

# Datenbanken 1

## Relationale Algebra

Nikolaus Augsten

nikolaus.augsten@sbg.ac.at  
FB Computerwissenschaften  
Universität Salzburg



<http://dbresearch.uni-salzburg.at>

Sommersemester 2018

Version 9. April 2018

- 1 Relationale Algebra
  - Elementare Operatoren
  - Zusätzliche Operatoren
  - Erweiterte Relationale Algebra
  - Relationale Manipulationssprache

# Literatur und Quellen

**Lektüre** zum Thema “Relationale Algebra”:

- Kapitel 3 (3.4) aus Kemper und Eickler: Datenbanksysteme: Eine Einführung. 9. Auflage, Oldenbourg Verlag, 2013.
- Kapitel 6 (6.1) aus Silberschatz, Korth, and Sudarashan: Database System Concepts, McGraw Hill, 2011.

**Literaturquellen**

- Elmasri and Navathe: Fundamentals of Database Systems. Fourth Edition, Pearson Addison Wesley, 2004.

**Danksagung** Die Vorlage zu diesen Folien wurde entwickelt von:

- Michael Böhlen, Universität Zürich, Schweiz
- Johann Gamper, Freie Universität Bozen, Italien

# Inhalt

- 1 Relationale Algebra
  - Elementare Operatoren
  - Zusätzliche Operatoren
  - Erweiterte Relationale Algebra
  - Relationale Manipulationssprache

# Relationale Algebra

- Die relationale Algebra ist eine **prozedurale Anfragesprache**.
- Besteht aus **sechs (notwendigen) Operatoren**:
  - Selektion:  $\sigma$
  - Projektion:  $\pi$
  - Mengenvereinigung:  $\cup$
  - Mengendifferenz:  $-$
  - Kartesisches Produkt:  $\times$
  - Umbenennung:  $\rho$  (Hilfsoperation)
- Die relationale Algebra ist **abgeschlossen**:
  - Argumente der Operatoren sind (ein oder zwei) Relationen.
  - Ergebnis der Operatoren ist wieder eine Relation.

# Syntaktische Konventionen

- Es ist hilfreich bei der Namensgebung systematisch zu sein.
- Wir verwenden folgende Regeln.
  - Tabellennamen: Großschreibung und Plural  
Beispiele: **Vorlesungen**, **Studenten**, **Module**, **R**, **S**
  - Attributnamen: Großschreibung und Singular  
Beispiele: **Semester**, **Jahr**, **Name**, **A**, **B**
  - Konstanten (Werte):
    - Numerische Werte: **12**, **17.6**
    - Zeichenketten: durch Hochkommas begrenzen  
Beispiele: **'Martin'**, **'Mehr als ein Wort'**
- Es gibt keinen einheitlichen Standard; verschiedene Lehrbücher verwenden verschiedene Notationen

# Elementare Operatoren

- Selektion  $\sigma$
- Projektion  $\pi$
- Mengenvereinigung  $\cup$
- Mengendifferenz  $-$
- Kartesisches Produkt  $\times$
- Umbenennung  $\rho$

# Selektion

- Notation:  $\sigma_p(R)$  (sigma)
- Selektionsprädikat  $p$  ist aus folgenden Elementen aufgebaut:
  - Attributnamen der Argumentrelation  $R$  oder Konstanten als Operatoren
  - arithmetische Vergleichsoperatoren ( $=, <, \leq, >, \geq$ )
  - logische Operatoren:  $\wedge$  (**and**),  $\vee$  (**or**),  $\neg$  (**not**)
- $p(t)$ ,  $t \in R$  heißt: Prädikat  $p$  ist für Tupel  $t$  aus Relation  $R$  erfüllt.
- Definition:  $t \in \sigma_p(R) \Leftrightarrow t \in R \wedge p(t)$
- Beispiel:  $\sigma_{FiName='Brugg'}(Konten)$
- Beispiel:  $\sigma_{A=B \wedge D > 5}(R)$

R

A	B	C	D
$\alpha$	$\alpha$	1	7
$\alpha$	$\beta$	5	7
$\beta$	$\beta$	12	3
$\beta$	$\beta$	23	10

