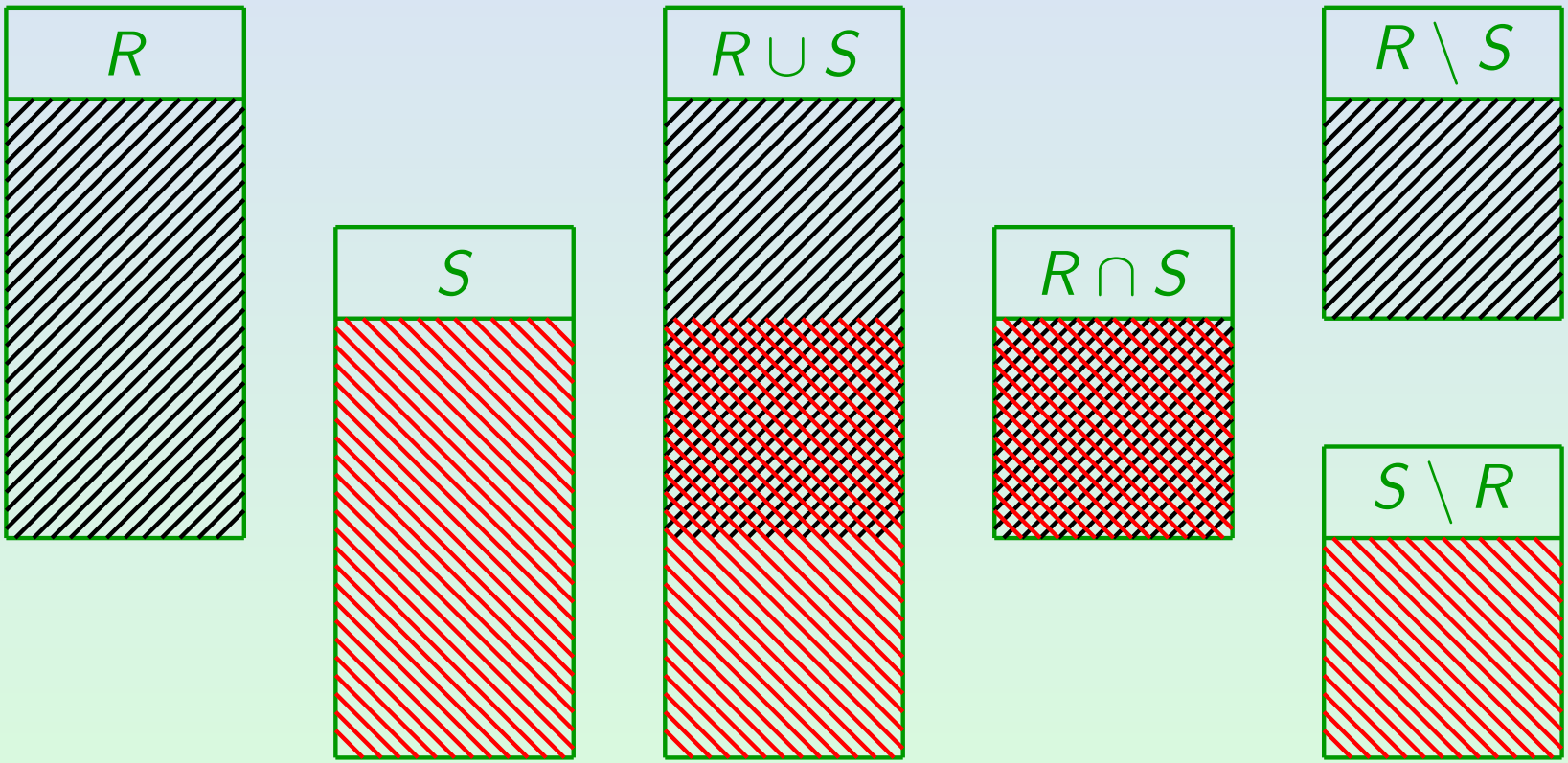
- Da Relationen Mengen (von Tupeln) sind, können auch die üblichen Mengenoperationen $\cup$, $\cap$, $\setminus$ auf Relationen angewandt werden.

