
# Die relationale Algebra

# 4

Die relationale Algebra kann als Abfragesprache des Relationenmodells verstanden werden. Sie geht von einigen Grundoperationen (Vereinigung, Mengendifferenz, kartesisches Produkt, Projektion, Selektion und Umbenennung) aus, welche aus einer oder zwei gegebenen Relationen eine neue Relation berechnen. Aus diesen Grundoperationen können komplexe Ausdrücke geformt werden, welche komplexe Berechnungen von Relationen beschreiben. Diese Ausdrücke entsprechen komplexen Abfragen einer Datenbank und geben (auf einer abstrakten Ebene) vor, wie das Resultat der Abfrage berechnet werden kann.

In diesem Kapitel führen wir die relationale Algebra formal ein. Wir zeigen dann, wie in der Sprache der relationalen Algebra verschiedene Join-Operationen und die relationale Division definiert werden können. Mit einer Reihe von konkreten Beispielen illustrieren wir, wie Datenbankabfragen als Ausdrücke der relationalen Algebra formuliert werden können. Zum Schluss erweitern wir die relationale Algebra noch mit Gruppierungsoperationen und Aggregatsfunktionen.

## 4.1 Basisrelationen

Die relationale Algebra bezieht sich auf ein gegebenes DB-Schema, welches die Basisrelationen der relationalen Algebra festlegt. Mit den Operationen der relationalen Algebra können wir dann aus diesen Basisrelationen komplexe Ausdrücke aufbauen, welche neue Relationen definieren.



T. Studer, *Relationale Datenbanken*,
https://doi.org/10.1007/978-3-662-58976-2_4

### Input Relationen

Sei $\mathscr{S}$ ein Schema des gegebenen DB-Schemas und sei $R$ eine Instanz von $\mathscr{S}$. Dann ist $R$ eine Basisrelation der relationalen Algebra.

### Konstante Relationen

Sei $A$ ein Attribut mit Domäne $D$, welches im gegebenen DB-Schema vorkommt. Weiter sei $a$ eine Element von $D$. Dann ist die konstante 1-stellige Relation $\{(a)\}$ eine Basisrelation der relationalen Algebra. Offensichtlich ist die Relation $\{(a)\}$ eine Instanz des Schemas $(A)$.

*Anmerkung 4.1.* Wir erinnern uns, dass der Wert `Null` zu jeder Domäne gehört. Somit gilt für jedes Attribut $A$, dass die konstante Relation $\{(\texttt{Null})\}$ eine Instanz von $(A)$ ist.

## 4.2 Grundoperationen

Die Grundoperationen der relationalen Algebra sind einfache mengentheoretische Operationen auf den Relationen des Relationenmodells.

### Projektion

Die Projektion ist eine einstellige Operation, welche die Input-Relation, vermindert um einige Attributwerte, wieder ausgibt. Wir gehen aus von Attributen $A_1, \ldots, A_n$, dem Relationenschema

$$\mathscr{S} = (A_1, \ldots, A_n)$$

sowie einer Teilmenge $\{A_{i_1}, \ldots, A_{i_m}\}$ von $\{A_1, \ldots, A_n\}$. Ist $R$ eine Instanz von $\mathscr{S}$, so setzen wir

$$\pi_{A_{i_1},\ldots,A_{i_m}}(R) := \big\{(b_1, \ldots, b_m) \mid \text{es gibt ein } a \in R \text{ mit } b_1 \simeq \pi_{i_1}(a) \text{ und } \cdots \text{ und } b_m \simeq \pi_{i_m}(a)\big\}.$$

Das Resultat dieser Projektion ist also eine Instanz des Schemas

$$(A_{i_1}, \ldots, A_{i_m}).$$

Da die erhaltene Relation wieder eine Menge ist, werden etwa auftretende Duplikate selbstverständlich entfernt.

*Beispiel 4.2.* Betrachte eine Relation `besucht`. Diese Relation modelliert, welche Studierenden welche Vorlesungen besuchen. Sie sei wie folgt gegeben:

**besucht**

| MatNr | Vorlesung |
|---|---|
| 1 | Datenbanken |
| 1 | Programmieren |
| 2 | Programmieren |

Für die Projektion $\pi_{\texttt{Vorlesung}}(\texttt{besucht})$ erhalten wir folgende Tabelle:

$\pi_{\texttt{Vorlesung}}(\texttt{besucht})$

| Vorlesung |
|---|
| Datenbanken |
| Programmieren |

Mit einer Projektion können nicht nur Spalten eliminiert werden. Es kann auch die Reihenfolge der Spalten geändert werden und es können Spalten verdoppelt werden.

*Beispiel 4.3.* Betrachte eine Relation `Studierende`, welche Daten zu den Studierenden enthält. Diese sei gegeben durch:

**Studierende**

| MatNr | Name |
|---|---|
| 1 | Ann |
| 2 | Tom |

Für die Projektion $\pi_{\texttt{Name,MatNr,MatNr}}(\texttt{Studierende})$ erhalten wir folgende Tabelle.

$\pi_{\texttt{Name,MatNr,MatNr}}(\texttt{Studierende})$

| Name | MatNr | MatNr |
|---|---|---|
| Ann | 1 | 1 |
| Tom | 2 | 2 |

*Anmerkung 4.4.* Im obigen Beispiel erhalten wir als Resultat des relationalen Ausdrucks eine Tabelle, welche zwei Spalten mit dem Attributnamen `MatNr` enthält. Dies ist zulässig, wenn im weiteren Verlauf nicht auf eine Spalte mit diesem Namen zugegriffen wird. Das heisst, der Ausdruck

$$\pi_{\texttt{Name}}\big(\pi_{\texttt{Name,MatNr,MatNr}}(\texttt{Studierende})\big)$$

ist ein zulässiger relationaler Ausdruck, da der Attributname `Name` der äusseren Projektion eindeutig eine Spalte von $\pi_{\texttt{Name,MatNr,MatNr}}(\texttt{Studierende})$ bezeichnet. Dagegen ist der Ausdruck

$$\pi_{\texttt{MatNr}}\big(\pi_{\texttt{Name,MatNr,MatNr}}(\texttt{Studierende})\big)$$

nicht zulässig, da der Attributname `MatNr` nicht eindeutig eine Spalte des Zwischenresultates bezeichnet.

### Kartesisches Produkt

Wir gehen aus von Attributen $A_1, \ldots, A_m$ und $B_1, \ldots, B_n$, den Relationenschemata

$$\mathscr{R} = (A_1, \ldots, A_m) \text{ und } \mathscr{S} = (B_1, \ldots, B_n),$$

sowie einer Instanz $R$ von $\mathscr{R}$ und einer Instanz $S$ von $\mathscr{S}$. Wie üblich setzen wir

$$R \times S := \{(x_1, \ldots, x_{m+n}) \mid (x_1, \ldots, x_m) \in R \text{ und } (x_{m+1}, \ldots, x_{m+n}) \in S\}.$$

Nun wollen wir das zu $R \times S$ gehörende Relationenschema definieren. Dabei gilt es zu beachten, dass die Mengen $\{A_1, \ldots, A_m\}$ und $\{B_1, \ldots, B_n\}$ gemeinsame Elemente besitzen dürfen. Da wir unterscheiden wollen, ob ein Attribute aus $\mathscr{R}$ stammt oder aus $\mathscr{S}$, wählen wir folgende Definition. Das Relationenschema für das kartesische Produkt $R \times S$ von $R$ und $S$ hat die Form

$$(R.A_1, \ldots, R.A_m, S.B_1, \ldots, S.B_n).$$

Manchmal vereinfachen wir diese Notation, indem wir für diejenigen Attribute $C$, die nur in einem der beiden Schemata $\mathscr{R}$ oder $\mathscr{S}$ vorkommen, lediglich

$$C \text{ anstelle von } R.C, \text{ beziehungsweise } S.C, \tag{4.1}$$

schreiben.

