

# Datenbanksysteme

## Relationale Algebra

Burkhardt Renz

Fachbereich MNI  
Technische Hochschule Mittelhessen

Sommersemester 2018

# Inhalt

- Relationen und relationale Algebra
  - Motivation: Relationen in der Mathematik
  - Relationen
  - Relationen und Relationsvariablen
- Operatoren der relationalen Algebra
  - Übersicht
  - Operatoren im Detail
  - Gesetze der relationalen Algebra
- Datenbanken und Integritätsbedingungen
  - Integritätsbedingungen für eine Relationsvariable
  - Integritätsbedingungen für mehrere Relationsvariablen
  - Definition Datenbank

# Relationen in der Mathematik

Seien  $M_1, M_2, \dots, M_n$  Mengen.

Eine  **$n$ -äre Relation**  $R$  ist eine Teilmenge des kartesischen Produkts  $M_1 \times M_2 \times \dots \times M_n$  dieser Mengen:

$$R = \{(m_1, m_2, \dots, m_n) / m_1 \in M_1, m_2 \in M_2, \dots, m_n \in M_n\}$$

Der **Grad** der Relation ist  $n$ .

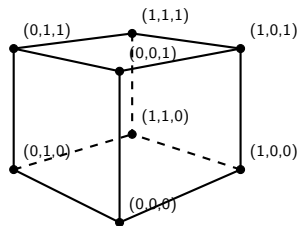
Ist  $n = 2$  sagt man **binäre** Relation, ist  $n = 3$  **ternäre** Relation.

# Beispiel 1

Wir betrachten die Teilmenge  $E \subset \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$  mit

$$E = \{(0,0,0), (0,0,1), (0,1,0), (1,0,0), (1,1,0), (0,1,1), (1,0,1), (1,1,1)\}$$

Man kann  $E$  als die Menge der Ecken des Einheitswürfels in  $\mathbb{Z}^3$  sehen:



## Beispiel 2

Gegeben seien:

$A$  eine Menge von Fluggesellschaften,

$N$  eine Menge von Flugnummern,

$F$  eine Menge von Flughäfen und

$T$  eine Menge von Abflugzeiten.

Wir betrachten die 5-stellige Relation

$$\textit{Flugplan} \subset A \times N \times F \times F \times T$$

Elemente dieser Relation könnten sein:

('Lufthansa', 'LH314', 'FRA', 'PDX', '11:30')

('Delta', 'DX423', 'SFI', 'LAX', '12:00')

('Singapore Airlines', 'SG297', 'SIN', 'CHC', '20:50')

Die  $\langle \text{Fluggesellschaft} \rangle$  fliegt unter  $\langle \text{Flugnummer} \rangle$  von  
 $\langle \text{Flughafen} \rangle$  zu  $\langle \text{Flughafen} \rangle$  um  $\langle \text{Abflugzeit} \rangle$ .

# Konzept einer Algebra/ algebraischen Struktur

- Menge von Elementen
- Operationen, d.h. Verknüpfungen der Elemente
- Axiome und Gesetze, d.h. Gesetze der Operationen

Operatoren, die auf ein einzelnes Element der Menge angewendet werden, heißen **unäre** Operatoren.

Operatoren, die auf Paare von Elemente der Menge angewendet werden, heißen **binäre** Operatoren.

Eine Algebra ist **abgeschlossen** unter einer Operation, wenn das Ergebnis der Operation wieder ein Element der Algebra ist.