  Hierbei müssen beide Eingaberelationen das gleiche Schema haben. Z.B. ist es nicht möglich, die zwei Relationen $R(A)$ und $S(B, C)$ zu vereinigen, da es kein gemeinsames Schema für die Ausgaberelation gibt.
  Die Übungssoftware RA hat die Verknüpfungen `\union`, `\diff` und `\intersect`.
  In RelaX schreibt man entsprechend `union`, `\` oder `except` und `intersect`.

- $R \cup S$ enthält alle Tupel, die in R, in S, oder in beiden Relationen enthalten sind (Vereinigung).
- $R \setminus S$ enthält alle Tupel, die in R, aber nicht in S sind (Mengendifferenz).
- $R \cap S$ enthält Tupel, die in beiden, R und S, sind (Schnitt).

  Der Schnitt ist eine abgeleitete Operation: $R \cap S = R \setminus (R \setminus S)$.

Mengenoperationen (2)



Mengenoperationen (3)

- $R \cup S$ kann wie folgt implementiert werden:

```
(1)  Create new temporary relation T;
(2)  foreach tuple t in R do
(3)      insert t into T;
(4)  od;
(5)  foreach tuple t in S do
(6)      insert t into T;
(7)  od;
(8)  return T;
```

- **insert** muss dabei eventuell Duplikate eliminieren.

  In SQL gibt es `UNION` (mit Duplikatelimination) und `UNION ALL` (ohne Duplikatelimination, läuft schneller).

Mengenoperationen (4)

- $R \setminus S$ kann wie folgt implementiert werden:

```
(1)  Create new temporary relation T;
(2)  foreach tuple t in R do
(3)      Remove := false;
(4)      foreach tuple u in S do
(5)          if u = t then
(6)              Remove := true;
(7)      od;
(8)      if not Remove then
(9)          insert t into T;
(10) od;
(11) return T;
```

Vereinigung (1)

- Ohne $\cup$ kann jede Ausgabespalte nur Werte einer einzigen Spalte der gespeicherten Tabellen enthalten.

  Oder eine einzelne Konstante. Wenn Datentyp-Operationen in der Projektion erlaubt sind, kann der Ergebniswert mit einer einzelnen Formel aus verschiedenen Eingabespalten berechnet werden, aber selbst dies ist noch kein „Union“-Verhalten.

- Beispiel: Neben registrierten Studenten, die Hausaufgaben abgeben und Prüfungen machen, gibt es auch Gasthörer, die sich nur die Vorlesung anhören:

  GASTHÖRER(VORNAME, NACHNAME, EMAIL°).

- Aufgabe: Erstellung einer Liste der Email-Adressen von Studenten und Gasthörern in einer Anfrage.

Vereinigung (2)

- Mit $\cup$ ist dies einfach:

$$\pi_{\texttt{EMAIL}}(\texttt{STUDENTEN}) \cup \pi_{\texttt{EMAIL}}(\texttt{GASTHÖRER}).$$

- Diese Anfrage kann nicht ohne $\cup$ formuliert werden.
- Fallunterscheidungen sind auch typische Anwendungen von $\cup$:

$$\texttt{ZPT} := \pi_{\texttt{SID,PUNKTE}}\Big(\sigma_{\texttt{ATYP='Z'} \wedge \texttt{ANR=1}}(\texttt{BEWERTUNGEN})\Big);$$

$$\begin{aligned}
&\pi_{\texttt{SID, NOTE} \leftarrow \texttt{'1'}}\Big(\sigma_{\texttt{PUNKTE} \geq 12}(\texttt{ZPT})\Big) \\
\cup\ &\pi_{\texttt{SID, NOTE} \leftarrow \texttt{'2'}}\Big(\sigma_{\texttt{PUNKTE} \geq 10 \wedge \texttt{PUNKTE} < 12}(\texttt{ZPT})\Big) \\
\cup\ &\pi_{\texttt{SID, NOTE} \leftarrow \texttt{'3'}}\Big(\sigma_{\texttt{PUNKTE} \geq 7 \ \wedge \texttt{PUNKTE} < 10}(\texttt{ZPT})\Big) \\
\cup\ &\pi_{\texttt{SID, NOTE} \leftarrow \texttt{'5'}}\Big(\sigma_{\texttt{PUNKTE} < 7}\ (\texttt{ZPT})\Big)
\end{aligned}$$

Mengendifferenz (1)