*Beispiel 4.5.* Betrachte die Relationen `Studierende` und `besucht`. Diese modellieren, welche Studierenden welche Vorlesungen besuchen. Die beiden Relationen seien wie folgt gegeben:

| **Studierende** | |
|---|---|
| **MatNr** | **Name** |
| 1 | Ann |
| 2 | Tom |

| **besucht** | |
|---|---|
| **MatNr** | **Vorlesung** |
| 1 | Datenbanken |
| 1 | Programmieren |
| 2 | Programmieren |

Die entsprechenden Relationenschemata sind

$$(\texttt{MatNr}, \texttt{Name}) \quad \text{und} \quad (\texttt{MatrNr}, \texttt{Vorlesung}).$$

Damit hat das zur Relation $\texttt{Studierende} \times \texttt{besucht}$ gehörende Relationenschema die folgende Form:

$$(\texttt{Studierende.MatNr}, \texttt{Studierende.Name}, \texttt{besucht.MatrNr}, \texttt{besucht.Vorlesung}).$$

Falls wir bei der Namensgebung die Vereinfachung gemäss (4.1) vornehmen, so erhalten wir eine Relation über dem Schema

$$(\texttt{Studierende.MatNr}, \texttt{Name}, \texttt{besucht.MatrNr}, \texttt{Vorlesung}).$$

Die Relation $\texttt{Studierende} \times \texttt{besucht}$ selber wird dann durch folgende Tabelle repräsentiert:

| **Studierende.MatNr** | **Name** | **besucht.MatNr** | **Vorlesung** |
|---|---|---|---|
| 1 | Ann | 1 | Datenbanken |
| 1 | Ann | 1 | Programmieren |
| 1 | Ann | 2 | Programmieren |
| 2 | Tom | 1 | Datenbanken |
| 2 | Tom | 1 | Programmieren |
| 2 | Tom | 2 | Programmieren |

*Anmerkung 4.6.* Die konstanten 1-stelligen Relationen gehören zu den Basisrelationen der relationalen Algebra. Mit Hilfe des kartesischen Produktes können wir nun auch konstante $n$-stellige Relationen für beliebige $n$ definieren. Insbesondere werden wir folgende Konstruktion benötigen.

Seien $A_1, \ldots, A_n$ Attribute. Dann gibt es einen Ausdruck der relationalen Algebra für die konstante $n$-stellige Relation

$$\{(\texttt{Null}, \ldots, \texttt{Null})\}$$

über dem Schema $(A_1, \ldots, A_n)$.

In der Tat, aus Anmerkung 4.1 wissen wir, dass für jedes $i$ der Ausdruck $\{(\texttt{Null})\}$ eine konstante 1-stellige Relation über dem Schema $(A_i)$ beschreibt. Somit ist das kartesische Produkt

$$\{(\texttt{Null})\} \times \cdots \times \{(\texttt{Null})\} = \{(\texttt{Null}, \ldots, \texttt{Null})\}$$

eine Relation über $A_1, \ldots, A_n$.

*Anmerkung 4.7.* Genauso wie bei der Projektion ist es auch beim kartesischen Produkt möglich, dass Spaltennamen mehrfach im Schema der Resultatrelation vorkommen können. Dies ist der Fall, wenn das kartesische Produkt einer Relation mit sich selber berechnet wird. Betrachte eine Relation `S` über einem Schema `(A, B)`. Das kartesische Produkt `S` $\times$ `S` ist dann eine Relation über dem Schema

$$(\texttt{S.A}, \texttt{S.B}, \texttt{S.A}, \texttt{S.B}).$$

## Selektion

Die Selektion ist ebenfalls eine einstellige Operation. Sie sortiert aus einer Relation diejenigen Tupel aus, die ein bestimmtes Prädikat erfüllen. Dabei ist ein Prädikat über Attributen $A_1, \ldots, A_n$ folgendermassen aufgebaut:

- Argumente sind konstante Werte und die Attributnamen $A_1, \ldots, A_n$;
- als Vergleichsoperatoren verwenden wir

$$<, \quad \leq, \quad >, \quad \geq, \quad =, \quad \neq;$$

- komplexe Prädikate werden aufgebaut durch die Junktoren

$$\neg \text{ (nicht)}, \quad \vee \text{ (oder)}, \quad \wedge \text{ (und)}.$$

Bei der Auswertung von Prädikaten ist zu beachten, dass wir nicht nur die Wahrheitswerte `true` und `false` zur Verfügung haben, sondern noch einen dritten Wahrheitswert `unknown`. Dieser wird für das Resultat von Vergleichen verwendet, bei denen der

| ¬ (nicht) | |
|---|---|
| true | false |
| false | true |
| unknown | unknown |

| ∨ (oder) | true | false | unknown |
|---|---|---|---|
| true | true | true | true |
| false | true | false | unknown |
| unknown | true | unknown | unknown |

| ∧ (und) | true | false | unknown |
|---|---|---|---|
| true | true | false | unknown |
| false | false | false | false |
| unknown | unknown | false | unknown |

**Abb. 4.1** Wahrheitstafeln für die logischen Junktoren

Wert `Null` involviert ist. Beispielsweise liefert der Vergleich $7 < \texttt{Null}$ das Ergebnis `unknown` und auch $\texttt{Null} = \texttt{Null}$ resultiert in `unknown` (vergleiche (2.1)).

Die Bedeutung der logischen Junktoren ist auf diesen drei Wahrheitswerten durch die Wahrheitstafeln in Abb. 4.1 gegeben. Dieses System der 3-wertigen Logik ist auch unter dem Namen *Kleene*-Logik bekannt.

*Anmerkung 4.8.* Die Definition der logischen Negation ist verträglich mit der Semantik des `Null` Wertes. Insbesondere gilt für alle $a$:

$$\neg(a = \texttt{Null}) \quad \text{ist gleich} \quad a \neq \texttt{Null}.$$

In der Tat, wir haben

$$\neg(a = \texttt{Null}) \quad \text{ist} \quad \neg(\texttt{unknown}) \quad \text{ist} \quad \texttt{unknown}$$

und

$$a \neq \texttt{Null} \quad \text{ist} \quad \texttt{unknown}.$$

Wir können die Selektion nun wie folgt definieren. Gegeben seien wieder Attribute $A_1, \ldots, A_n$, ein Prädikat $\Theta$ über $A_1, \ldots, A_n$ sowie das Relationenschema

$$\mathscr{S} = (A_1, \ldots, A_n).$$

Ist $R$ eine Instanz von $\mathscr{S}$, so ist die Selektion

$$\sigma_\Theta(R)$$

die Menge aller Tupel aus $R$, deren Werte dem Prädikat $\Theta$ den Wahrheitswert `true` geben. Formal setzen wir

$$\sigma_\Theta(R) := \{t \mid t \in R \text{ und } \Theta(t)\}.$$

Damit ist $\sigma_\Theta(R)$ ebenfalls eine Instanz des Schemas $\mathscr{S}$.

*Beispiel 4.9.* Wir arbeiten noch einmal mit der Relation `Autos` aus Beispiel 2.11. Diese ist gegeben durch:

**Autos**

| **Marke** | **Farbe** | **Baujahr** | **FahrerId** |
|---|---|---|---|
| Opel | silber | 2010 | 1 |
| Opel | schwarz | 2010 | 2 |
| VW | rot | 2014 | 2 |
| Audi | schwarz | 2014 | 3 |
| VW | blau | 2015 | - |

Betrachten wir das Prädikat $\Theta$, das durch

$$\Theta \quad :\Longleftrightarrow \quad \texttt{Marke} = \texttt{'Opel'} \vee \texttt{FahrerId} = 2$$

gegeben ist. Die Relation $\sigma_\Theta(\texttt{Autos})$ wird nun durch folgende Tabelle repräsentiert:

**$\sigma_\Theta$(Autos)**

| **Marke** | **Farbe** | **Baujahr** | **FahrerId** |
|---|---|---|---|
| Opel | silber | 2010 | 1 |
| Opel | schwarz | 2010 | 2 |
| VW | rot | 2014 | 2 |

Oft geben wir das Prädikat explizit in der Selektionsoperation an. Das heisst, wir schreiben direkt

$$\sigma_{\texttt{Marke='Opel'} \vee \texttt{FahrerId=2}}(\texttt{Autos})$$

und führen keinen eigenen Namen für das Selektionsprädikat ein.