 $\sigma_{A=B \wedge D > 5}(R)$ 

A	B	C	D
$\alpha$	$\alpha$	1	7
$\beta$	$\beta$	23	10



# Projektion

- **Notation:**  $\pi_{A_1, \dots, A_k}(R)$  ( $\pi$ )
- $A_1, A_2, \dots, A_k$  sind Attribute von  $R$  und heißen **Projektionsliste**
- **Definition:**  $t \in \pi_{A_1, \dots, A_k}(R) \Leftrightarrow \exists x(x \in R \wedge t = x[A_1, \dots, A_k])$ , wobei  $x[A_1, A_2, \dots, A_k]$  ein neues Tupel bezeichnet, welches für die Werte von  $A_i$ ,  $1 \leq i \leq k$ , die Werte der entsprechenden Attribute von  $x$  annimmt (alle Attribute  $A_i$  müssen in  $x$  vorkommen)
- **Beachte:** Allfällige Duplikate (identische Tupel), die sich aus der Projektion ergeben, müssen entfernt werden.
- Beispiel:  $\pi_{KoNr, Guthaben}(Konten)$
- Beispiel:  $\pi_{A,C}(R)$

R

A	B	C
$\alpha$	10	1
$\alpha$	20	1
$\beta$	30	1
$\beta$	40	2

$\pi_{A,C}(R)$

A	C
$\alpha$	1
$\beta$	1
$\beta$	2

# Mengenvereinigung

- Notation:  $R \cup S$
- Definition:  $t \in (R \cup S) \Leftrightarrow t \in R \vee t \in S$
- $R \cup S$  ist nur definiert, wenn  $r$  und  $s$  das gleiche Schema haben (**union compatible**). Namensdifferenzen können durch explizites Umbenennung der Attribute eliminiert werden (s. weiter unten).
- Beispiel:  $\pi_{KuName}(Kontoinhaber) \cup \pi_{KuName}(Kreditnehmer)$
- Beispiel:  $R \cup S$

R	
A	B
$\alpha$	1
$\alpha$	2
$\beta$	1

S	
A	B
$\alpha$	2
$\beta$	3

$R \cup S$	
A	B
$\alpha$	1
$\alpha$	2
$\beta$	1
$\beta$	3

# Mengendifferenz

- Notation:  $R - S$
- Definition:  $t \in (R - S) \Leftrightarrow t \in R \wedge t \notin S$
- Die Argumentrelationen der Mengendifferenz müssen das **gleiche Schema** haben (union compatible).
- Beispiel:  $R - S$

$R$	
A	B
$\alpha$	1
$\alpha$	2
$\beta$	1

$S$	
A	B
$\alpha$	2
$\beta$	3

$R - S$	
A	B
$\alpha$	1
$\beta$	1

# Kartesisches Produkt (Kreuzprodukt)

- Notation:  $R \times S$
- Definition:  $t \in (R \times S) \Leftrightarrow \exists x, y (x \in R \wedge y \in S \wedge t = x \circ y)$
- $\circ$  bezeichnet die Konkatenation von Tupeln:  $[1, 2] \circ [5] = [1, 2, 5]$
- Die Attribute von  $R$  und  $S$  müssen **unterschiedliche Namen** haben.
- Beispiel:  $R \times S$

$R$

A	B
$\alpha$	1
$\beta$	2

$S$

C	D	E
$\alpha$	10	a
$\beta$	10	a
$\beta$	20	b
$\gamma$	10	b

$R \times S$

A	B	C	D	E
$\alpha$	1	$\alpha$	10	a
$\alpha$	1	$\beta$	10	a
$\alpha$	1	$\beta$	20	b
$\alpha$	1	$\gamma$	10	b
$\beta$	2	$\alpha$	10	a
$\beta$	2	$\beta$	10	a
$\beta$	2	$\beta$	20	b
$\beta$	2	$\gamma$	10	b

# Umbenennung

- Erlaubt es den **Namen der Relation und der Attribute** eines algebraischen Ausdrucks  $E$  zu spezifizieren.
- Wird auch verwendet um **Namenskonflikte aufzulösen** (z.B., in Mengenvereinigung oder Kreuzprodukt)
- Verschiedene **Variationen** ( $E$  ist ein relationaler Ausdruck):
  - $\rho_R(E)$  ist eine Relation mit Namen  $R$ .
  - $\rho_{R[A_1, \dots, A_k]}(E)$  ist eine Relation mit Namen  $R$  und Attributnamen  $A_1, \dots, A_k$ .
  - $\rho_{[A_1, \dots, A_k]}(E)$  ist eine Relation mit Attributnamen  $A_1, \dots, A_k$ .
- Beispiel:  $\rho_{S[X, Y, U, V]}(R)$

R			
A	B	C	D
$\alpha$	$\alpha$	1	7
$\beta$	$\beta$	23	10

S			
X	Y	U	V
$\alpha$	$\alpha$	1	7
$\beta$	$\beta$	23	10

# Zusammengesetzte Ausdrücke

- **Geschachtelte Ausdrücke:** Da die relationale Algebra abgeschlossen ist, d.h. das Resultat eines Operators der relationalen Algebra ist wieder eine Relation, ist es möglich Ausdrücke zu schachteln.
- Beispiel:  $\sigma_{A=C}(R \times S)$

