Unit 0x06

Relationale Algebra II

Operationen der relationalen Algebra

Wiederholung

- Die bisher gezeigten Joins sind sog. *Inner Joins* und SQL hat eigene (äquivalente und optimierte) Befehle dafür (`join`).
- Dies, und auch einen Teil der anderen *Joins*, besprechen wir schon vorab im SQL-Praktikum, damit man in der SQL-Praxis weiter kommt. Die jeweiligen formalen Definitionen folgen dann hier im zweiten Teil der relationalen Algebra.

Operationen der relationalen Algebra

SQL-Einschub `with`

- Um die folgenden Operationen zu verdeutlichen, konstruieren wir bei Bedarf spezielle Relationen mit jeweils 'passenden' Attributen und Werten. Hierzu sehr hilfreich sind sog. 'Common Table Expressions (CTE)' mit `with`.
- Bei CTEs handelt es sich, kurz gesagt, um temporäre Ergebnismengen, die unter einem eigenen Namen in darauf folgenden SQL-Befehlen zur Verfügung stehen.

Beispiel

```
SELECT * FROM produkt;
```

id	bezeichnung	einhei
1	Spinat	PK
2	Vier Käse Pizza	ST
3	Spinatpizza	ST

Bekannte Struktur von `produkt`

```
WITH Prods as (
    SELECT id as 'prod_id',bezeichnung as 'bez' FROM produkt
)
SELECT * FROM Prods WHERE bez like '%pizza%';
```

prod_id	bez
2	Vier Käse Pizza
3	Spinatpizza

Neue Relation `Prods` nur mit Spalten `prod_id` und `bez`, nutzbar in `select`

Operationen der relationalen Algebra

Theta-Verbund/-Join vs Equi-join

- Der Join (Verbund) bezeichnet allgemein die beiden Operationen kartesisches Produkt mit anschliessender Selektion und Selektionsbedingung Θ, daher auch Theta-Verbund:
  - $S \bowtie_{\Theta} T = \sigma_{\Theta}(S \times T)$
- Ein Spezialfall ist der *Equi Join* mit der Bedingung, dass der Inhalt bestimmter Attribute, z.B. a_1 und a_2, identisch sein muss, d.h. der speziellen Form $a_1 = a_2$ genügt:
  - $S \bowtie_{a1=a2} T = \sigma_{a1=a2}(S \times T)$

Operationen der relationalen Algebra

Natürlicher Verbund

- Ausgehend von der Idee, dass Primär- und Fremdschlüssel gleich heissen (mögen), nutzt der natürliche Verbund dies aus und verbindet Entitäten, die in gleich benannten Attributen gleiche Werte besitzen – ein Equi-Join auf gleichen Attributen.
- Im Unterschied zum Theta-Verbund enthält der Natural Join die gleichen Attribute nur einmal, eliminiert so also ungewünschte Redundanz.
- Für zwei Relationen S, T mit $schema(S)=\{[a_1,\dots,a_n,b_1,\dots,b_k]\}$, $schema(T)=\{[b_1,\dots,b_k,c_1,\dots,c_m]\}$ (für die Übersicht ohne Typangabe) ist der natürliche Verbund $S \bowtie T$ definiert durch
  - $S \bowtie T = \Pi_{a1..an,S.b1..S.bk,c1..cm}\ \sigma_{S.b1=T.b1 \wedge \dots \wedge S.bn=T.bn}(S \times T)$
- Das entspricht dem Schema $\{[a_1,\dots,a_n,b_1,\dots,b_k,c_1,\dots,c_m]\}$ (wieder ohne Typangabe, ggf. ergänzen) und man sieht, dass die 'doppelten' Attribute $S._{b1}$ bzw. $T._{b1}$ etc. durch die Projektion Π weggefallen sind.