*Beispiel 4.10.* Wir betrachten noch einmal die Relationen `Studierende` und `besucht`:

| **Studierende** | |
|---|---|
| **MatNr** | **Name** |
| 1 | Ann |
| 2 | Tom |

| **besucht** | |
|---|---|
| **MatNr** | **Vorlesung** |
| 1 | Datenbanken |
| 1 | Programmieren |
| 2 | Programmieren |

Wir haben oben gesehen, dass das kartesische Produkt `Studierende` $\times$ `besucht` alle Kombinationen aus den Tupeln der beiden Relationen berechnet. Mit einer Selektion können wir nun diejenigen Kombinationen heraussuchen, bei denen die Werte der beiden `MatNr` Attribute übereinstimmen. Dazu führen wir folgendes Prädikat ein:

$$\Theta \quad :\Longleftrightarrow \quad \texttt{Studierende.MatNr} = \texttt{besucht.MatNr}.$$

Damit liefert der relationale Ausdruck

$$\sigma_{\Theta}(\texttt{Studierende} \times \texttt{besucht})$$

folgendes Resultat, wobei wiederum `St` für `Studierende` und `be` für `besucht` steht:

| **St.MatNr** | **St.Name** | **be.MatNr** | **be.Vorlesung** |
|---|---|---|---|
| 1 | Ann | 1 | Datenbanken |
| 1 | Ann | 1 | Programmieren |
| 2 | Tom | 2 | Programmieren |

Bei der Selektionsoperation ist es wichtig zu beachten, dass `Null` *nicht nur* für *unbekannter Wert* steht, sondern auch verwendet wird, wenn ein Attribut keinen Wert hat. Dies wird im folgendem Beispiel erläutert.

*Beispiel 4.11.* Wir betrachten nochmals die Tabelle `Autos`. Für das Selektionsprädikat

$$\Theta := \texttt{FahrerId} = 1 \vee \texttt{FahrerId} \neq 1$$

gilt

$$\texttt{Autos} \neq \sigma_{\Theta}(\texttt{Autos}).$$

In der Tat erfüllt das Tupel

$$(\texttt{VW}, \texttt{blau}, 2015, \texttt{Null})$$

das Prädikat $\Theta$ nicht, da sowohl $\texttt{Null} = 1$ als auch $\texttt{Null} \neq 1$ zu `unknown` ausgewertet werden und damit auch $\Theta$ zu `unknown` ausgewertet wird. Dies ist korrekt, wenn wir beachten, dass `Null` für *keinen Wert* stehen kann.

Hätte `Null` nur die Bedeutung *unbekannter Wert*, so müsste dieser unbekannte Wert entweder $= 1$ oder $\neq 1$ sein. Obwohl wir nicht wissen, welcher dieser beiden Fälle gilt, so wissen wir doch, dass einer von beiden gelten muss, und damit müsste $\Theta$ zu `true` evaluiert werden.

## Umbenennung

Durch Verknüpfung der bisher eingeführten Operationen erhalten wir Ausdrücke der relationalen Algebra. Mit diesen können aus den Basisrelationen neue Relationen berechnet werden. Diesen berechneten Relationen haben wir implizit immer auch ein Schema zugeordnet. Mit Hilfe der Umbenennungsoperation $\rho$ können wir nun die Attribute dieses Schemas umbenennen.

Ist $E$ ein Ausdruck der relationalen Algebra, der eine $n$-stellige Relation beschreibt, so liefert der Ausdruck

$$\rho_{A_1,\ldots,A_n}(E)$$

das Ergebnis von $E$ unter dem Schema $(A_1, \ldots, A_n)$.

*Beispiel 4.12.* Ausgangspunkt ist wiederum die Relation `Autos` aus Beispiel 2.11, sowie das Prädikat $\Theta$ aus Beispiel 4.9. Wir können den Ausdruck

$$\pi_{\texttt{Marke},\texttt{Baujahr}}(\sigma_{\Theta}(\texttt{Autos}))$$

bilden. Er beschreibt dann die Relation, die durch folgende Tabelle gegeben ist:

| **Marke** | **Baujahr** |
|---|---|
| Opel | 2010 |
| VW | 2014 |

Durch

$$\rho_{\texttt{Automarke},\texttt{Jahrgang}}(\pi_{\texttt{Marke},\texttt{Baujahr}}(\sigma_{\Theta}(\texttt{Autos})))$$

werden die Attributnamen dieser Tabelle umbenannt. Wir erhalten damit folgendes Resultat:

| Automarke | Jahrgang |
|---|---|
| Opel | 2010 |
| VW | 2014 |

*Beispiel 4.13.* Mit der Umbenennungsoperation können wir das Problem von mehrfach auftretenden Attributnamen lösen. In Anmerkung 4.4 haben wir gesehen, dass

$$\pi_{\texttt{MatNr}}\big(\pi_{\texttt{Name},\texttt{MatNr},\texttt{MatNr}}(\texttt{Studierende})\big)$$

kein zulässiger relationaler Ausdruck ist. Mit Hilfe der $\rho$-Operation können wir nun eines der `MatNr` Attribute umbennen und so einen zulässigen Ausdruck formulieren durch:

$$\pi_{\texttt{MatNr}}\Big(\rho_{\texttt{Name},\texttt{MatNr},\texttt{MatNr2}}\big(\pi_{\texttt{Name},\texttt{MatNr},\texttt{MatNr}}(\texttt{Studierende})\big)\Big).$$

*Beispiel 4.14.* In Anmerkung 4.7 haben wir gesehen, dass doppelte Attributnamen auch als Resultat eines kartesischen Produktes auftreten können. Im vorherigen Beispiel haben wir gesehen, dass wir mit einer $\rho$-Operation die mehrfach auftretenden Attribute umbennen können. Im Falle des kartesischen Produktes können wir auch zuerst die Attribute umbenennen und anschliessen das Produkt bilden.

Betrachte eine Relation `S` über einem Schema (`A`, `B`). Das kartesische Produkt

$$\rho_{\texttt{L.A},\texttt{L.B}}(\texttt{S}) \times \rho_{\texttt{R.A},\texttt{R.B}}(\texttt{S})$$

ist dann eine Relation über dem Schema

$$(\texttt{L.A},\ \texttt{L.B},\ \texttt{R.A},\ \texttt{R.B}).$$

In diesem Schema sind die Namen der Attribute wieder eindeutig.

Wir werden diesen Ansatz später noch verwenden und führen dafür folgende abkürzende Schreibweise ein. Sei $S$ eine Relation über einem Schema $(A_1, \ldots, A_n)$. Dann schreiben wir

$$\rho_L(\texttt{S})$$

für

$$\rho_{L.A_1,\ldots,L.A_n}(\texttt{S}).$$

### Vereinigung und Mengendifferenz

Die mengentheoretische Vereinigung und die mengentheoretische Differenz von zwei Relationen sind weitere Grundoperationen der relationalen Algebra. Wir gehen aus von den Attributen $A_1, \ldots, A_n$ und dem Relationenschema

$$\mathscr{S} := (A_1, \ldots, A_n),$$

sowie Instanzen $R$ und $S$ von $\mathscr{S}$. Die Vereinigung $(R \cup S)$ und die Mengendifferenz $(R \setminus S)$ von $R$ und $S$ definieren wir dann durch

$$(R \cup S) := \{t \mid t \in R \text{ oder } t \in S\}$$

und

$$(R \setminus S) := \{t \mid t \in R \text{ und } t \notin S\}.$$

Offensichtlich sind $(R \cup S)$ und $(R \setminus S)$ ebenfalls Instanzen des Relationenschemas $\mathscr{S}$. Wir beachten, dass Tupel, die sowohl in $R$ als auch in $S$ vorkommen in $(R \cup S)$ nicht mehrfach gezählt werden.