*R*

A	B
$\alpha$	1
$\beta$	2

*S*

C	D	E
$\alpha$	10	a
$\beta$	10	a
$\beta$	20	b
$\gamma$	10	b

*R* × *S*

A	B	C	D	E
$\alpha$	1	$\alpha$	10	a
$\alpha$	1	$\beta$	10	a
$\alpha$	1	$\beta$	20	b
$\alpha$	1	$\gamma$	10	b
$\beta$	2	$\alpha$	10	a
$\beta$	2	$\beta$	10	a
$\beta$	2	$\beta$	20	b
$\beta$	2	$\gamma$	10	b

$\sigma_{A=C}(R \times S)$

A	B	C	D	E
$\alpha$	1	$\alpha$	10	a
$\beta$	2	$\beta$	10	a
$\beta$	2	$\beta$	20	b

# Integrierte Übung 4.1

- Identifizieren und korrigieren Sie Fehler in den nachfolgenden relationalen Algebra Ausdrücken. Relation  $R$  hat Schema  $sch(R) = [A, B]$ .
- $\sigma_{R.A>5}(R)$
- $\sigma_{A,B}(R)$
- $R \times R$

# Integrierte Übung 4.2

- Identifizieren und korrigieren Sie Fehler in den nachfolgenden relationalen Algebra Ausdrücken. Relation  $Pers$  hat Schema  $sch(Pers) = [Name, Alter, Stadt]$ .
- $\sigma_{Name='Name'}(Pers)$
- $\sigma_{Stadt=Zuerich}(Pers)$
- $\sigma_{Alter > '20'}$



# Beispiel: Banken

Filialen[FiName, Stadt, Umsatz]  
Kunden[KuName, Strasse, Ort]  
Konten[KoNr, FiName, Guthaben]  
Kredite[KrNr, FiName, Betrag]  
Kontoinhaber[KuName, KoNr]  
Kreditnehmer[KuName, KrNo]

## Fremdschlüssel:

- $\pi_{\text{FiName}}(\text{Konten}) \subseteq \pi_{\text{FiName}}(\text{Filialen})$
- $\pi_{\text{FiName}}(\text{Kredite}) \subseteq \pi_{\text{FiName}}(\text{Filialen})$
- $\pi_{\text{KuName}}(\text{Kontoinhaber}) \subseteq \pi_{\text{KuName}}(\text{Kunden})$
- $\pi_{\text{KoNr}}(\text{Kontoinhaber}) \subseteq \pi_{\text{KoNr}}(\text{Konten})$
- $\pi_{\text{KuName}}(\text{Kreditnehmer}) \subseteq \pi_{\text{KuName}}(\text{Kunden})$
- $\pi_{\text{KrNo}}(\text{Kreditnehmer}) \subseteq \pi_{\text{KrNr}}(\text{Kredite})$

# Anfragebeispiele/1

- Jene Kredite die größer als \$1200 sind.

$\sigma_{\text{Betrag} > 1200}(\text{Kredite})$

Filialen[FiName, Stadt, Umsatz]  
Kunden[KuName, Strasse, Ort]  
Konten[KoNr, FiName, Guthaben]  
Kredite[KrNr, FiName, Betrag]  
Kontoinhaber[KuName, KoNr]  
Kreditnehmer[KuName, KrNo]

- Die Nummern jener Kredite die größer als \$1200 sind.

$\pi_{\text{KrNr}}(\sigma_{\text{Betrag} > 1200}(\text{Kredite}))$

- Die Namen aller Kunden die einen Kredit oder ein Konto (oder beides) haben.

$\pi_{\text{KuName}}(\text{Kreditnehmer}) \cup \pi_{\text{KuName}}(\text{Kontoinhaber})$

# Anfragebeispiele/2

- Die Namen aller Kunden die einen Kredit bei der Brugg Filiale haben.
  - Anfrage 1

$$\pi_{KuName}(\sigma_{FiName='Brugg'}(\sigma_{KrNo=KrNr}(Kreditnehmer \times Kredite)))$$

- Anfrage 2

$$\pi_{KuName}(\sigma_{KrNo=KrNr}(\sigma_{FiName='Brugg'}(Kredite)) \times Kreditnehmer)$$

Filialen[ <u>FiName</u> , Stadt, Umsatz]
Kunden[ <u>KuName</u> , Strasse, Ort]
Konten[ <u>KoNr</u> , FiName, Guthaben]
Kredite[ <u>KrNr</u> , FiName, Betrag]
Kontoinhaber[ <u>KuName</u> , <u>KoNr</u> ]
Kreditnehmer[ <u>KuName</u> , <u>KrNo</u> ]

# Anfragebeispiele/3

- Die Namen aller Kunden die einen Kredit bei der Brugg Filiale haben, aber kein Konto bei der Bank.