# Beispiel 1

$(\mathbb{Z}, -, +)$  die Menge der ganzen Zahlen mit

- Negation, unäre Operation:  $a \in \mathbb{Z} \mapsto -a \in \mathbb{Z}$
- Addition, binäre Operation:  $(a, b) \in \mathbb{Z}^2 \mapsto a + b \in \mathbb{Z}$

Axiom (im Beispiel die Kommutativität):

$$\forall a, b \in \mathbb{Z} : a + b = b + a$$

## Beispiel 2

$(\mathbb{B}, \wedge, \vee, \neg)$  Boole'sche Algebra mit

Menge  $\mathbb{B} = \{true, false\}$

- Negation, unärer Operator:  $\neg$
- Konjunktion (and), binärer Operator:  $\wedge$
- Disjunktion (or), binärer Operator:  $\vee$

Beispiel für Gesetze in dieser Algebra:

$$\neg(p \wedge q) \equiv (\neg p) \vee (\neg q)$$

$$\neg(p \vee q) \equiv (\neg p) \wedge (\neg q)$$



# Beispiel 3

$(\mathbb{R}, \sigma, \pi, \times, \bowtie, \cap, \cup, \dots)$

Menge  $\mathbb{R}$  der Relationen

- Restriktion/Selektion  $\sigma$
- Projektion  $\pi$
- ...

In der relationalen Algebra gelten eine Vielzahl von Gesetzen, von denen wir später einige betrachten werden.

⇒ Relationale Algebra

# Datentyp

Ein **Datentyp** ist eine Menge von Werten. (In der Literatur wird auch der Begriff *Domäne* für Datentyp verwendet.)

## Beispiel

- Integers  $\{\dots, -2, -1, 0, 1, 2, 3, \dots\}$
- Wahrheitswerte  $\{\text{true}, \text{false}\}$
- Menge deutscher KFZ-Kennzeichen
- ...

Ein Datentyp hat üblicherweise selbst Operatoren

## Beispiel

- $+$ ,  $-$ ,  $\times$ ,  $/$  arithmetische Operatoren für Integer
- $<$ ,  $\leq$ ,  $=$ ,  $\geq$ ,  $>$  Vergleichsoperatoren
- $\wedge$ ,  $\vee$ ,  $\neg$  Operatoren für Wahrheitswerte

# Attribute

Ein **Attribut** ist ein **benannter Datentyp**, d.h.

Ein Attribut besteht aus

- einem Namen und
- einem zugeordneten Datentyp (einer Domäne)

## Beispiel

- ArtNr: numeric(6)
- Bez: varchar(40)
- ...

Wir gehen in dem Abschnitt über die relationale Algebra immer davon aus, dass Name und Datentyp zusammengehören, d.h. es kann nicht sein, dass zwei Attribute denselben Namen, aber verschiedenen Datentyp haben.

# Tupel

Ein **Tupel** ist eine Zuordnung von Werten zu einer Menge von Attributen.

Dabei muss der zugeordnete Wert stets vom Datentyp des jeweiligen Attributs sein.

## Beispiel

[ArtNr = 100101, Bez = 'Les Châteaux', Weingut = 'Louis Max']

In der Mathematik sind die Tupel von Relationen in der Regel *geordnete* Tupel, d.h. die Zugehörigkeit zu einer Wertemenge ergibt sich aus der Position im Tupel.

In der relationalen Algebra sind die Tupel Zuordnungen von Werten zu *benannten* Attributen.

# Relationsschema

Ein **Relationsschema** ist eine Menge von Attributen.

Das Relationsschema definiert die Struktur der Tupel in einer Relation.

## Beispiel (Relationsschema für Artikel)

(ArtNr: numeric(6), Bez: varchar(40), Weingut: varchar(30) , ... )

Man schreibt auch gerne kurz:

Artikel( ArtNr, Bez, Weingut, ... )

### Bemerkung:

Die Reihenfolge der Attribute spielt keine Rolle, d.h.

A1(ArtNr, Bez, Weingut) und A2(Weingut, ArtNr, Bez)  
haben *dasselbe* Relationsschema.

# Relation

Eine **Relation**  $R$  besteht aus einem Relationsschema  $R_S$  und einer Menge  $R_T$  von Tupeln, die dem Relationsschema entsprechen.