Operationen der relationalen Algebra

Beispiel Natürlicher Verbund

```
SELECT id as 'wid', bezeichnung as 'wbez' FROM warengruppe;
SELECT id as 'pid', bezeichnung as 'pbez', warengruppe_id as 'wid' FROM produkt;
```

wg

wid	wbez
1	TK
2	Obst, Gemüse
3	Wurst, Aufschnitt
4	Milchprodukte
5	Kaltgetränke

pg

pid	pbez	wid
1	Spinat	1
2	Vier Käse Pizza	1
3	Spinatpizza	1
4	Fischstäbchen	1
5	Nudelpfanne	1

- Vorbereitung der Relationen `produkt` und `warengruppe` zu `wg` bzw. `pg` mit umbenannten Attributen.
  Man sieht, dass das 'neue' gemeinsame Attribut *nur* `wid`, also der Fremdschlüssel der Warengruppe ist und insbesondere auch die Bezeichnung jetzt `wbez` bzw. `pbez`, also *unterschiedlich*, ist! So verwenden wir `wg` und `pg` mit `with` im Beispiel für Join bzw. `Natural Join`.
- Beim Natural Join fällt die 'doppelte' Spalte weg.

```
WITH wg as (SELECT id as 'wid', bezeichnung as 'wbez' FROM
     pd as (SELECT id as 'pid', bezeichnung as 'pbez', war
SELECT * FROM pd JOIN wg on pd.wid=wg.wid;
```

pid	pbez	pd.wid	wg.wid	wbez
1	Spinat	1	1	TK
2	Vier Käse Pizza	1	1	TK
3	Spinatpizza	1	1	TK

```
WITH wg as (SELECT id as 'wid', bezeichnung as 'wbez' FR
     pd as (SELECT id as 'pid', bezeichnung as 'pbez', w
SELECT * FROM pd NATURAL JOIN wg;
```

wid	pid	pbez	wbez
1	1	Spinat	TK
1	2	Vier Käse Pizza	TK
1	3	Spinatpizza	TK

Operationen der relationalen Algebra

Anmerkungen Natürlicher Verbund

- Von der Verwendung von *Natural Joins* wird abgeraten!
- Beim *Natural Join* werden immer *alle* gleichnamigen Attribute verwendet, d.h. Hinzufügen neuer Attribute oder Umbenennen vorhandener Attribute führt schnell zu einer Änderung der Abfrage!
- *Achtung:* Erweitern einer Tabelle ist nichts ungewöhnliches und passiert ggf. sogar automatisiert, wenn z.B. die zu persistierende Objektstruktur erweitert wird und die Tabellen die Daten widerspiegeln.

Beispiel

- Hier wurden Bezeichnungen gleich gewählt. Damit muss `wid` *und* `bez` für den *Natural Join* übereinstimmen… Oops.

```
WITH wg as (SELECT id as 'wid', bezeichnung as 'bez' FROM warengruppe),
     pd as (SELECT id as 'pid', bezeichnung as 'bez', warengruppe_id as 'wid' FROM produkt)
SELECT * FROM pd NATURAL JOIN wg;
```

wid	pid

Operationen der relationalen Algebra

Semi-Join bzw. Halbverbund

- Manchmal interessiert nur die Existenz, nicht aber die Attributwerte der assoziierten Entität beim (Natural) Join ➜ Halbverbund bzw. Semi-Join.
- Der Halbverbund $R \ltimes S$ ist für zwei Relationen S und T definiert durch:
  - $S \ltimes T = \Pi_{ident(S)} (S \bowtie T)$

Beispiel

- Projiziert man das Beispiel des *Natural Joins* zuvor auf die Attribute von `pd`, so ergibt das den Halbverbund.

```
WITH wg as (SELECT id as 'wid', bezeichnung as 'wbez' FROM warengruppe),
     pd as (SELECT id as 'pid', bezeichnung as 'pbez', warengruppe_id as 'wid' FROM produkt)
SELECT pd.pid,pd.pbez,pd.wid FROM pd NATURAL JOIN wg;
```

pid	pbez	wid
1	Spinat	1
2	Vier Käse Pizza	1
3	Spinatpizza	1
4	Fischstäbchen	1

Operationen der relationalen Algebra

Anti-Semi-Join

- Beim Anti-Semi-Join $S \triangleright T$ zweier Relationen S und T werden die Tupel aus S selektiert, die am natürlichen Verbund *nicht* teilnehmen:
  - $S \triangleright T = S - S \ltimes T = S - \boldsymbol{\Pi}_{ident(S)} (S \bowtie T)$

Beispiel

- Folgt noch, da wir den Minus-Operator benötigen, der wiederum in MySQL nicht definiert ist aber über einen Outer Join abbildbar ist.