- Die Operatoren σ, π, $\times$, $\bowtie$, $\cup$ haben ein monotones Verhalten, z.B.

$$R \subseteq S \implies \sigma_F(R) \subseteq \sigma_F(S)$$

- Damit folgt, dass sich auch jede Anfrage Q, die obige Operatoren verwendet, monoton verhält:
  - Sei $\mathcal{I}_1$ ein DB-Zustand, und resultiere $\mathcal{I}_2$ aus $\mathcal{I}_1$ durch Einfügen von einem oder mehreren Tupeln.
  - Dann ist jedes als Antwort auf Q in $\mathcal{I}_1$ enthaltene Tupel t, auch als Antwort auf Q in $\mathcal{I}_2$ enthalten.

    D.h. korrekte Antworten bleiben auch nach Einfügungen gültig.

- Wenn sich die gewünschte Anfrage nichtmonoton verhält, so folgt, dass man die Mengendifferenz „\“ verwenden muss.

Mengendifferenz (2)

- Z.B. welcher Student hat noch keine Übung gelöst?

  $\texttt{KEINE\_LÖS} := \pi_{\texttt{SID}}(\texttt{STUDENTEN}) \setminus \pi_{\texttt{SID}}(\texttt{BEWERTUNGEN});$

  $\pi_{\texttt{VORNAME, NACHNAME}}(\texttt{STUDENTEN} \bowtie \texttt{KEINE\_LÖS})$

- Wenn man $\setminus$ verwendet, ist der Anti-Join ein typisches Muster. Er wird gelegentlich $\overline{\ltimes}$ geschrieben.
  - Beispiel: Die Tupel aus $R(A, B)$, die keinen Verbundpartner in $S(B, C)$ haben, können wie folgt berechnet werden:

    $$R \bowtie (\pi_B(R) \setminus \pi_B(S)).$$

  - Folgendes ist äquivalent: $R \setminus \pi_{A,B}(R \bowtie S)$.

    Man muss beachten, dass die Mengendifferenz gleiche Schemata auf beiden Seiten erfordert. Dazu benötigt man Projektion und Verbund.

Aufgaben zur Mengendifferenz

Schreiben Sie folgende Anfragen in relationaler Algebra:

- Wer hat die meisten Punkte für Hausaufgabe 1?

  Hinweis: Berechnen Sie zunächst die Studenten, die nicht die meisten Punkte haben, d.h. für die es einen Studenten mit mehr Punkten gibt. Verwenden Sie dann die Mengendifferenz.

- Wer hat alle Übungen der DB gelöst?

Dies bezieht sich auf das Schema auf Folie 4:

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

Vereinigung vs. Verbund

- Zwei alternative Darstellungen der Punkte für HA und Zwischen- und Endklausur der Studenten sind:

Resultate_1

STUDENT	H	Z	E
Jim Ford	95	60	75
Ann Lloyd	80	90	95

Resultate_2

STUDENT	ATYP	PROZENT
Jim Ford	H	95
Jim Ford	Z	60
Jim Ford	E	75
Ann Lloyd	H	80
Ann Lloyd	Z	90
Ann Lloyd	E	95

- Geben Sie Algebraausdrücke an, um die Tabellen in einander zu überführen.

Zusammenfassung

Die fünf Basisoperationen der relationalen Algebra sind:

- σ_F: Selektion
- $\pi_{A_1,\ldots,A_k}$: Projektion
- $\times$: Kartesisches Produkt
- $\cup$: Vereinigung
- $\setminus$: Mengendifferenz

Abgeleitete Operationen sind u.a.:

- $\bowtie_F$, $\bowtie$: Allgemeiner und natürlicher Verbund
- ρ: Umbenennung
- $\cap$: Schnitt.

Inhalt

Äußerer Verbund (1)

- Der gewöhnliche Verbund („innerer Verbund“) eliminiert Tupel ohne Verbundpartner:

A	B
a_1	b_1
a_2	b_2

$\bowtie$

B	C
b_2	c_2
b_3	c_3

$=$

A	B	C
a_2	b_2	c_2

- Der linke äußere Verbund stellt sicher, dass Tupel der linken Tabelle auch im Ergebnis vorhanden sind:

A	B
a_1	b_1
a_2	b_2

$\overset{\leftarrow}{\bowtie}$

B	C
b_2	c_2
b_3	c_3

$=$

A	B	C
a_1	b_1	
a_2	b_2	c_2

Zeilen der linken Seite werden, falls notwendig, mit „Null“ aufgefüllt.