Filialen[FiName, Stadt, Umsatz]  
Kunden[KuName, Strasse, Ort]  
Konten[KoNr, FiName, Guthaben]  
Kredite[KrNr, FiName, Betrag]  
Kontoinhaber[KuName, KoNr]  
Kreditnehmer[KuName, KrNo]

$\pi_{KuName}(\sigma_{FiName='Brugg'}(\sigma_{KrNo=KrNr}(Kreditnehmer \times Kredite)))$

—

$\pi_{KuName}(Kontoinhaber)$

# Integrierte Übung 4.3

- Gegeben: Relation  $R[A] = \{[1], [2], [3]\}$ . Schreiben Sie einen relationalen Algebra Ausdruck der den größten Wert in  $R$  bestimmt.

# Definition von relationalen Algebra Ausdrücken

- Ein **elementarer Ausdruck** der relationalen Algebra ist eine **Relation** in der Datenbank (z.B. Konten).
- Falls  $E_1$  und  $E_2$  relationale Algebra Ausdrücke sind, dann lassen sich weitere **relationale Algebra Ausdrücke** wie folgt bilden:
  - $E_1 \cup E_2$
  - $E_1 - E_2$
  - $E_1 \times E_2$
  - $\sigma_p(E_1)$ ,  $p$  ist ein Prädikat in  $E_1$
  - $\pi_s(E_1)$ ,  $s$  ist eine Liste mit Attributen aus  $E_1$
  - $\rho_x(E_1)$ ,  $x$  ist der Name für  $E_1$

# Notationsvarianten der Relationalen Algebra

- Im Laufe der Zeit sind **unterschiedliche Notationen** entstanden.
- Notation von Kemper&Eikler (Lehrbuch) unterscheidet sich wie folgt.
- **Qualifizierte Attributnamen**
  - Attributnamen werden durch Voranstellen des Relationsnamen eindeutig gemacht (wo nötig), z.B.,  $R.B$ ,  $S.B$
  - Kreuzprodukt  $R \times S$  ist auch dann erlaubt, wenn  $R$  und  $S$  gleichnamige Attribute haben
  - Beispiele: Gegeben  $R[A, B]$ ,  $S[B, C]$ 
    - $sch(R \times S) = [A, R.B, S.B, C]$
    - $\sigma_{R.B=S.B}(R \times S)$  ist syntaktisch korrekt
- **Umbenennung mit Zuordnung**
  - Syntax von  $\rho$  unterscheidet sich für Relationen und Attribute
  - Relation:  $\rho_R(E)$  benennt relationalen Ausdruck  $E$  mit  $R$
  - Attribut:  $\rho_{A \leftarrow B}(R)$  benennt Attribut  $A$  zu  $B$  um ( $A \in sch(R)$ )

**In der Prüfung ist die Notation aus der Vorlesung zu verwenden.**

# Zusammenfassung: Elementare Operatoren

- Relationale Algebra ist **prozedural** und **abgeschlossen**.
- **Elementare Operatoren**:
  - unär: Selektion  $\sigma$ , Projektion  $\pi$ , Umbenennung  $\rho$
  - binär: Mengenvereinigung  $\cup$ , Mengendifferenz  $-$ , Kreuzprodukt  $\times$
- Ein **relationaler Ausdruck** kann sein:
  - ein elementarer Ausdruck (Relation)
  - eine Kombination von relationalen Ausdrücken, die über relationale Operatoren verbunden sein müssen



# Zusätzliche Operatoren der Relationalen Algebra

- Neben den elementaren Operatoren gibt es **zusätzliche Operatoren**:
  - Mengendurchschnitt  $\cap$
  - Join  $\bowtie$
  - Zuweisung  $\leftarrow$
- Die zusätzlichen Operatoren machen Algebra **nicht ausdrucksstärker**:
  - man kann die zusätzlichen Operatoren mithilfe der elementaren Operatoren ausdrücken
  - deshalb sind die zusätzlichen Operatoren *redundant*
- **Formulierung** häufiger Anfragen wird zum Teil erheblich **vereinfacht**.