Operationen der relationalen Algebra

Outer Join

- 'Zusammengehörige' Daten *und Daten, zu denen kein Pendant beim Join existiert*, abfragen. Das sind: *Left Outer Join*, *Right Outer Join* und *Full Outer Join*.
- Bei einem *Left Outer Join* zweier Relationen S und T, in Zeichen
  - $S ⟕ T$

  werden *alle* Entitäten der Entitätenmenge *links* der Relation, also S, berücksichtigt, auch wenn es keine zugehörigen Entitäten in Entitätenmenge T gibt. Diese Attribute sind dann `NULL`.
- Bei einem *Right Outer Join*, in Zeichen
  - $S ⟖ T$
- gilt das analog, nur mit vertauschten Rollen. D.h. es werden alle Entitäten der *rechten* Relation T berücksichtigt, auch wenn es keine zugehörigen Entitäten in S gibt.

Operationen der relationalen Algebra

Outer Join

- Der *Full Outer Join* ist die Vereinigung von *Left* und *Right Outer Join*. Das bedeutet, es sind alle Entitäten beider Seiten dabei, nur ggf. mit `NULL`-Einträgen in den Attributen der anderen Seite, wenn es keine zugehörige Entität gibt. In Zeichen
  - $S ⟗ T$
- Bei den *Outer Joins* sind nicht nur *Natural Joins*, sondern allgemein Theta-Verbünde gemeint. Im Gegensatz zu *Outer Joins* spricht man auch explizit von *Inner Joins*.

Beispiel Outer Join

- Tiere mit und ohne 'Besitzer' aus der Menge der Mitarbeiter.
- Zur Erinnerung: 'Rosa' ist beim *Inner Join* nicht dabei:

id	name	mitarbeiter_id
1	Petra	4
2	Mini	18
3	Rosa	<null>

```
SELECT T.name, M.name
FROM tier T INNER JOIN mitarbeiter M
ON M.id = T.mitarbeiter_id;
```