Wir verlangen hier, dass bei der Bildung von Vereinigung und Mengendifferenz die beteiligten Relationen zum selben Relationenschema gehören. Manchmal wird auch etwas liberaler vorgegangen und nur gefordert, dass die involvierten Relationen dieselbe Stelligkeit und verträgliche Domänen haben.

*Anmerkung 4.15.* In der relationalen Algebra können nur *relative* Komplemente von Relationen $S$ bezüglich von Relationen $R$ eingeführt werden (mit Hilfe der Mengendifferenz $R \setminus S$). Dagegen ist es nicht möglich, das absolute Komplement einer Relation $S$, d. h. die Menge aller Tupel $\{t \mid t \notin S\}$, zu definieren. Im Falle von unendlichen Domänen würden dadurch nämlich Relationen mit unendlich vielen Elementen auftreten.

## 4.3 Weitere Operationen

In den beiden vorangehenden Abschnitten haben wir Ausdrücke der relationalen Algebra eingeführt. Diese sind aufgebaut aus den Basisrelationen

1. Inputrelationen,
2. konstante Relationen

und den Grundoperationen

1. Projektion,
2. kartesisches Produkt,
3. Selektion,
4. Umbenennung,
5. Vereinigung,
6. Differenz.

Die so definierten Ausdrücke reichen aus, um sehr viele Abfragen im relationalen Modell zu formulieren. Es wäre jedoch zu umständlich und nicht besonders bequem, nur damit zu arbeiten. Deshalb führen wir nun weitere Operationen ein, die sich mit Hilfe der Grundoperationen definieren lassen und daher die Ausdrucksstärke nicht erhöhen.

### Durchschnitt

Der Durchschnitt von Relationen kann mit Hilfe der Mengendifferenz definiert werden (siehe Lemma 1.19). Gegeben seien Attribute $A_1, \ldots, A_n$, das Relationenschema

$$\mathscr{S} := (A_1, \ldots, A_n),$$

sowie Instanzen $R$ und $S$ des Schemas $\mathscr{S}$. Der Durchschnitt $(R \cap S)$ von $R$ und $S$ ist dann gegeben durch

$$(R \cap S) := (R \setminus (R \setminus S)).$$

### Natürlicher Verbund (Natural Join)

Um den natürlichen Verbund von Relationen ganz allgemein einzuführen, gehen wir von Attributen $A_1, \ldots, A_m$ und $B_1, \ldots, B_n$, den Relationenschemata

$$\mathscr{S} := (A_1, \ldots, A_m) \text{ und } \mathscr{T} := (B_1, \ldots, B_n)$$

sowie einer Instanz $S$ von $\mathscr{S}$ und einer Instanz $T$ von $\mathscr{T}$ aus. Ausserdem gelte

$$\begin{aligned}
\{A_1, \ldots, A_m\} \cap \{B_1, \ldots, B_n\} &= \{A_{i_1}, \ldots, A_{i_p}\} && \text{mit } i_l < i_h \text{ für } l < h \\
\{A_1, \ldots, A_m\} \setminus \{A_{i_1}, \ldots, A_{i_p}\} &= \{A_{j_1}, \ldots, A_{j_q}\} && \text{mit } j_l < j_h \text{ für } l < h \\
\{B_1, \ldots, B_n\} \setminus \{A_{i_1}, \ldots, A_{i_p}\} &= \{B_{k_1}, \ldots, B_{k_r}\} && \text{mit } k_l < k_h \text{ für } l < h.
\end{aligned}$$

Die Bedingung

$$\text{mit } i_l < i_h \text{ für } l < h$$

sagt, dass in der Sequenz $A_{i_1}, \ldots, A_{i_p}$ ein Attribut $A_{i_l}$ genau dann vor einem Attribut $A_{i_h}$ auftritt, wenn auch in der Sequenz $A_1, \ldots, A_m$ das Attribut $A_{i_l}$ vor dem Attribut $A_{i_h}$ auftritt. Der natürliche Verbund $(S \bowtie T)$ ist nun definiert als

$$\begin{aligned} S \bowtie T \quad := \quad & \rho_{A_{i_1},\ldots,A_{i_p},A_{j_1},\ldots,A_{j_q},B_{k_1},\ldots,B_{k_r}}( \\ & \pi_{L.A_{i_1},\ldots,L.A_{i_p},L.A_{j_1},\ldots,L.A_{j_q},R.B_{k_1},\ldots,R.B_{k_r}}( \\ & \sigma_{L.A_{i_1}=R.A_{i_1}\wedge\cdots\wedge L.A_{i_p}=R.A_{i_p}}(\rho_L(S) \times \rho_R(T)))). \end{aligned} \tag{4.2}$$

Der natürliche Verbund ist also eine zweistellige Operation, die

1. die Attributnamen so umbenennt, dass sie mit $L.$, beziehungsweise mit $R.$, beginnen,
2. ein kartesisches Produkt bildet,
3. eine Selektion durchführt, welche Gleichheit auf den Attributen verlangt, die beiden Schemata gemeinsam sind,
4. mit einer Projektion die Duplikate dieser gemeinsamen Attribute entfernt und die gemeinsamen Attribute an den Anfang stellt,
5. mit einer Umbenennung den Attributen wieder ihre ursprünglichen Namen gibt.

Die Umbenennung im ersten Schritt ist nötig, damit der natürliche Verbund $S \bowtie S$ einer Relation $S$ mit sich selbst definiert ist. Würde man diese ersten Umbenennungen weglassen, so würde das kartesische Produkt $S \times S$ mehrfach auftretende Spaltennamen enthalten. Damit wäre die anschliessende Selektion nicht zulässig, weil die Attributnamen nicht mehr eindeutig eine Spalte bezeichnen würden.

Durch diese Umbenneng haben alle Attribute unterschiedliche Namen. Deshalb können wir für das kartesische Produkt die Vereinfachung der Attributnamen gemäss (4.1) verwenden.

Die Umbenennung im letzten Schritt wäre auch nötig, wenn man die Umbennung im ersten Schritt weglässt. Dann würde nämlich das kartesische Produkt neue Namen für die Attribute einführen.

*Beispiel 4.16.* Wir betrachten eine Relation `Autos` über dem Schema

$$(\underline{\texttt{Marke}}, \texttt{Jahrgang}, \texttt{PersId}).$$

Wir verwenden `Marke` als Primärschlüssel, damit das Beispiel übersichtlich bleibt. Das Attribut `PersId` ist ein Fremdschlüssel auf eine Relation `Personen`, um die Fahrer der Autos anzugeben. `Personen` ist eine Relation über dem Schema

(PersId, Name).

Die Relationen Autos und Personen seien gegeben durch:

**Autos**

| Marke | Jahrgang | PersId |
|---|---|---|
| Opel | 2010 | 1 |
| VW | 1990 | 1 |
| Audi | 2014 | - |
| Skoda | 2014 | 2 |

**Personen**

| PersId | Name |
|---|---|
| 1 | Studer |
| 2 | Meier |

Mit dem natürlichen Verbund Autos ⋈ Personen können wir nun die Verknüpfung zwischen Autos und ihren Fahrern herstellen. Wir erhalten so:

**Autos ⋈ Personen**

| PersId | Marke | Jahrgang | Name |
|---|---|---|---|
| 1 | Opel | 2010 | Studer |
| 1 | VW | 1990 | Studer |
| 2 | Skoda | 2014 | Meier |

Bei der Verwendung eines natürlichen Verbundes ist es wichtig zu beachten, dass der Join über die gemeinsamen Attribute ausgeführt wird. Im obigen Beispiel ist es also wichtig, dass beide Tabellen das gemeinsame Attribut PersId besitzen. Es ist jedoch irrelevant, dass es einen Fremdschlüssel von Autos zu Personen gibt.