# Mengendurchschnitt

- Notation:  $R \cap S$
- Definition:  $t \in (R \cap S) \Leftrightarrow t \in R \wedge t \in S$
- Voraussetzung:  $R$  und  $S$  haben das gleiche Schema
- Beachte:  $R \cap S = R - (R - S)$
- Beispiel:  $R \cap S$

$R$	
A	B
$\alpha$	1
$\alpha$	2
$\beta$	1

$S$	
A	B
$\alpha$	2
$\beta$	3

$R \cap S$	
A	B
$\alpha$	2

# Theta Join (Verbund)/1

- Notation:  $R \bowtie_{\theta} S$
- Annahme:  $R$  und  $S$  sind Relationen.  $\theta$  ist ein Prädikat über den Attributen von  $R$  und  $S$ .
- $R \bowtie_{\theta} S$  ist eine Relation mit einem Schema das aus allen Attributen von  $sch(R)$  und allen Attributen von  $sch(S)$  besteht.
- Beachte:  $R \bowtie_{\theta} S = \sigma_{\theta}(R \times S)$
- Beispiel:
  - $sch(R) = [A, B, D]$  und  $sch(S) = [X, Y, Z]$
  - $R \bowtie_{A=Z} S$
  - Schema des Resultats ist  $[A, B, D, X, Y, Z]$
  - Äquivalent zu:  $\sigma_{A=Z}(R \times S)$

R

A	B	D
$\alpha$	1	a
$\beta$	2	a
$\gamma$	4	b

S

X	Y	Z
1	a	$\alpha$
3	a	$\beta$
3	b	$\epsilon$

$R \bowtie_{A=Z} S$

A	B	D	X	Y	Z
$\alpha$	1	a	1	a	$\alpha$
$\beta$	2	a	3	a	$\beta$

# Theta Join (Verbund)/2

- Beispiel:

- $sch(R) = [A, B, D]$  und  $sch(S) = [X, Y, Z]$
- $R \bowtie_{A=Z \wedge B < X} S$
- Schema des Resultats ist  $[A, B, D, X, Y, Z]$
- Äquivalent zu:  $\sigma_{A=Z \wedge B < X}(R \times S)$

R

A	B	D
$\alpha$	1	a
$\beta$	2	a
$\gamma$	4	b

S

X	Y	Z
1	a	$\alpha$
3	a	$\beta$
3	b	$\epsilon$

$R \bowtie_{A=Z \wedge B < X} S$

A	B	D	X	Y	Z
$\beta$	2	a	3	a	$\beta$

# Natürlicher Join

- Notation:  $R \bowtie S$
- Annahme:  $R$  und  $S$  sind Relationen.
- Der natürliche Join verlangt, dass Attribute die sowohl in  $R$  als auch in  $S$  vorkommen identische Werte haben.
- Das Resultat von  $R \bowtie S$  ist eine Relation mit einem Schema das alle Attribute von  $R$  enthält und alle Attribute von  $S$  die nicht in  $R$  vorkommen.
- Beispiel:
  - $R \bowtie S$  mit  $sch(R) = [A, B, D]$  und  $sch(S) = [B, D, E]$
  - Schema des Resultats ist  $[A, B, D, E]$
  - Äquivalent zu:  $\pi_{A,B,D,E}(\sigma_{B=Y \wedge D=Z}(R \times \rho_{[Y,Z,E]}(S)))$

R		
A	B	D
$\alpha$	1	a
$\beta$	2	a

S		
B	D	E
1	a	$\alpha$
3	a	$\beta$

$R \bowtie S$			
A	B	D	E
$\alpha$	1	a	$\alpha$

# Semi- und Anti-Join

- **Semi-Join:**  $R \bowtie S$ 
  - alle Tupel von  $R$  die in einem natürlichen Join mit  $S$  **mindestens einen** Join-Partner finden.
  - $R \bowtie S = \pi_{sch(R)}(R \Join S)$
- **Anti-Join:**  $R \not\bowtie S$ 
  - alle Tupel von  $R$  die in einem natürlichen Join mit  $S$  **keinen** Join-Partner finden.
  - $R \not\bowtie S = R - (R \bowtie S)$

# Zuweisung

- Die **Zuweisung** ( $\leftarrow$ ) erlaubt es, komplexe Ausdrücke in kleinere übersichtliche Blöcke aufzubrechen.
  - links von  $\leftarrow$  steht eine Variable
  - rechts von  $\leftarrow$  steht ein relationaler Algebra Ausdruck
  - das Resultat rechts von  $\leftarrow$  wird der Variablen links von  $\leftarrow$  zugewiesen
  - komplexe Ausdrücke werden als Sequenz von Zuweisungen geschrieben

# Bankbeispiel Anfragen/1

- Das Konto (bzw. die Konten) mit dem höchsten Kontostand.
- Lösung:

Filialen	[ <u>FiName</u> , Stadt, Umsatz]
Kunden	[ <u>KuName</u> , Strasse, Ort]
Konten	[ <u>KoNr</u> , FiName, Guthaben]
Kredite	[ <u>KrNr</u> , FiName, Betrag]
Kontoinhaber	[ <u>KuName</u> , <u>KoNr</u> ]
Kreditnehmer	[ <u>KuName</u> , <u>KrNo</u> ]

1. Bestimmen jener Konten die **nicht** den höchsten Kontostand haben (indem man jedes Konto mit allen anderen Konten vergleicht)

$$K \leftarrow \pi_{KoNr}(\sigma_{Guthaben < Guth}(Konten \times \rho_{[Nr, Fil, Guth]}(Konten)))$$

2. Mit Hilfe der Mengendifferenz werden jene Konten bestimmt die im ersten Schritt nicht gefunden wurden.

$$Result \leftarrow \pi_{KoNr}(Konten) - K$$



# Bankbeispiel Anfragen/2

Filialen[FiName, Stadt, Umsatz]  
Kunden[KuName, Strasse, Ort]  
Konten[KoNr, FiName, Guthaben]  
Kredite[KrNr, FiName, Betrag]  
Kontoinhaber[KuName, KoNr]  
Kreditnehmer[KuName, KrNo]

- Alle Kunden die sowohl ein Konto als auch einen Kredit haben.

$$\pi_{KuName}(Kreditnehmer) \cap \pi_{KuName}(Kontoinhaber)$$

- Name und Kreditbetrag aller Kunden die einen Kredit haben.

Lösung 1:  $\pi_{KuName, Betrag}(Kreditnehmer \bowtie_{KrNo=KrNr} Kredite)$

Lösung 2:  $\pi_{KuName, Betrag}(\rho_{[KuName, KrNr]}(Kreditnehmer) \bowtie Kredite)$

# Bankbeispiel Anfragen/3

Filialen[FiName, Stadt, Umsatz]  
Kunden[KuName, Strasse, Ort]  
Konten[KoNr, FiName, Guthaben]  
Kredite[KrNr, FiName, Betrag]  
Kontoinhaber[KuName, KoNr]  
Kreditnehmer[KuName, KrNo]

- Kunden die sowohl ein Konto bei der Filiale Chur als auch der Filiale Lanquart haben.

- Lösung:

$$\pi_{KuName}(\sigma_{FiName='Chur'}(Kontoinhaber \bowtie Konten))$$
$$\cap$$
$$\pi_{KuName}(\sigma_{FiName='Lanquart'}(Kontoinhaber \bowtie Konten))$$

# Zusammenfassung: Zusätzliche Operatoren

- **Zusätzliche Operatoren** der relationalen Algebra:
  - Mengendurschnitt  $\cap$
  - Join (theta, natural)  $\bowtie$
  - Zuweisung  $\leftarrow$
- Zusätzliche Operatoren **verändern nicht die Ausdruckstärke** der relationalen Algebra, vereinfachen aber die Anfragen.
- Besonders der **Join** Operator spielt eine große Rolle in der **effizienten Implementierung** der relationalen Algebra in Systemen.

# Operatoren der Erweiterten Relationalen Algebra

Die erweiterten Operatoren **erhöhen die Ausdruckstärke** der relationalen Algebra.

- Verallgemeinerte Projektion  $\pi$
- Gruppierung und Aggregation  $\gamma$
- Äußerer Join (outer join)  $\bowtie$ ,  $\ltimes$ ,  $\ltimes$

# Verallgemeinerte Projektion

- Erlaubt **arithmetische Funktionen** in der Projektionsliste:

$$\pi_{F_1, F_2, \dots, F_n}(E)$$

- $E$  ist ein relationaler Ausdruck.
- $F_1, F_2, \dots, F_n$  sind jeweils arithmetische Ausdrücke, welche Konstanten und Attribute des Schemas von  $E$  enthalten.
- **Beispiel:** Gegeben eine Relation  $Kredite[Kunde, Limit, KreditBetrag]$ , finde heraus, wieviel jeder Kunde noch ausgeben darf:

$$\pi_{Kunde, Limit - KreditBetrag}(Kredite)$$

# Aggregationsfunktionen

- **Aggregationsfunktionen** erhalten eine Multimenge von Werten als Argument und liefern als Ergebnis einen einzigen Funktionswert.
  - avg: Durchschnitt
  - min: kleinster Wert
  - max: größter Wert
  - sum: Summe aller Werte
  - count: Anzahl der Werte (Kardinalität der Menge/Multimenge)
- Elemente der Argumentmenge und Funktionswert sind **atomar**, nicht Tupel.
- **Multimenge** (Menge mit Duplikaten):  $k$ -fache Werte gehen  $k$ -fach in die Berechnung ein.
- **Beispiele**: ( $\{\dots\}_m$  ist eine Multimenge)
  - $\min(\{3, 1, 5, 5\}_m) = 1$
  - $\text{count}(\{3, 1, 5, 5\}_m) = 4$
  - $\text{avg}(\{3, 1, 5, 5\}_m) = 3.5$