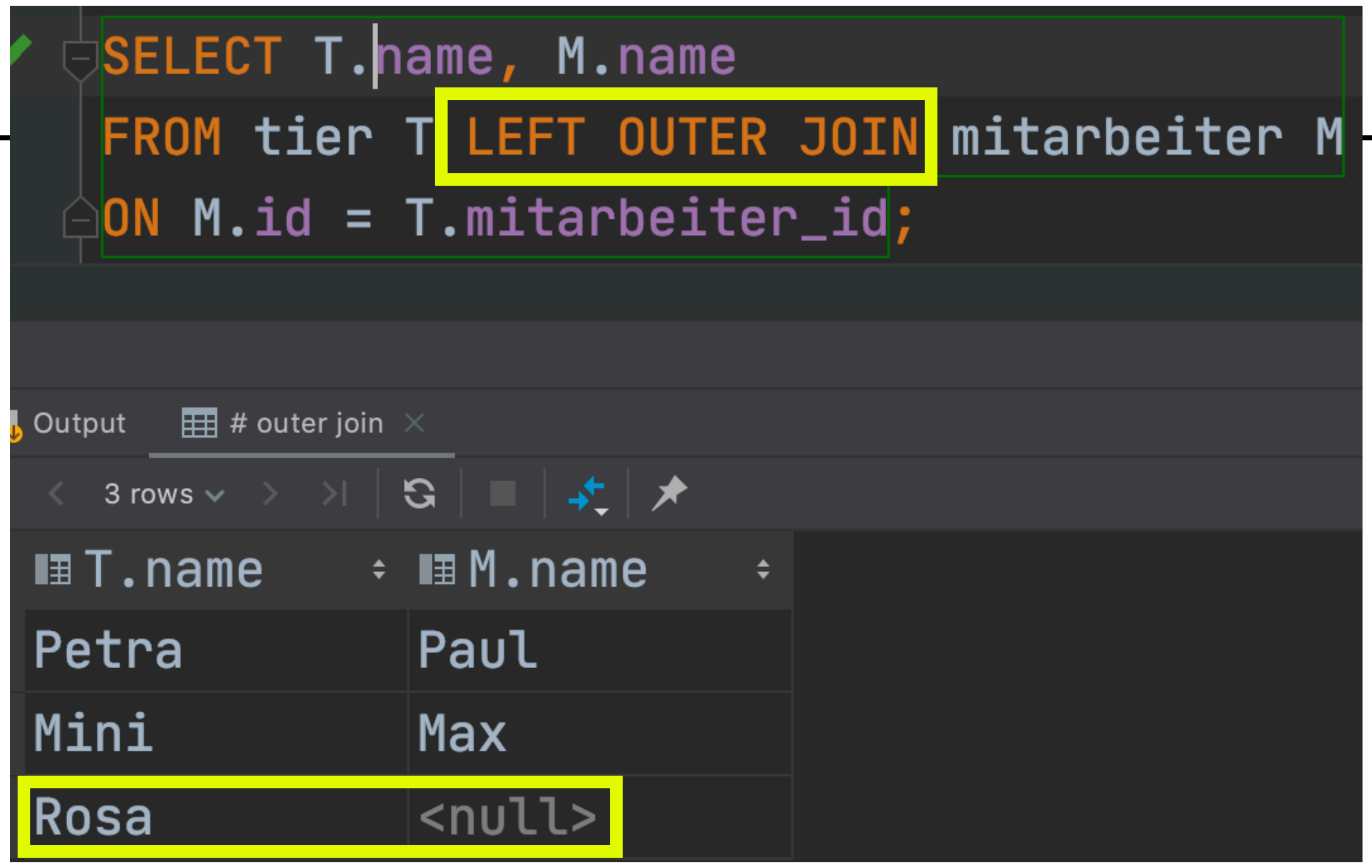
T.name	M.name
Petra	Paul
Mini	Max

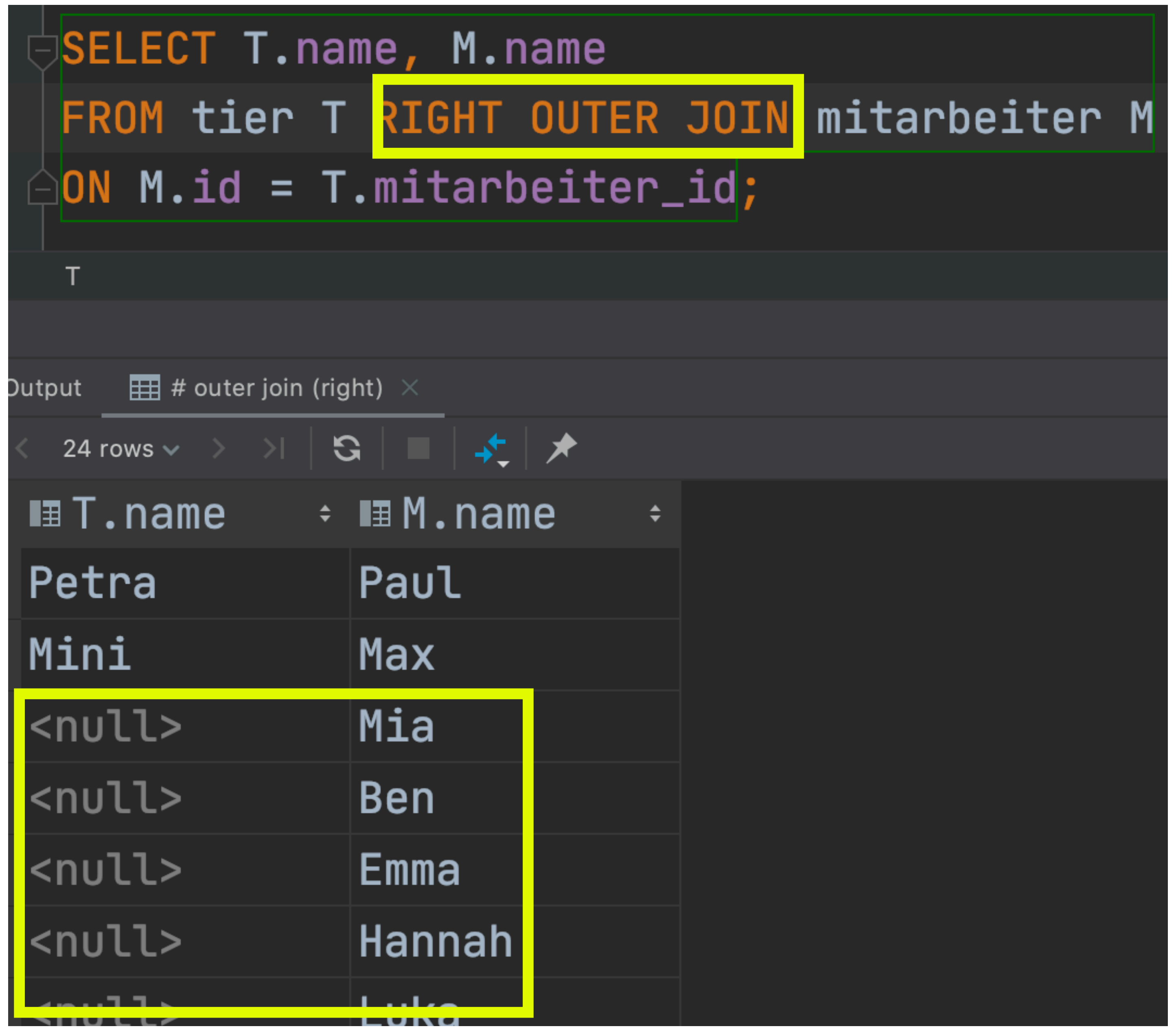
Beispiele `matse_mhist`

Operationen der relationalen Algebra

Beispiel Outer Join Fortsetzung

- Beim *Left Outer Join* ist 'Rosa' dabei, da *alle* Entitäten des Typs Tier (links des Joins) berücksichtigt werden, auch wenn sie nicht in Relation stehen.
  Die zugehörigen Attribute des (nicht vorhandenen) Partners sind `NULL`.
- Betrachten wir das gleiche SQL-Kommando als *Right Outer Join*, so kommen *alle* Mitarbeiter (rechte Seite des Joins) vor und die zugehörigen Attribute sind `NULL`, wenn der Mitarbeiter nicht in Relation zu einem Tier steht.
  Hier ist 'Rosa' nicht dabei!





Beispiele `matse_mhist`

Operationen der relationalen Algebra

Beispiel Outer Join Fortsetzung

- Der Full Outer Join ist die Vereinigung von *Left* und *Right Outer Join*. Das bedeutet, es sind alle Entitäten beider Seiten dabei, nur ggf. mit `NULL`-Einträgen.
- Manche DBMS haben keinen Befehl für einen *Full Outer Join*, dann vereinigt man die Ergebnisse der beiden *Outer Joins* einfach mit `union`.
- Vertauschen von Entitätstypen *und Left* und *Right Outer Joins*, ergibt wieder den ersten Outer Join.

```sql
SELECT T.name, M.name
FROM mitarbeiter M RIGHT OUTER JOIN tier T
ON M.id = T.mitarbeiter_id;
```

Output | # change left and right in join

3 rows

T.name	M.name
Petra	Paul