## Beispiel (Relation für Artikel)

ArtNr: numeric(6)	Bez: varchar(40)	Weingut: varchar(30)	...
100001	Les Châteaux	Louis Max	...
100002	Chablis	Louis Max	...
...	...	...	...

Der **Grad** einer Relation  $R$  ist die Anzahl der Attribute im Relationsschema  $R_S$

# Zusammenfassung soweit

**Wert** ein in Raum und Zeit unveränderliches Merkmal  
z.B. die Zahl 12 oder der String 'Hans' ...

**Datentyp** eine Menge von Werten z.B. varchar(40)

**Attribut** ein benannter Datentyp, z.B Name: varchar(40)

**Tupel** ein durch Attribute „indiziertes“ Tupel von Werten,  
z.B.

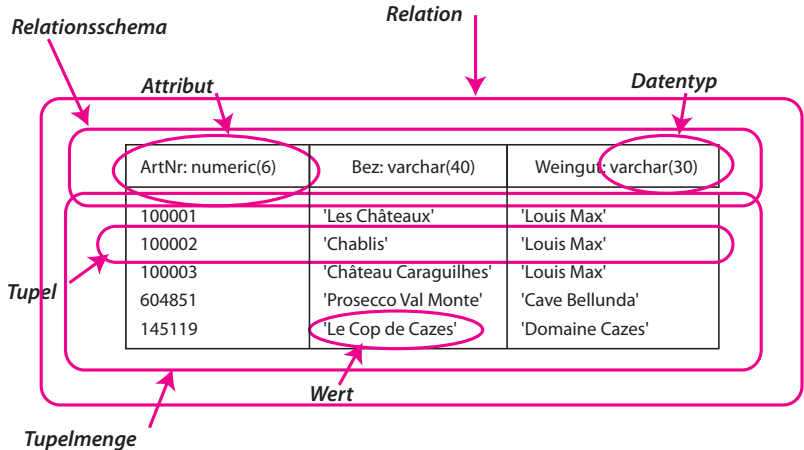
[Name = 'Schneider', Vorname = 'Hans', GebDat = '2001-01-05']

**Relationsschema** eine Menge von Attributen

**Tupelmenge** eine Menge von Tupeln zu einem gegebenen  
Relationsschema

**Relation** ein Relationsschema (auch Relationskopf genannt)  
zusammen mit einer Tupelmenge zu diesem Schema

# Zusammenfassung soweit (Illustration)





# Relationen und Relationsvariablen

Eine Relation  $R$  ist ein **Wert**. Man kann  $R$  in einer Datenbank speichern, dies entspricht der Zuweisung des Werts  $R$  zu einer Variablen.

Eine **Relationsvariable** (kurz: **RelVar**) ist eine Variable, die als Wert eine Relation aufnehmen kann.

Eine Relationsvariable  $V$  hat ein Relationsschema  $V_S$ . In der Relationsvariablen  $V$  können nur Relationen  $R$  gespeichert werden, die dasselbe Relationsschema haben, d.h. es muss gelten:  $R_S = V_S$ .

Für die **Zuweisung** einer Relation  $R$  an eine Relationsvariable  $V$  schreiben wir:

$$V := R$$

# Datenbank (Erste Definition)

Eine **Datenbank** ist eine Menge von Relationsvariablen.

Das **Datenbankschema** besteht aus den Relationsschemata der Relationsvariablen.

Der **Datenbankzustand** besteht aus den Relationen, die zu einem bestimmten Zeitpunkt den Relationsvariablen zugewiesen sind, d.h. in der Datenbank gespeichert sind.

Im SQL-„Jargon“, aber auch in Lehrbüchern zu Datenbanksystemen werden Relationen *und* RelVars als Tabellen bzw. Relationen bezeichnet.

Man muss also aus dem Kontext entnehmen, ob Relationen (Werte) oder Relationsvariablen (Variablen) gemeint sind.