# Gruppierung

- **Partitionierung der Tupel** einer Relation gemäß ihrer Werte in einem oder mehreren Attributen.
- **Gruppe (Partition):** Alle Tupel mit identischen Werten in allen Gruppierungsattributen.
- **Hauptzweck:** Aggregation auf Teilen einer Relation (Gruppen)
- **Beispiel:** Gegeben Relation  
 $R = \{[1, 2, 3], [1, 2, 5], [1, 4, 3], [2, 3, 5], [2, 4, 5]\}$  mit Schema  
 $sch(R) = [A, B, C]$ .
  - Gruppierung nach Attribut  $A$  ergibt die Gruppen  $\{[1, 2, 3], [1, 2, 5], [1, 4, 3]\}$  und  $\{[2, 3, 5], [2, 4, 5]\}$
  - Gruppierung nach den Attributen  $A, C$  ergibt die Gruppen  $\{[1, 2, 3], [1, 4, 3]\}$ ,  $\{[1, 2, 5]\}$ ,  $\{[2, 3, 5], [2, 4, 5]\}$

# Gruppierungsoperator

- Die **Gruppierung** der relationalen Algebra:

$$\gamma_{G_1, G_2, \dots, G_m; F_1(A_1), F_2(A_2), \dots, F_n(A_n)}(R)$$

$R$  ist eine Relation:

- Gruppierungsattribute:  $G_1, G_2, \dots, G_m$  ist eine Liste von Attributen aus  $R$ , über die gruppiert wird (kann leer sein)
- Aggregationsfunktionen:  $F_i$  ist eine Aggregationsfunktion
- Aggregierte Attribute:  $A_i$  ist ein Attribut von  $R$
- **Leere Attributliste**: Gruppe besteht aus der gesamten Relation  $R$ .
- **Ergebnis**: Relation mit  $m + n$  Attributen
  - Anzahl der Tupel entspricht Anzahl der Gruppen (ein Tupel pro Gruppe)
  - die Werte der ersten  $m$  Attribute des Tupels einer Gruppe entsprechen  $G_1, G_2, \dots, G_m$  (Wert gleich für alle Tupel in der Gruppe)
  - die letzten  $n$  Attribute entsprechen den Funktionsergebnissen von  $F_i$  über die (Multi-)menge aller Werte von  $A_i$  in der Gruppe



# Beispiel: Gruppierungsoperator

- Relation  $R$ ,  $Res \leftarrow \rho_{[SumC]}(\gamma_{sum(C)}(R))$

$r$			$Res$	
A	B	C	sumC	
$\alpha$	$\alpha$	7		
$\alpha$	$\beta$	7		
$\beta$	$\beta$	3		
$\beta$	$\beta$	10		
			27	

- Gesamteinlagen pro Filiale:

$$Res \leftarrow \rho_{[FiName, SumEinlagen]}(\gamma_{FiName; sum(Guthaben)}(Konten))$$

Konten			$Res$	
$FiName$	$KoNr$	$Guthaben$	$FiName$	$SumEinlagen$
Perryridge	A-102	400		
Perryridge	A-201	900		
Brighton	A-217	750		
Brighton	A-215	750		
Redwood	A-222	700		
			Perryridge	1300
			Brighton	1500
			Redwood	700

# Äußerer Join (Outer Join)

- Erweiterung des Join Operators, welche Informationsverlust verhindert.
- Berechnet Join und fügt die Tupel, die keinen Join-Partner haben, zum Join-Ergebnis hinzu.
- Varianten:
  - (Voller) äußerer Join ( $R \bowtie S$ ): erhält Tupel von  $R$  und  $S$
  - Linker äußerer Join ( $R \ltimes S$ ): erhält nur Tupel von  $R$  (linke Relation)
  - Rechter äußerer Join ( $R \rtimes S$ ): erhält nur Tupel von  $S$  (rechte Relation)
- Verwendet **null Werte**, um die neuen Attribute der Tupel ohne Join-Partner zu füllen.
- Analog zum “normalen” (inneren) Join gibt es einen **natürlichen** und einen **theta** äußeren Join.