Mini	Max
Rosa	<null>

```sql
SELECT T.name, M.name
FROM tier T LEFT OUTER JOIN mitarbeiter M
ON M.id = T.mitarbeiter_id
UNION
SELECT T.name, M.name
FROM tier T RIGHT OUTER JOIN mitarbeiter M
ON M.id = T.mitarbeiter_id;
```

Output | Result 82

25 rows

T.name	M.name
Petra	Paul
Mini	Max
Rosa	<null>
<null>	Mia
<null>	Ben
<null>	Emma
<null>	Hannah

Beispiele `matse_mhist`

Operationen der relationalen Algebra

Anmerkungen Inner / Outer Join

- *Nur die* Kunden, zu denen eine Bestellung *existiert*, d.h. die, die in Relation stehen – somit *Inner Join*:
- Kunden, zu denen eine Bestellung existiert und die, die noch keine Bestellung aufgegeben haben, d.h. faktisch *alle* Kunden, mit und ohne Partner – somit *Outer Join*:
  - Der Entitätstyp `kunde` steht *rechts* am *Join*, daher hier *Right Outer Join*.
- Da beim *Inner Join* nur Entitäten teilnehmen, die einen Partner haben, gibt es keinen *Left* oder *Right Inner Join*.

```sql
SELECT B.id, B.kunde_id, K.id, K.name
FROM bestellung B INNER JOIN kunde K
ON B.kunde_id=K.id;
```

B.id	kunde_id	K.id	name
1	30	30	Sparkasse
2	30	30	Sparkasse
3	32	32	E.ON

```sql
SELECT B.id, B.kunde_id, K.id, K.name
FROM bestellung B RIGHT OUTER JOIN kunde K
ON B.kunde_id=K.id;
```

B.id	kunde_id	K.id	name
<null>	<null>	28	Daimler
<null>	<null>	29	Deutsche Bank
1	30	30	Sparkasse
2	30	30	Sparkasse
<null>	<null>	31	Börse
3	32	32	E.ON
<null>	<null>	33	Infinion