# Eigenschaften von Relationen I

- Die Tupelmeng e einer Relation ist eine Menge, hat also keine Duplikate und keine definierte Reihenfolge  
(In SQL kann eine Tabelle Duplikate haben)
- Die Attribute im Relationsschema einer Relation sind eine Menge, d.h. es gibt keine Duplikate und keine definierte Reihenfolge  
(In SQL wird die Reihenfolge der Attribute in `create table` als definierte Reihenfolge der Attribute genommen)

# Eigenschaften von Relationen II

- Das Relationsschema ist Bestandteil der Relation, nicht nur die Tupelmenge  
(In manchen Lehrbüchern wird nur die Tupelmenge genommen)
- Man kann das Relationsschema einer Relation oder RelVar als *Prädikat* auffassen. Die RelVar enthält also eine Menge von wahren Aussagen zu diesem Prädikat.  
(Das Prädikat ergibt sich nicht einfach aus dem Relationsschema, sondern enthält zusätzliche Informationen)

# Inhalt

- Relationen und relationale Algebra
  - Motivation: Relationen in der Mathematik
  - Relationen
  - Relationen und Relationsvariablen
- Operatoren der relationalen Algebra
  - Übersicht
  - Operatoren im Detail
  - Gesetze der relationalen Algebra
- Datenbanken und Integritätsbedingungen
  - Integritätsbedingungen für eine Relationsvariable
  - Integritätsbedingungen für mehrere Relationsvariablen
  - Definition Datenbank

# Übersicht

- 1  $\sigma$  (sigma) Selektion, Restriktion
- 2  $\pi$  (pi) Projektion
- 3  $\rho$  (rho) Rename, Umbenennung
- 4  $\pi$  (pi) erweiterte Projektion
- 5  $\cup$  Vereinigung
- 6  $\cap$  Durchschnitt
- 7  $-$  Differenz
- 8  $\times$  Kartesisches Produkt, Kreuzprodukt
- 9  $\bowtie$  Natural Join/Natürlicher Verbund
- 10  $\bowtie_{\theta}$  Theta-Join/Equi-Join
- 11  $\gamma$  (gamma) Gruppierung
- 12  $\tau$  (tau) Sortierung

# $\sigma$ (sigma) Selektion/Restriktion

Beispiel:

$\sigma_{(Weingut='LouisMax')}(Artikel)$

Definition:

$\sigma_C(R)$ , mit

$R$  ist eine Relation (oder ein Ausdruck, der eine Relation ergibt)  
 $C$  ist eine Bedingung, in der vorkommen können: Attributnamen von  $R_S$ , Werte, Operatoren der Datentypen der Attribute, Vergleichsoperatoren, logische Operatoren wie `and`, `or`, `not`

# $\sigma$ (sigma) Selektion/Restriktion

Ergebnis:

$\sigma_C(R)$  hat

dasselbe Schema  $R_S$  wie  $R$  und

als Tupelmenge  $\{t \in R_T / t \text{ erfüllt } C\}$

in SQL:

$\sigma_C(R) \hat{=} \text{ select } * \text{ from } R \text{ where } C$



# $\pi$ (pi) Projektion

Beispiel:

$\pi_{(Bez, Weingut)}(Artikel)$

Definition:

$\pi_L(R)$ , mit

$R$  ist eine Relation (oder ein Ausdruck, der eine Relation ergibt)

$L$  ist eine mit Kommata getrennte Liste von (paarweise verschiedenen) Attributen von  $R$

# $\pi$ (pi) Projektion

Ergebnis:

$\pi_L(R)$  hat

als Schema die Menge der Attribute in  $L$  und

als Tupelmenge alle Tupel in  $R$  bezüglich dieser Attribute

in SQL:

$\pi_L(R) \cong \text{select distinct } L \text{ from } R$

## $\rho$ (rho) Rename/Umbenennung

Beispiel:

$$\rho_{(Nr, Text)}(\pi_{(ArtNr, Bez)}(Artikel))$$

Definition:

$\rho_{(b_1, b_2, \dots, b_n)}(R)$ , mit

$R$  ist eine Relation (oder ein Ausdruck, der eine Relation ergibt) mit dem Relationsschema  $R(a_1, a_2, \dots, a_n)$  und somit dem Grad  $n$ . Die  $b_i$  sind paarweise verschiedene Namen für Attribute.