Wichtig ist auch, dass der Join über *alle* gemeinsamen Attribute berechnet wird, auch wenn diese nur zufällig gleich heissen und eine unterschiedliche Bedeutung haben. So kann es passieren, dass der natürliche Verbund nicht das gewünschte Resultat liefert. Dieser Fall wird im folgenden Beispiel illustriert.

*Beispiel 4.17.* Wir betrachen wieder die Relation Autos aus Beispiel 4.16. Die Relation Personen ergänzen wir nun durch ein neues Attribut Jahrgang wie folgt:

**Personen**

| PersId | Name | Jahrgang |
|---|---|---|
| 1 | Studer | 1990 |
| 2 | Meier | 1994 |

Das Attribut `Jahrgang` in der Tabelle `Autos` bezeichnet den Jahrgang des Autos, das Attribut `Jahrgang` in der Tabelle `Personen` meint den Jahrgang der Person. Für den natürlich Verbund erhalten wir nun aber folgende Tabelle:

**Autos ⋈ Personen**

| Jahrgang | PersId | Marke | Name |
|---|---|---|---|
| 1990 | 1 | VW | Studer |

Der Grund ist natürlich, dass sowohl `Jahrgang` als auch `PersId` als Join-Attribute verwendet werden. Damit müssen Tupel aus `Autos` und `Personen` in *beiden* Attributen übereinstimmen, damit ihre Kombination in das Resultat aufgenommen wird.

Auch im Beispiel 3.3 zur Hochschul-Datenbank darf man keinen natürlichen Verbund über `Vorlesungen` und `Übungen` verwenden, um die Vorlesungen mit den dazugehörigen Übungen zu erhalten. Es würden nämlich nur diejenigen gefunden, bei denen die Vorlesung und die Übungen im selben Hörsaal stattfinden.

*Anmerkung 4.18.* Diese Eigenschaft des natürlichen Verbundes ist etwas tückisch. Nehmen wir an, für eine Datenbank sei eine Abfrage mit Hilfe eines natürlichen Verbundes implementiert worden. Diese, auf der ursprünglichen Datenbank korrekte Implementierung der Abfrage, kann später falsche Antworten liefern, falls neue Attribute zu den Relationen des Joins hinzugefügt werden.

*Anmerkung 4.19.* Sei $S$ eine Relation über dem Schema $(A_1, \ldots, A_m)$ und $T$ eine Relation über dem Schema $(B_1, \ldots, B_n)$. Weiter nehmen wir an, dass die beiden Schemata keine gemeinsamen Attribute besitzen, d. h.

$$\{A_1, \ldots, A_m\} \cap \{B_1, \ldots, B_n\} = \emptyset.$$

In der Definition des natürlichen Verbundes (4.2) ist dann das Selektionsprädikat eine Konjunktion über der leeren Menge. Wie üblich verwenden wir

$$\bigwedge \emptyset = \texttt{true}.$$

Damit erhalten wir

$$S \bowtie T \quad = \quad S \times T.$$

Das heisst, falls die Schemata von $S$ und $T$ keine gemeinsamen Attribute besitzen, so ist der natürliche Verbund von $S$ und $T$ gleich dem kartesischen Produkt von $S$ und $T$. Dabei verwenden wir die Vereinfachung der Attributnamen gemäss (4.1).

### $\Theta$-Verbund ($\Theta$-Join)

Im natürlichen Verbund ist das Kriterium für den Join implizit vorgegeben: gleichnamige Attribute müssen den gleichen Wert haben. Im $\Theta$-Verbund kann nun ein beliebiges Join-Kriterium $\Theta$ verwendet werden. In diesem Kriterium kann nicht nur auf Gleichheit von Argumenten getestet werden, sondern es sind beliebige Prädikate (im Sinne der Selektionsoperation) möglich.

Der $\Theta$-Verbund $R \bowtie_\Theta S$ von zwei Relationen $R$ und $S$ ist definiert durch

$$R \bowtie_\Theta S := \sigma_\Theta(R \times S).$$

*Beispiel 4.20.* Mit den Daten aus Beispiel 4.17 erhalten wir für

$$\texttt{Autos} \bowtie_{\texttt{Autos.PersId=Personen.PersId}} \texttt{Personen}$$

folgende Tabelle, wobei `A` für `Autos` und `P` für `Personen` steht:

| A.Marke | A.Jahrgang | A.PersId | P.PersId | P.Name | P.Jahrgang |
|---|---|---|---|---|---|
| Opel | 2010 | 1 | 1 | Studer | 1990 |
| VW | 1990 | 1 | 1 | Studer | 1990 |
| Skoda | 2014 | 2 | 2 | Meier | 1994 |

Ein $\Theta$-Join bei dem im Prädikat $\Theta$ nur auf Gleichheit getestet wird (so wie im Beispiel 4.20) wird oft *Equi-Join* genannt. In einem allgemeinen $\Theta$-Join können aber beliebige Prädikate verwendet werden. Beispielsweise liefert der $\Theta$-Join

$$\texttt{Autos} \bowtie_{\texttt{Autos.PersId=Personen.PersId} \wedge \texttt{Personen.Jahrgang} \leq 1990} \texttt{Personen}$$

die Fahrer, welche 1990 oder früher geboren wurden, zusammen mit ihren Autos. Das heisst, wir erhalten folgende Tabelle für die Relationen `Autos` und `Personen` aus Beispiel 4.20:

| A.Marke | A.Jahrgang | A.PersId | P.PersId | P.Name | P.Jahrgang |
|---|---|---|---|---|---|
| Opel | 2010 | 1 | 1 | Studer | 1990 |
| VW | 1990 | 1 | 1 | Studer | 1990 |

### Linker äusserer Verbund

Betrachten wir noch einmal die Relationen `Autos` und `Personen` aus Beispiel 4.16:

**Autos**

| Marke | Jahrgang | PersId |
|---|---|---|
| Opel | 2010 | 1 |
| VW | 1990 | 1 |
| Audi | 2014 | - |
| Skoda | 2014 | 2 |

**Personen**

| PersId | Name |
|---|---|
| 1 | Studer |
| 2 | Meier |

Im natürlichen Verbund `Autos` $\bowtie$ `Personen` erscheint der `Audi` nicht, weil sein `PersId` Attribut den Wert `Null` hat. Wir können aber auch verlangen, dass *alle* Autos in der Resultatrelation des Verbundes vorkommen müssen. Dazu verwenden wir den linken äusseren Verbund `Autos` $\overset{\leftarrow}{\bowtie}$ `Personen`, welcher wie folgt definiert ist.

Gegeben seien eine Relation $R$ über einem Schema $\mathscr{R}$ und eine Relation $S$ über einem Schema $\mathscr{S}$. Weiter sei $\mathscr{T}$ das Schema des natürlichen Verbundes $R \bowtie S$.

Mit Anmerkung 4.6 können wir annehmen, dass

$$\{(\texttt{Null}, \ldots, \texttt{Null})\}$$

eine konstante Relation über dem Schema $(B_{k_1}, \ldots, B_{k_r})$ ist, wobei $B_{k_1}, \ldots, B_{k_r}$ diejenigen Attribute aus $\mathscr{S}$ sind, die nicht in $\mathscr{R}$ vorkommen.

Wir definieren nun

$$R \overset{\leftarrow}{\bowtie} S := (R \bowtie S) \cup \pi_{\mathscr{T}}\Big(\big(R \setminus \pi_{\mathscr{R}}(R \bowtie S)\big) \times \{(\texttt{Null}, \ldots, \texttt{Null})\}\Big)$$

Wir erhalten so:

**Autos** $\overset{\leftarrow}{\bowtie}$ **Personen**

| PersId | Marke | Jahrgang | Name |
|---|---|---|---|
| 1 | Opel | 2010 | Studer |
| 1 | VW | 1990 | Studer |
| 2 | Skoda | 2014 | Meier |
| - | Audi | 2014 | - |

Im Abschn. 5.4 werden wir noch weitere Varianten der Join-Operation betrachten, welche für Anwendungen wichtig sind und von der Datenbanksprache SQL unterstützt werden.