Beispiele `matse_mhist`

Operationen der relationalen Algebra

Differenz/Minus, Schnittmenge, Symmetrische Differenz

- Diese beiden Operationen sind beispielsweise in MySQL bzw. MariaDB nicht vorhanden, können über Joins aber leicht abgebildet werden.
- Für zwei Relationen S und T mit schema(S)=schema(T) und Schlüsselattribut `K` gilt:
  - S - T: `SELECT S.* FROM S LEFT OUTER JOIN T USING(K) WHERE isnull(T.K)`
  - S ∩ T: *`SELECT S.* FROM S JOIN T USING(K);`*
- Die Symmetrische Differenz ist somit auch definiert:
  - $S \Delta T = (S - T) \cup (T - S)$

Operationen der relationalen Algebra

Umbenennung von Relationen oder Attributen

- *Motivation:* Bei einem Join oder kartesischen Produkt kommt es zuweil vor, dass in der Ergebnisrelation eigentlich Attribute gleich heissen würden – was mathematisch und technisch ein Problem ist. Analog kann es notwendig sein, ganzen Relationen einen eigenen Namen zu geben, um etwa bei einem Self-Join diese zu unterscheiden.
- *Idee:* Umbenennen einer Relation S zu S' mittels:
  - $R[S \rightarrow S']$ *oder* $\rho_{S'}(S)$
- *u*nd Umbenennen eines Attributes a zu a' einer Relation T:
  - $\rho_{a \rightarrow a'}(T)$

Beispiel

- $\rho_{id \rightarrow no, bezeichnung \rightarrow bez}(\rho_{pr}(\mathtt{produkt}))$

```
SELECT pr.id as 'no', pr.bezeichnung as 'bez'
FROM produkt pr;
```

no	bez
1	Spinat
2	Vier Käse Pizza
3	Spinatpizza
4	Fischstäbchen

Operationen der relationalen Algebra

Division

- *Hintergrund:* Fragen, in denen es um 'für alle' (Allquantor ∀) geht.
- Für zwei Relationen S und T mit schema(S)={[$a_1,\dots,a_n,b_1,\dots,b_k$]}, schema(T)={[$b_1,\dots,b_k$]} (hier ohne Typangabe) ist die Division S ÷ T definiert durch
  - $S \div T = \{ (a_1,\dots,a_n) \mid \forall (b_1,\dots,b_k) \in T : (a_1,\dots,a_n,b_1,\dots,b_k) \in S \}$, oder
  - $S \div T = \Pi_{(S\text{-}T)}(S) - \Pi_{(S\text{-}T)}((\Pi_{(S\text{-}T)}(S) \times T) - S)$
- Division ÷ ist 'Umkehroperation' zum kartesischen Produkt, denn es gilt (symbolisch):
  - $(S \times T) \div T = S$
- *Achtung:* Ergebnismenge ist nur der 'a'-Anteil von S, also $\Pi_a(S)$

Operationen der relationalen Algebra

Beispiel Foto-Weihnachten

- Bestimme alle Fotos (Datum, Zeit), auf denen 'Vadder' und 'Mutti' zu sehen sind. Oder anders formuliert:
- Bestimme alle Fotos einer Menge R, für die gilt, dass *alle* Personen einer Teilmenge S (Vadder', 'Mutti') zu sehen sind, d.h. wir suchen R ÷ S, mit R und S gegeben durch

R={[Datum, Zeit, Name]}

Datum	Zeit	Name
24.12.18	09:00	Mutti
24.12.18	09:00	Omma
24.12.18	09:00	Vadder
24.12.18	10:00	Vadder
25.12.18	12:00	Mutti
25.12.18	12:00	Filius
25.12.18	12:00	Vadder
26.12.18	15:00	Omma

S={[Name]}

Name
Mutti
Vadder

Operationen der relationalen Algebra

Fortsetzung Beispiel Foto-Weihnachten

- Die Division (siehe Definition) kann man schrittweise durchführen

  $R \div S = \Pi_{(R-S)}(R) - \Pi_{(R-S)}((\Pi_{(R-S)}(R) \times S) - R)$

$\pi_{R-S}(R)$

Datum	Zeit
24.12.18	09:00
24.12.18	10:00
25.12.18	12:00
26.12.18	15:00

$\pi_{R-S}(R) \times S$

Datum	Zeit	Name
24.12.18	09:00	Mutti
24.12.18	09:00	Vadder
24.12.18	10:00	Mutti
24.12.18	10:00	Vadder
25.12.18	12:00	Mutti
25.12.18	12:00	Vadder
26.12.18	15:00	Mutti
26.12.18	15:00	Vadder