## $\rho$ (rho) Rename/Umbenennung

### Ergebnis:

$\rho(b_1, b_2, \dots, b_n)(R)$  hat

als Relationsschema die Menge der Attribute  $\{b_1, b_2, \dots, b_n\}$  und  
als Tupelmengende dieselben Tupel wie  $R$  jedoch bezüglich dieser  
Attribute

### in SQL:

$\rho(b_1, b_2, \dots, b_n)(R) \cong$

`select a1 as b1, a2 as b2, ..., an as bn from R`

# $\pi$ (pi) Erweiterte Projektion

Beispiel:

$\pi(\text{Bez} || ' \sqcup \text{von} \sqcup ' || \text{Weingut} \rightarrow \text{Wein})(\text{Artikel})$

Definition:

$\pi_L(R)$ , mit

$R$  ist eine Relation (oder ein Ausdruck, der eine Relation ergibt)

$L$  ist eine Liste bestehend aus Elementen folgender Art:

Attribute von  $R$ , Ausdrücke wie  $a \rightarrow b$  mit Attributnamen  $a$  in  $R_S$  und Bezeichnungen  $b$ , Ausdrücken wie  $E \rightarrow z$  mit einem Ausdruck

für eine Berechnung  $E$  sowie einer Bezeichnung  $z$  für das Ergebnis

Die Namen rechts von den Pfeilen müssen paarweise verschieden sein

## $\pi$ (pi) Erweiterte Projektion

Ergebnis:

$\pi_L(R)$  hat

als Relationsschema die Menge der Attribute  $\{z_1, z_2, \dots, z_m\}$ , wenn  $L = E_1 \rightarrow z_1, E_2 \rightarrow z_2, \dots, E_m \rightarrow z_m$  ist, und

als Tupelmenge die Tupel, die sich durch Berechnung der Ausdrücke  $E_1, E_2, \dots, E_m$  aus den Tupeln von  $R$  ergeben.

in SQL:

$\pi_L(R) \cong$

`select distinct E1 as z1, E2 as z2, ..., Em as zm from R`

# ∪ Vereinigung

Beispiel:

$$\pi_{(Name)}(Kunde) \cup \pi_{(Firma \rightarrow Name)}(Lieferant)$$

Definition:

$R \cup S$ , mit

$R, S$  sind Relationen (oder Ausdrücke, die Relationen ergeben)

$R$  und  $S$  haben dasselbe Relationsschema, d.h.  $R_S = S_S$

## ∪ Vereinigung

Ergebnis:

$R \cup S$  hat

als Relationsschema das Relationsschema  $R_S$  von  $R$  und  
als Tupelmenge die Vereinigung der Tupel von  $R$  und  $S$ , d.h.  
 $(R \cup S)_T = R_T \cup S_T$ .

in SQL:

$R \cup S \triangleq$  `select * from R union select * from S`



# $\cap$ Durchschnitt

Beispiel:

$$\pi_{(Ort)}(Kunde) \cap \pi_{(Ort)}(Lieferant)$$

Definition:

$R \cap S$ , mit

$R, S$  sind Relationen (oder Ausdrücke, die Relationen ergeben)

$R$  und  $S$  haben dasselbe Relationsschema, d.h.  $R_S = S_S$

## $\cap$ Durchschnitt

Ergebnis:

$R \cap S$  hat

als Relationsschema das Relationsschema  $R_S$  von  $R$  und  
als Tupelmenge den Durchschnitt der Tupel von  $R$  und  $S$ , d.h.  
 $(R \cap S)_T = R_T \cap S_T$ .