In RelaX kann man `left join` oder `left outer join` schreiben. Man kann danach eine Bedingung angeben oder nicht (dann natürlicher äußerer Verbund).

Äußerer Verbund (2)

- Der rechte äußere Verbund erhält die Tupel der rechten Tabelle:

A	B
a_1	b_1
a_2	b_2

$\bowtie$

B	C
b_2	c_2
b_3	c_3

$=$

A	B	C
a_2	b_2	c_2
	b_3	c_3

RelaX: `right join` oder `right outer join`.

- Der volle äußere Verbund eliminiert gar kein Tupel:

A	B
a_1	b_1
a_2	b_2

$\bowtie$

B	C
b_2	c_2
b_3	c_3

$=$

A	B	C
a_1	b_1	
a_2	b_2	c_2
	b_3	c_3

RelaX: `full outer join`.

Äußerer Verbund (3)

$R \overset{\supset\bowtie}{\underset{A=B}{}} S$:

```
(1)  Create new temporary relation T;
(2)  foreach tuple t in R do
(3)      HasJoinPartner := false;
(4)      foreach tuple u in S do
(5)          if t.A = u.B then
(6)              insert t ∘ u into T;
(7)              HasJoinPartner := true;
(8)          fi;
(9)      od;
(10)     if not HasJoinPartner then
(11)         insert t ∘ (null,...,null) into T;
(12) od;
(13) return T;
```

Äußerer Verbund (4)

- Z.B. Studenten mit ihren Hausaufgabenergebnissen, Studenten ohne Hausaufgabenergebnis werden mit Null-Werten aufgeführt:

$$\texttt{STUDENTEN} \ \overline{\bowtie} \ \pi_{\texttt{SID,ANR,PUNKTE}}\left(\sigma_{\texttt{ATYP}=\texttt{'H'}}(\texttt{BEWERTUNGEN})\right)$$

SID	VORNAME	NACHNAME	EMAIL	ANR	PUNKTE
101	Lisa	Weiss	...	1	10
101	Lisa	Weiss	...	2	8
102	Michael	Grau	NULL	1	9
102	Michael	Grau	NULL	2	9
103	Daniel	Sommer	...	1	5
104	Iris	Winter	...	NULL	NULL

Äußerer Verbund (5)

- Übung: Gibt es irgendeinen Unterschied zwischen
  - STUDENTEN $\bowtie$ BEWERTUNGEN und
  - STUDENTEN $\bowtie\!\!\!\!\subset$ BEWERTUNGEN?
- Der äußere Verbund ist vor allem mit Aggregationsfunktionen (z.B. `count`, `sum`) sinnvoll, siehe unten.

  Aggregationsfunktionen werden hier nur für SQL eingeführt.
- Mit einer Selektion auf das Ergebnis des äußeren Verbunds, kann man ihn wie eine Mengendifferenz verwenden: Aber das ist fragwürdiger Stil.

  Früher notwendig in MySQL (hat Unteranfragen erst ab Version 5.1).

  Auch heute noch soll die Lösung mit Outer Join schneller sein.

Äußerer Verbund (6)

- Der äußere Verbund ist eine abgeleitete Operation (wie $\bowtie$, $\cap$), d.h. man kann ihn mit den Basisoperationen der relationalen Algebra darstellen.
- Z.B. seien $R(A, B)$ und $S(B, C)$ zwei Relationen.
- Dann ist der linke äußere Verbund $R \ \overline{\bowtie}\ S$ eine Abkürzung für

$$R \bowtie S \quad \cup \quad (R \setminus \pi_{A,B}(R \bowtie S)) \times \{(C : \mathit{null})\}$$

(wobei $\bowtie$ noch durch $\times$, σ, π ersetzt werden kann).

D.h. der äußere Verbund fügt dem normalen Verbund die Tupel aus R zu, die keinen Partner haben (aufgefüllt mit $C : \mathit{null}$, um dasselbe Schema zu erhalten, da man sonst die Vereinigung nicht anwenden kann).