Operationen der relationalen Algebra

Fortsetzung Beispiel Foto-Weihnachten

- R ÷ S = $\Pi_{(R\text{-}S)}(R) - \Pi_{(R\text{-}S)}((\Pi_{(R\text{-}S)}(R) \times S) - R)$

$\pi_{R-S}(R) \times S - R$

Datum	Zeit	Name
24.12.18	10:00	Mutti
26.12.18	15:00	Mutti
26.12.18	15:00	Vadder

$\pi_{R-S}(\pi_{R-S}(R) \times S - R)$

Datum	Zeit
24.12.18	10:00
26.12.18	15:00

$\pi_{R-S}(R) - \pi_{R-S}(\pi_{R-S}(R) \times S - R)$

Datum	Zeit
24.12.18	09:00
25.12.18	12:00

$= R \div S$

Operationen der relationalen Algebra

Fortsetzung Beispiel Foto-Weihnachten

R

Datum	Zeit	Name
24.12.18	09:00	Mutti
24.12.18	09:00	Omma
24.12.18	09:00	Vadder
24.12.18	10:00	Vadder
25.12.18	12:00	Mutti
25.12.18	12:00	Filius
25.12.18	12:00	Vadder
26.12.18	15:00	Omma

S

Name
Mutti
Vadder

$R \div S$

Datum	Zeit
24.12.18	09:00
25.12.18	12:00

$R \div S$ sind alle die Datensätze, die mit allen Daten aus S kombiniert vorkommen – aber nur die Attribute R-S !

- *Alternativ (überschaubare Datenmenge):* Es sind alle Datensätze in R gesucht, die mit *allen* S vorkommen…

Operationen der relationalen Algebra

Aggregation und Gruppierung

- *Hintergrund:* Gruppierung von Tupeln, d.h. Tupel mit identischen Werten in Attributen $a_1,\ldots,a_n$ (Gruppierungsattribute) werden zu einer Gruppe zusammengefasst und ggf. Aggregatfunktionen $f_1,\ldots,f_k$ angewendet. Notation:
  - $\gamma_{a1,\ldots,an;\ f1,\ldots,fk}(S)$
- Jede Aggregatfunktion $f_1,\ldots,f_k$ ergibt eine Spalte in der Ergebnistabelle.
- Aggregatfunktion sind z.B. `count`, `avg`, `sum`, `min`, `max`, siehe SQL-Praktikum.

Beispiel

- Gruppiere Studierende nach Semesterzahl und zähle sie pro Gruppe

  $\gamma_{\text{Semetser; count(*)}}$(`studenten`)

Übersicht Operationen der relationalen Algebra

Grundoperationen

- Vereinigung ✔
- Differenz ✔
- Kartesisches Produkt ✔
- Selektion ✔
- Projektion ✔
- Umbenennung ✔

Erweiterte Grundfunktionen

- Gruppierung / Aggregation ✔

Keine Grundoperationen

- Schnittmenge ✔
- Symmetrische Differenz ✔
- Theta-Verbund, Inner Join ✔
- Equi-Join ✔
- Natural Join ✔
- Semi Join ✔
- Anti-Semi-Join ✔
- Outer Join ✔
- Division ✔