## Division

Die Division wird häufig zur Formulierung von allquantifizierten Abfragen verwendet. Um die etwas umfängliche Definition besser verstehen zu können, beginnen wir mit einem Beispiel.

*Beispiel 4.21.* In diesem Beispiel geht es um Mechaniker, welche bestimmte Automarken reparieren können. Ausserdem betrachten wir eine Garage in der Autos von verschiedenen Marken abgestellt sind. Wir wählen die Attribute `Name` (des Mechanikers) sowie `Marke` (des Autos). Die Relation `Mechaniker` speichert, welcher Mechaniker welche Automarken reparieren kann:

**Mechaniker**

| **Name** | **Marke** |
|---|---|
| Studer | Opel |
| Meier | Opel |
| Meier | VW |
| Meier | Audi |

Die Relation `Garage` speichert, welche Automarken in der Garage abgestellt sind:

**Garage**

| **Marke** |
|---|
| Opel |
| Audi |

Nun möchten wir die Namen derjenigen Mechaniker, welche *alle* Automarken die in der Garage vorkommen, reparieren können. Dies wird durch die (noch zu definierende) Division $\texttt{Mechaniker} \div \texttt{Garage}$ erreicht. Wir erhalten nämlich:

**Mechaniker ÷ Garage**

| **Name** |
|---|
| Meier |

Bevor wir die Division formal einführen können, benötigen wir noch eine abkürzende Schreibweise, die auch später nützlich sein wird.

**Definition 4.22.** Gegeben seien paarweise verschiedene Attribute

$$A_1, \ldots, A_m, A_{m+1}, \ldots, A_{m+n}$$

sowie die Schemata

$$\mathscr{S} = (A_1, \ldots, A_m) \qquad \text{und} \qquad \mathscr{T} = (A_{m+1}, \ldots, A_{m+n}).$$

Weiter sei

$$\mathscr{R} = (A_{i_1}, \ldots, A_{i_{m+n}}) \qquad \text{mit } i_j \text{ und } i_k \text{ paarweise verschieden}$$

ein Schema mit den Attributen $A_1, \ldots, A_{m+n}$.

Ist $S$ eine Instanz von $\mathscr{S}$ und $T$ eine Instanz von $\mathscr{T}$, ist $s$ ein $m$-Tupel aus $S$ und ist $t$ ein $n$-Tupel aus $T$, so schreiben wir

$$(s * t)$$

für das $(m+n)$-Tupel, das wir durch geeignete Konkatenation von $s$ und $t$ erhalten, so dass gilt:

$$\pi_j(s * t) \simeq s[A_{i_j}] \quad \text{falls } i_j \leq m$$
$$\pi_j(s * t) \simeq t[A_{i_j}] \quad \text{falls } i_j > m$$

Damit kann $(s * t)$ in eine Instanz des Schemas $\mathscr{R}$ eingefügt werden. Die Bedeutung der $*$-Operation hängt also wesentlich von den beteiligten Schemata ab. Wir geben diese jedoch nicht explizit an, da sie immer aus dem Kontext klar sein werden.

*Beispiel 4.23.* Gegeben seien Attribute `A`, `B`, `C` und `D`, die Schemata

$$\mathscr{S} = (A, B) \qquad \mathscr{T} = (C, D) \qquad \mathscr{R} = (C, B, D, A)$$

sowie eine Instanz $S$ von $\mathscr{S}$ und eine Instanz $T$ von $\mathscr{T}$. Es seien nun

$$s = (1, 2) \in S \qquad \text{und} \qquad t = (3, 4) \in T.$$

Damit gilt

$$s * t = (3, 2, 4, 1).$$

Zur allgemeinen Einführung der Division von Relationen gehen wir aus von Attributen $A_1, \ldots, A_m$ und einer echten Teilmenge $\{B_1, \ldots, B_n\}$ der Attributmenge $\{A_1, \ldots, A_m\}$. Ausserdem sei

$$\{A_{i_1}, \ldots, A_{i_{m-n}}\} := \{A_1, \ldots, A_m\} \setminus \{B_1, \ldots, B_n\} \text{ mit } i_l < i_k \text{ für } l < k.$$

Schliesslich betrachten wir noch die Schemata

$$\mathscr{R} := (A_1, \ldots, A_m),$$
$$\mathscr{S} := (B_1, \ldots, B_n),$$
$$\mathscr{T} := (A_{i_1}, \ldots, A_{i_{m-n}}).$$

Ist $R$ eine Instanz von $\mathscr{R}$ und $S$ eine Instanz von $\mathscr{S}$, so ist die Division $(R \div S)$ von $R$ durch $S$ eine Relation die zum Schema $\mathscr{T}$ gehört. Ein $(m-n)$-Tupel $t$ aus $\pi_{A_{i_1},\ldots,A_{i_{m-n}}}(R)$ soll genau dann Element der Relation $(R \div S)$ sein, falls für alle Tupel $s$ aus $S$ das Tupel $(t * s)$ zu $R$ gehört, d. h.

$$(R \div S) := \{t \in \pi_{A_{i_1},\ldots,A_{i_{m-n}}}(R) \mid (\forall s \in S)((t * s) \in R)\}.$$

Man kann leicht zeigen, dass sich $(R \div S)$ aus unseren Grundoperationen gewinnen lässt. Dazu muss man nur nachrechnen, dass

$$(R \div S) = \pi_{A_{i_1},\ldots,A_{i_{m-n}}}(R) \setminus \pi_{A_{i_1},\ldots,A_{i_{m-n}}}(\pi_{A_1,\ldots,A_m}(\pi_{A_{i_1},\ldots,A_{i_{m-n}}}(R) \times S) \setminus R).$$

Für das Beispiel 4.21 bedeutet dieser relationale Ausdruck etwa Folgendes:

Finde alle Mechaniker,
für die es keine Automarke in der Garage gibt,
die sie nicht reparieren können. (4.3)

Diese doppelte Negation ist zwar schlechtes Deutsch aber gute Logik!

Der Divisor (die Relation durch die geteilt wird) kann natürlich auch mehrere Attribute haben. Folgendes Beispiel illustriert diesen Fall.

*Beispiel 4.24.* Wir wählen Attribute `A1`, `A2` und `B1`, `B2` sowie die Relationen `R`, `S`, die durch folgende Tabellen gegeben sind:

**R**

| A1 | A2 | B1 | B2 |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 1 | 2 | 5 | 6 |
| 2 | 3 | 5 | 6 |
| 5 | 4 | 3 | 4 |
| 5 | 4 | 5 | 6 |
| 1 | 2 | 4 | 5 |
| 1 | 2 | 7 | 8 |

**S**

| B1 | B2 |
|---|---|
| 3 | 4 |
| 5 | 6 |
| 7 | 8 |

Dann ist die Division `R` ÷ `S` von `R` durch `S` die Relation, die der folgenden Tabelle entspricht:

| R ÷ S | |
|---|---|
| **A1** | **A2** |
| 1 | 2 |

Das folgende Lemma zeigt, dass die Division von Relationen in einem gewissen Sinne die Umkehroperation zum kartesischen Produkt ist. Der Beweis dieser Behauptung wird durch einfaches Nachrechnen geführt.