in SQL:

$R \cap S \triangleq$  `select * from R intersect select * from S`

## – Differenz

Beispiel:

$$\pi_{(Ort)}(Kunde) - \pi_{(Ort)}(Lieferant)$$

Definition:

$R - S$ , mit

$R, S$  sind Relationen (oder Ausdrücke, die Relationen ergeben)

$R$  und  $S$  haben dasselbe Relationsschema, d.h.  $R_S = S_S$

## – Differenz

Ergebnis:

$R - S$  hat

als Relationsschema das Relationsschema  $R_S$  von  $R$  und  
als Tupelmenge die Differenz der Tupel von  $R$  und  $S$ , d.h.  
 $(R - S)_T = R_T - S_T$ .

in SQL:

$R - S \triangleq$  `select * from R except select * from S`

## × Kartesisches Produkt/Kreuzprodukt

Beispiel:

$Kunde \times \pi_{(AuftrNr)}(Auftrag)$

Definition:

$R \times S$ , mit

$R, S$  sind Relationen (oder Ausdrücke, die Relationen ergeben)

$R$  und  $S$  haben *keine* gemeinsamen Attribute, d.h.  $R_S \cap S_S = \emptyset$

## × Kartesisches Produkt/Kreuzprodukt

Ergebnis:

$R \times S$  hat

als Relationsschema die Attribute beider Relationen, d.h.

$(R \times S)_S = R_S \cup S_S$  und

als Tupelmenge die Kombination aller Tupel von  $R_T$  mit jedem von  $S_T$ :  $(R \times S)_T = \{[r \circ s] / r \in R_T, s \in S_T\}$ .

in SQL:

$R \times S \triangleq$

`select * from R cross join S (SQL92)`

`select * from R, S (SQL89)`

## ⋈ Natürlicher Verbund/Natural Join

Beispiel:

*Kunde* ⋈ *Auftrag*

Definition:

$R \bowtie S$ , mit

$R, S$  sind Relationen (oder Ausdrücke, die Relationen ergeben)

## ⋈ Natürlicher Verbund/Natural Join

Ergebnis:

$R \bowtie S$  hat

als Relationsschema die Vereinigung der Attribute beider

Relationen, d.h.  $(R \bowtie S)_S = R_S \cup S_S$  und

als Tupelmengende die Kombination aller Tupel von  $R_T$  mit denen von

$S_T$  die an den gemeinsamen Attributen  $G = R_S \cap S_S$

übereinstimmen:

$$(R \bowtie S)_T = \{[r \circ s] / r \in R_T, s \in S_T \text{ und } \pi_G(r) = \pi_G(s)\}.$$

in SQL:

$R \bowtie S \triangleq$

```
select * from R natural join S
```

```
select * from R join S using (g1, g2, ...)
```



## $\bowtie_{\theta}$ Theta-Join bzw. Equi-Join

Beispiel:

$Kunde \bowtie_{(KndNr \neq AKndNr)} \pi_{(KndNr \rightarrow AKndNr, AuftrNr)}(Auftrag)$

$Kunde \bowtie_{(KndNr = AKndNr)} \pi_{(KndNr \rightarrow AKndNr, AuftrNr)}(Auftrag)$

Definition:

$R \bowtie_{\theta} S$ , mit

$R, S$  sind Relationen (oder Ausdrücke, die Relationen ergeben) und

$R_S \cap S_S = \emptyset$

## $\bowtie_{\theta}$ Theta-Join bzw. Equi-Join

Ergebnis:

$$R \bowtie_C S = \sigma_C(R \times S)$$

Ist  $C$  eine Bedingung der Form  $a = b$ , dann spricht man vom **Equi-Join**.