Übersicht Operationen der relationalen Algebra

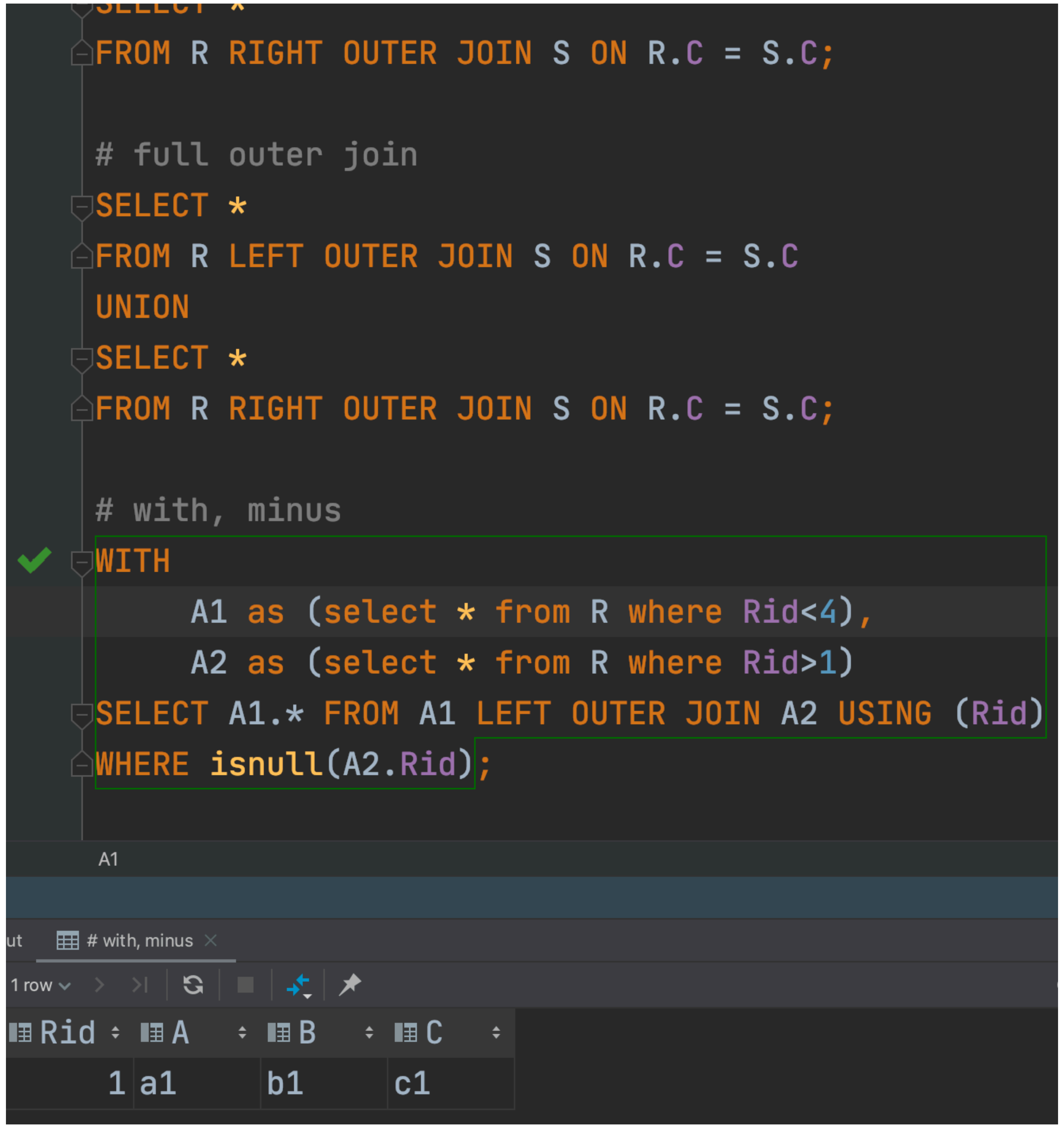
SQL-Beispiele

- Synthetische Mengen R und S, wobei C ein gemeinsames Attribut ist und S.FB einen Fremd-schlüssel C aus R darstellen soll.

Rid	A	B	C
1	a1	b1	c1
2	a2	b2	c2
3	a3	b3	c3
4	a4	b4	<null>

Sid	C	D	E	FB
11	c1	d1	e1	b1
13	c3	d3	e3	b3
14	c4	d4	e4	b4

- Die Beispiele nutzen zum Teil `with` zur Zusammenstellung geeigneter Teilmengen.



Auszug SQL,
Schema `matse_algebra`

Alternativen zur relationalen Algebra

Tupelkalkül

- Idee: Ergebnis einer Anfrage wird als Menge von Tupeln beschrieben, die einer prädikatenlogischen Formel ψ entsprechen (wie bei mathematischen Mengen)
  - $\{ s \in S \mid \psi(s) \text{ wahr} \}$,

  die wiederum aus sog. Atomen a, a Attribut in S, zusammengesetzt ist.
  Beispiel: { s ∈ studenten | s.Semester>5 }

Domänenkalkül

- Grundidee wie zuvor, aber Variablen stehen hier für Tupelkomponente, d.h.
  - $\{ [a_1,\ldots,a_n] \in S \mid \psi(a_1,\ldots,a_n) \text{ wahr} \}$,

  Beispiel: { [matrnr, name, sem] ∈ studenten | sem>5 }

Drei Sprachen sind gleich mächtig, aber nicht Turing-vollständig.