**Lemma 4.25.** *Seien $\mathscr{R} = (A_1, \ldots, A_m)$ und $\mathscr{S} = (B_1, \ldots, B_n)$ zwei Relationenschemata, $R$ eine Instanz von $\mathscr{R}$ und $S$ eine Instanz von $\mathscr{S}$. Wir nehmen an, dass $\mathscr{R}$ und $\mathscr{S}$ keine gemeinsamen Attributnamen enthalten. Somit können wir das kartesische Produkt $R \times S$ als Relation über dem Schema $(A_1, \ldots, A_m, B_1 \ldots, B_n)$ auffassen.*

*Dann gilt, falls $S$ nicht-leer ist,*

$$(R \times S) \div S = R.$$

Die Einschränkung auf nicht-leere Relationen $S$ ist keine Überraschung. Über den rationalen Zahlen gilt $(a \cdot b)/b = a$ auch nur für $b \neq 0$.

*Anmerkung 4.26.* Die Division der relationalen Algebra verhält sich wie eine Division mit Rest (d. h. wie eine Ganzzahl-Division). Damit meinen wir, dass zwar

$$(R \div S) \times S \subseteq R$$

gilt, aber $(R \div S) \times S = R$ nicht gelten muss (siehe Beispiele 4.21 und 4.24). Genau so gilt für die Ganzzahl-Division nur $(7/3) \cdot 3 \leq 7$ aber nicht $(7/3) \cdot 3 = 7$.

## 4.4 Relationale Algebra als Query Sprache

Anhand der vorhergehenden Beispiele haben wir bereits gesehen, dass mithilfe der Operationen der relationalen Algebra Datenbankabfragen (Queries) formuliert werden können. Dieser Aspekt soll in diesem Abschnitt noch weiter diskutiert werden.

Als Beispiel betrachten wir hier eine Film-Datenbank. Dazu verwenden wir die Attribute `PId` (Personen-Id), `Fn` (Familienname), `Vn` (Vorname), `FId` (Film-Id), `Dt` (Datum), `Reg` (Regisseur), `Titel` und `Jahr` und betrachten die Relationenschemata

$$\mathscr{P} = (\underline{\texttt{PId}}, \texttt{Fn}, \texttt{Vn}),$$
$$\mathscr{PF} = (\underline{\texttt{PId}}, \underline{\texttt{FId}}, \underline{\texttt{Dt}}),$$
$$\mathscr{F} = (\underline{\texttt{FId}}, \texttt{Reg}, \texttt{Titel}, \texttt{Jahr}).$$

Ferner sei die Relation `Personen` eine Instanz von $\mathscr{P}$, die Relation `Schaut` eine Instanz von $\mathscr{PF}$ und `Filme` eine Instanz von $\mathscr{F}$.

Wir betrachten nun einige Datenbankabfragen und diejenigen Ausdrücke, mit denen diese Abfragen beantwortet werden können.

### Query 1

Wie lauten die Titel der Filme aus dem Jahr 2002 bei denen `Spielberg` Regie geführt hat?

$$\pi_{\texttt{Titel}}(\sigma_{\texttt{Jahr=2002} \wedge \texttt{Reg='Spielberg'}}(\texttt{Filme}))$$

### Query 2

Ermittle Familien- und Vornamen derjenigen Personen, die Filme geschaut haben mit `Spielberg` oder `Coppola` als Regisseur.

$$\pi_{\texttt{Fn,Vn}}(\texttt{Personen} \bowtie (\texttt{Schaut} \bowtie (\sigma_{\texttt{Reg='Spielberg'} \vee \texttt{Reg='Coppola'}}(\texttt{Filme}))))$$

### Query 3

Nenne `Titel` und `Jahr` der Filme, welche `Eva Meier` vor dem 30. November 2009 geschaut hat.

$$\pi_{\texttt{Titel,Jahr}}(\sigma_{\texttt{Fn='Meier'} \wedge \texttt{Vn='Eva'}}(\texttt{Personen}) \bowtie (\sigma_{\texttt{Dt} < 20091130}(\texttt{Schaut}) \bowtie \texttt{Filme}))$$

### Query 4

Wie lauten die Personennummern der Personen, die alle Filme von `Spielberg` geschaut haben?

$$\pi_{\texttt{PId,FId}}(\texttt{Schaut}) \div \pi_{\texttt{FId}}(\sigma_{\texttt{Reg='Spielberg'}}(\texttt{Filme}))$$

In diesem Ausdruck ist es wichtig, dass in der Division die Projektion

$$\pi_{\texttt{PId,FId}}(\texttt{Schaut})$$

verwendet wird und nicht die ursprüngliche Relation `Schaut`. Die Abfrage

$$\pi_{\texttt{PId}}(\texttt{Schaut} \div \pi_{\texttt{FId}}(\sigma_{\texttt{Reg='Spielberg'}}(\texttt{Filme})))$$

liefert nämlich die Personennummer der Personen, die *an einem einzigen Tag* alle Filme von `Spielberg` geschaut haben.

## 4.5 Gruppierung und Aggregation

In diesem Abschnitt studieren wir Erweiterungen, die im strengen Sinne nicht zur relationalen Algebra gehören, aber für praktische Anwendungen sehr wichtig sind. Vor allem die sogenannten *Aggregatsfunktionen* spielen dabei eine grosse Rolle. Dazu gehören unter anderem folgende Funktionen:

- **count** (Zählen),
- **sum** (Summieren),
- **min**, **max** (Minimum bzw. Maximum),
- **avg** (Durchschnitt).

Häufig werden Aggregatsfunktionen nicht über eine ganze Relation berechnet, sondern die Relation wird in Gruppen zerlegt und die Aggregatsfunktion wird über jede Gruppe einzeln ausgewertet. Dies wird wie folgt definiert.

Gegeben seien Attribute $A_1, \ldots, A_m$, eine Teilmenge $\{B_1, \ldots, B_n\}$ der Attributmenge $\{A_1, \ldots, A_m\}$, ein Element $C$ von $\{A_1, \ldots, A_m\} \setminus \{B_1, \ldots, B_n\}$ und das Relationenschema

$$\mathscr{S} = (A_1, \ldots, A_m).$$

Ist **agg** eine unserer Aggregatsfunktionen und $R$ eine Instanz von $\mathscr{S}$, so definieren wir die *GroupBy* Operation $\gamma$ durch:

$$\begin{aligned}\gamma(R, (B_1, \ldots, B_n), \mathbf{agg}, C) := \\ \{(b_1, \ldots, b_n, a) \mid (b_1, \ldots, b_n) \in \pi_{B_1,\ldots,B_n}(R) \text{ und} \\ a = \mathbf{agg}(\{x \mid (b_1, \ldots, b_n, x) \in \pi_{B_1,\ldots,B_n,C}(R)\})\}.\end{aligned}$$

Als Schema dieser Relation verwenden wir:

$$(B_1, \ldots, B_n, \mathbf{agg}(C)).$$

Wir betrachten nochmals unsere Beispiel-Filmdatenbank aus Abschn. 4.4. Mit Hilfe der Aggregatsfunktionen und Gruppierung können wir nun folgende Fragen beantworten.