Ist  $C$  eine andere Bedingung etwa  $a \neq b$  oder  $a < b$ , allgemein  $a \theta b$  für einen Operator  $\theta$ , dann spricht man vom **Theta-Join**.

in SQL:

$$R \bowtie_C S \triangleq$$

```
select * from R join S on C
```

```
select * from R cross join S where C
```

# $\gamma$ (gamma) Gruppierung und Aggregation

## Beispiel:

$\gamma(\text{AuftrNr}, \text{sum}(\text{Anz}) \rightarrow \text{Gesamtmenge})(\text{AuftrPos})$

## Definition:

$\gamma_L(R)$ , mit

$R$  ist eine Relation (oder ein Ausdruck, der eine Relation ergibt)

$L$  ist eine Liste von Elementen, die sein können:

- (1) ein Attribut der Relation  $R$ , das für die Gruppierung verwendet werden soll (Gruppierungsattribute  $L_G$ )
- (2) eine Aggregatfunktion zusammen mit einem Attributnamen für das Ergebnis der Aggregation (Aggregatfunktionen  $L_A$ )

# $\gamma$ (gamma) Gruppierung und Aggregation

## Ergebnis:

$\gamma_L(R)$  hat

als Relationsschema die Attribute in  $L$  und die Tupelmengen, die so gebildet wird:

1. Die Relation  $R$  wird in Gruppen von Tupeln eingeteilt, die bezüglich der Gruppierungsattribute  $L_G$  übereinstimmen
2. Für jede Gruppe wird nun genau *ein* Tupel gebildet, indem die Aggregatfunktionen  $L_A$  auf die Gruppe angewendet werden.

## in SQL:

$\gamma_L(R) \cong$

`select L from R group by  $L_G$`

# $\tau$ (tau) Sortierung

Beispiel:

$\tau_{(ArtNr)}(Artikel)$

Definition:

$\tau_L(R)$ , mit

$R$  ist eine Relation (oder ein Ausdruck, der eine Relation ergibt)

$L$  ist eine Liste von Attributen von  $R$

## $\tau$ (tau) Sortierung

Ergebnis:

$\tau_L(R)$  hat

als Relationsschema das von  $R$ , d.h.  $R_S$  und

als Tupelmengende die Tupel von  $R$  lexikographisch sortiert nach den Attributen in  $L$ .

Das Ergebnis ist eine Liste, d.h. die Reihenfolge spielt eine Rolle – somit keine Relation mehr im engeren Sinne.

in SQL:

$\tau_L(R) \cong \text{select } * \text{ from } R \text{ order by } L$

# Gesetze der relationalen Algebra

In der relationalen Algebra gelten eine Vielzahl von **Gesetzen**. Diese kann man verwenden um Ausdrücke äquivalent umzuformen. Die **regelbasierte Optimierung** in Datenbankmanagementsystemen basiert auf diesen Gesetzen.

Beispiele:

$$\sigma_{C_1}(\sigma_{C_2}(R)) = \sigma_{C_2}(\sigma_{C_1}(R))$$

$$\sigma_{C_1}(\sigma_{C_2}(R)) = \sigma_{C_1 \text{ and } C_2}(R)$$

Wenn sich  $C$  nur auf  $S$  bezieht, gilt:

$$\sigma_C(R \bowtie S) = R \bowtie \sigma_C(S)$$

# Inhalt

- Relationen und relationale Algebra
  - Motivation: Relationen in der Mathematik
  - Relationen
  - Relationen und Relationsvariablen
- Operatoren der relationalen Algebra
  - Übersicht
  - Operatoren im Detail
  - Gesetze der relationalen Algebra
- Datenbanken und Integritätsbedingungen
  - Integritätsbedingungen für eine Relationsvariable
  - Integritätsbedingungen für mehrere Relationsvariablen
  - Definition Datenbank



# Schlüssel

Ein **Superschlüssel** ist eine Menge von Attributen einer RelVar  $V$  für die gilt:

Wenn zwei Tupel an diesen Attributen übereinstimmen, dann sind sie identisch.

Jede Relationsvariable hat einen Superschlüssel, nämlich die Menge ihrer Attribute. (In SQL ist das nicht notwendigerweise so)

Ein **Schlüssel** ist ein minimaler Superschlüssel, d.h. man kann kein Attribut weglassen ohne die Schlüsseleigenschaft zu verletzen.

## Beispiel

In der RelVar *Artikel* ist  $\{ArtNr, Bez\}$  ein Superschlüssel, und  $\{ArtNr\}$  ein Schlüssel.

# Primärschlüssel

Eine Relationsvariable kann mehrere Schlüssel haben  
(*Schlüsselkandidaten*)

Für eine Relationsvariable wählt man *einen* dieser  
Schlüsselkandidaten aus – den **Primärschlüssel**

in SQL:

```
create table Artikel (  
    ArtNr numeric(6) primary key,  
    ....  
);  
create table AuftrPos (  
    AuftrNr    numeric(8) references Auftrag(AuftrNr),  
    Anzahl     integer not null,  
    ArtNr      numeric(6) not null references Artikel(ArtNr),  
    primary key (AuftrNr, ArtNr)  
);
```

# Wichtige Integritätsbedingungen

- **Entitätsintegrität** = Die Werte des Primärschlüssels dürfen niemals `null` sein (in SQL automatisch erfüllt)
- **Eindeutigkeit von Werten, Schlüssel** = Die Werte bestimmter Attribute müssen eindeutig sein (in SQL: `UNIQUE`-Constraint)
- **Verbot von `null`** (in SQL: `NOT-NULL`-Constraint)
- **Bedingungen für Werte** z.B. über Wertebereiche o.ä. (in SQL: `Check`-Constraint)

# Fremdschlüssel

Ein **Fremdschlüssel** in der RelVar  $S$  ist eine Menge von Attributen im Relationsschema  $S_S$ , die eine Schema  $R_S$  referenziert, wenn gilt:

- 1 Die zum Fremdschlüssel korrespondierenden Attribute in  $R_S$  haben denselben Datentyp in  $S_S$
- 2 Ein Tupel kann in  $S$  nur vorkommen, wenn es die zum Fremdschlüssel korrespondierenden Werte in einem Tupel von  $R$  gibt. (Ausnahme in SQL: man kann `null` im Fremdschlüssel erlauben)

# Referenzielle Integrität

**Referenzielle Integrität** besteht darin, dass das Datenbankmanagementsystem Fremdschlüssel überwacht und sicherstellt, dass bei jeder Veränderung von Daten die Fremdschlüsselbeziehung erhalten bleibt.

In SQL kann man festlegen, was beim Löschen oder Ändern eines Schlüssels passieren soll:

- NO ACTION/RESTRICT = Aktion, die referenzielle Integrität verletzen würde, wird nach/vor Änderung überprüft und ggfs. nicht erlaubt
- CASCADE = Änderung an einem Schlüssel wird an den referenzierenden Fremdschlüssel weitergereicht
- SET NULL = Nicht mehr gültige Fremdschlüssel werden auf NULL gesetzt
- SET DEFAULT = Nicht mehr gültige Fremdschlüssel werden auf ihren Default-Wert gesetzt

# Datenbank (Vollständige Definition)

Eine **Datenbank** ist eine Menge von Relationsvariablen zusammen mit einer Menge von Integritätsbedingungen.

Das **Datenbankschema** besteht aus den Relationsschemata der Relationsvariablen.

Der **Datenbankzustand** besteht aus den Relationen, die zu einem bestimmten Zeitpunkt den Relationsvariablen zugewiesen sind, d.h. in der Datenbank gespeichert sind.

Das **Datenbankmanagementsystem** garantiert, dass in jedem Datenbankzustand die Integritätsbedingungen eingehalten werden, d.h. dass die Datenbank in einem konsistenten Zustand ist.