### Query 5

Welcher Regisseur hat in welchem Jahr wie viele Filme gedreht?

$$\gamma(\texttt{Filme}, (\texttt{Reg}, \texttt{Jahr}), \mathbf{count}, \texttt{FId})$$

### Query 6

In welchem Jahr hat `Spielberg` das letzte Mal Regie geführt?

$$\sigma_{\texttt{Reg='Spielberg'}}(\gamma(\texttt{Filme}, (\texttt{Reg}), \mathbf{max}, \texttt{Jahr}))$$

*Beispiel 4.27.* Wir betrachten die folgende Tabelle um die einzelnen Berechnungsschritte der Query 5

$$\gamma(\texttt{Filme}, (\texttt{Reg}, \texttt{Jahr}), \mathbf{count}, \texttt{FId})$$

zu illustrieren:

**Filme**

| FId | Reg | Titel | Jahr |
|---|---|---|---|
| 1 | Lynch | Absurda | 2007 |
| 2 | Lynch | Mulholland Drive | 2001 |
| 3 | Spielberg | Jurassic Parc | 1993 |
| 4 | Lynch | Boat | 2007 |
| 5 | Spielberg | Schindler's List | 1993 |
| 6 | Lynch | Bug Crawls | 2007 |

Wir berechnen zuerst die verschiedenen Gruppen:

$\pi_{\texttt{Reg,Jahr}}(\texttt{Filme})$

| **Reg** | **Jahr** |
|---|---|
| Lynch | 2007 |
| Lynch | 2001 |
| Spielberg | 1993 |

Für jede Gruppe berechnen wir nun die Aggregatsfunktion. Das heisst, für jedes Tupel $(b_1, b_2)$ aus der Relation $\pi_{\texttt{Reg,Jahr}}(\texttt{Filme})$ bestimmen wir

$$\textbf{count}(\{x \mid (b_1, b_2, x) \in \pi_{\texttt{Reg,Jahr,FId}}(\texttt{Filme})\}.$$

Für das Tupel $(\texttt{Lynch}, 2007)$ müssen wir also $\textbf{count}(\{1, 4, 6\})$ ausrechnen, was das Resultat 3 liefert. Insgesamt liefert die Query

$$\gamma(\texttt{Filme}, (\texttt{Reg}, \texttt{Jahr}), \textbf{count}, \texttt{FId})$$

somit das Resultat:

| **Reg** | **Jahr** | **count**(**FId**) |
|---|---|---|
| Lynch | 2007 | 3 |
| Lynch | 2001 | 1 |
| Spielberg | 1993 | 2 |

Bisher haben wir noch nicht spezifiziert, wie sich die Aggregatsfunktionen bei `Null` Werten und bei der leeren Menge verhalten sollen.

**Definition 4.28.** Sei $f$ eine Aggregatsfunktion und $X$ eine Menge von Werten, welche möglicherweise `Null` enthält. Wir setzen

$$f(X) := f(X \setminus \{\texttt{Null}\}).$$

Das heisst, dass die Aggregatsfunktionen `Null` Werte ignorieren.

Weiter definieren wir

$$\textbf{count}(\emptyset) := 0$$

und

$$\mathbf{sum}(\emptyset) := \mathbf{min}(\emptyset) := \mathbf{max}(\emptyset) := \mathbf{avg}(\emptyset) := \texttt{Null}.$$

*Beispiel 4.29.* Mit dieser Definition erhalten wir für $X = \{\texttt{Null}, 1, 2, 3\}$ die folgenden Resultate:

$$\mathbf{sum}(X) = 6, \qquad \mathbf{count}(X) = 3 \qquad \text{und} \qquad \mathbf{avg}(X) = 2.$$

Ausserdem finden wir $\mathbf{count}(\{\texttt{Null}\}) = 0$ und $\mathbf{avg}(\{\texttt{Null}\}) = \texttt{Null}$.

Nach unserer Definition von $\gamma$ wird die Aggregatsfunktion auf eine *Menge* angewendet, d. h. Duplikate von Elementen spielen keine Rolle (beziehungsweise werden entfernt). Dies kann zu unerwünschten Effekten führen, wie das folgende Beispiel zeigt.

*Beispiel 4.30.* Wir betrachten wieder die Filmdatenbank aus Abschn. 4.4. Jedoch erweitern wir das Schema $\mathscr{F}$ durch ein neues Attribut `Dauer`, um die Länge eines Films abzuspeichern. Wir haben also

$$\mathscr{F} = (\underline{\texttt{FId}}, \texttt{Reg}, \texttt{Titel}, \texttt{Jahr}, \texttt{Dauer}).$$

Die Instanz `Filme` des Schemas $\mathscr{F}$ habe folgende Einträge, wobei wir nur die Attribute `FId`, `Jahr`, `Dauer` betrachten.

**Filme**

| FId | Jahr | Dauer |
|---|---|---|
| 1 | 2010 | 120 |
| 2 | 2012 | 90 |
| 3 | 2012 | 120 |
| 4 | 2010 | 100 |
| 5 | 2012 | 120 |

Wir möchten nun mit Hilfe der Aggregationsfunktion **avg** folgende Frage beantworten.

### Query 7

Welches ist die durchschnittliche Dauer der Filme pro Jahr?

Analog zu den obigen Beispielen wollen wir dazu den Ausdruck

$$\gamma(\texttt{Filme}, (\texttt{Jahr}), \mathbf{avg}, \texttt{Dauer})$$

verwenden. Dieser liefert als Resultat:

| Jahr | avg(Dauer) |
|---|---|
| 2010 | 110 |
| 2012 | 105 |

Für das Jahr 2010 wird die durchschnittliche Dauer berechnet durch

$$\mathbf{avg}(\{100, 120\}),$$

was das korrekte Resultat liefert. Für das Jahr 2012 wird die durchschnittliche Dauer jedoch berechnet durch

$$\mathbf{avg}(\{90, 120, 120\}),$$

was das (unerwünschte) Resultat 105 liefert. Da der Durchschnitt einer *Menge* genommen wird, werden Duplikate nicht korrekt behandelt. Die Dauer 120 wird nur *einmal* (statt zweimal) berücksichtigt.

Um derartige Probleme zu umgehen, arbeiten wir in Fällen, bei denen Duplikate eine Rolle spielen, nicht mit Mengen, sondern mit *Multimengen*. Dabei handelt es sich um Kollektionen von Objekten, bei denen mehrfache Vorkommnisse von Elementen registriert werden, aber ihre Reihenfolge unwichtig ist. Wir können Multimengen also als Mengen auffassen, bei denen Elemente mehrfach vorkommen können, oder als Listen, bei denen die Reihenfolge nicht beachtet wird.

Für Attribute $A_1, \ldots, A_m$, eine Teilmenge $\{B_1, \ldots, B_n\}$ von $\{A_1, \ldots, A_m\}$, ein Element $C$ von $\{A_1, \ldots, A_m\} \setminus \{B_1, \ldots, B_n\}$ und eine Relation $R$ über dem Schema $(A_1, \ldots, A_m)$ definieren wir die *Multimengen-GroupBy* Operation

$$\Gamma(R, (B_1, \ldots, B_n), \mathbf{agg}, C)$$

analog zu $\gamma(R, \{B_1, \ldots, B_n\}, \mathbf{agg}, C)$ mit dem Unterschied, dass wir den Input der Aggregatsfunktion **agg** als Multimenge behandeln.

Query 7 können wir also korrekt beantworten mit

$$\Gamma(\texttt{Filme}, (\texttt{Jahr}), \mathbf{avg}, \texttt{Dauer}).$$

Wir erhalten so das richtige Resultat:

| Jahr | avg(Dauer) |
|---|---|
| 2010 | 110 |
| 2012 | 110 |

## Weiterführende Literatur[1]

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases: The Logical Level. Addison-Wesley (1995)
2. Codd, E.F.: A relational model of data for large shared data banks. Commun. ACM **13**(6), 377–387 (1970). https://doi.org/10.1145/362384.362685
3. Codd, E.F.: Extending the database relational model to capture more meaning. ACM Trans. Database Syst. **4**(4), 397–434 (1979). https://doi.org/10.1145/320107.320109
4. Grant, J.: Null values in SQL. SIGMOD Rec. **37**(3), 23–25 (2008). https://doi.org/10.1145/1462571.1462575

[1] Zusammen mit dem Relationenmodell führt Codd [2] auch die relationale Algebra ein. Diese erste Version enthält jedoch noch keine `Null` Werte. Verschiedene Interpretationen von `Null` Werten sowie äussere Verbund-Operationen werden später in [3] beschrieben. Grant [4] untersucht ebenfalls die unterschiedlichen Bedeutungen von `Null` und illustriert diese mit verschiedenen Beispielen (siehe auch unser Beispiel 4.11). Das Buch von Abiteboul et al. [1] bietet eine knappe und präzise Darstellung der relationalen Algebra und zeigt ihre Beziehung zu verschiedenen anderen formalen Query-Sprachen.