# Beispiel: Äußerer Join/1

- Beispiel Relationen:

Kredite

<i>KrNr</i>	<i>FiName</i>	<i>Betrag</i>
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

Kreditnehmer

<i>KuName</i>	<i>KrNr</i>
Jones	L-170
Smith	L-230
Hayes	L-155

- Join (auch "innerer" Join genannt)

*Kredite* ⋈ *Kreditnehmer*

<i>KrNr</i>	<i>FiName</i>	<i>Betrag</i>	<i>KuName</i>
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith

# Beispiel: Äußerer Join/2

- Beispiel Relationen:

Kredite

<i>KrNr</i>	<i>FiName</i>	<i>Betrag</i>
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

Kreditnehmer

<i>KuName</i>	<i>KrNr</i>
Jones	L-170
Smith	L-230
Hayes	L-155

- Linker äußerer Join (erhält Tupel der linken Relation)

*Kredite* ⋈ *Kreditnehmer*

<i>KrNr</i>	<i>FiName</i>	<i>Betrag</i>	<i>KuName</i>
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	<i>null</i>

# Beispiel: Äußerer Join/3

- Beispiel Relationen:

Kredite

<i>KrNr</i>	<i>FiName</i>	<i>Betrag</i>
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

Kreditnehmer

<i>KuName</i>	<i>KrNr</i>
Jones	L-170
Smith	L-230
Hayes	L-155

- Rechter äußerer Join (erhält Tupel der rechten Relation)

*Kredite* ⋈ *Kreditnehmer*

<i>KrNr</i>	<i>FiName</i>	<i>Betrag</i>	<i>KuName</i>
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-155	<i>null</i>	<i>null</i>	Hayes

# Beispiel: Äußerer Join/4

- Beispiel Relationen:

Kredite

<i>KrNr</i>	<i>FiName</i>	<i>Betrag</i>
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

Kreditnehmer

<i>KuName</i>	<i>KrNr</i>
Jones	L-170
Smith	L-230
Hayes	L-155

- (Vollständiger) äußerer Join (erhält Tupel beider Relationen)

*Kredite* ⋈ *Kreditnehmer*

<i>KrNr</i>	<i>FiName</i>	<i>Betrag</i>	<i>KuName</i>
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	<i>null</i>
L-155	<i>null</i>	<i>null</i>	Hayes

# Zusammenfassung: Erweiterte Relationale Algebra

- Erweiterte Relationale Algebra ist **ausdrucksstärker** als elementare relationale Algebra.
- **Verallgemeinerte Projektion  $\pi$** : Arithmetik in Projektionsliste
- **Gruppierung und Aggregation  $\gamma$** : Berechnung über Gruppen von Attributwerten
- **Äußerer Join  $\bowtie$ ,  $\ltimes$ ,  $\ltimes$** : Tupel-erhaltender Join

# Änderung der Datenbank

- Der Inhalt der Datenbank kann mithilfe folgenden Operatoren verändert werden:
  - Löschen (delete)
  - Einfügen (insert)
  - Ändern (update)
- All diese Operationen verwenden den **Zuweisungsoperator**.



# Löschen

- Ausdruck **ähnlich einer Anfrage**, wobei die Ergebnistupel von der Datenbank entfernt werden.
- **Nur ganze Tupel** können entfernt werden; Werte einzelner Attribute können nicht entfernt werden.
- **Löschen** wird in der relationalen Algebra folgendermaßen ausgedrückt:

$$R \leftarrow R - E$$

wobei  $R$  eine Relation ist und  $E$  ein Ausdruck der relationalen Algebra.

# Beispiel: Löschen

Filialen[FiName, Stadt, Umsatz]  
Kunden[KuName, Strasse, Ort]  
Konten[KoNr, FiName, Guthaben]  
Kredite[KrNr, FiName, Betrag]  
Kontoinhaber[KuName, KoNr]  
Kreditnehmer[KuName, KrNo]

- Lösche alle Konten in der Filiale Domplatz:

$$R_1 \leftarrow \sigma_{FiName='Domplatz'}(Konten)$$

$$Konten \leftarrow Konten - R_1$$

$$R_2 \leftarrow \pi_{KuName, KoNr}(R_1 \bowtie Kontoinhaber)$$

$$Kontoinhaber \leftarrow Kontoinhaber - R_2$$

# Einfügen

- Es gibt **zwei Möglichkeiten**, um Daten in die Relation einzufügen:
  - die einzufügenden Tupel explizit angeben
  - eine Anfrage schreiben deren Ergebnis eingefügt werden soll
- **Einfügen** wird in der relationalen Algebra folgendermaßen ausgedrückt:

$$R \leftarrow R \cup E$$

wobei  $R$  eine Relation und  $E$  ein relationaler Ausdruck sind.

- Wird ein **einzelnes, explizites Tupel** eingefügt, ist  $E$  eine konstante Relation die nur ein Tupel enthält.

# Beispiel: Einfügen/1

```
Filialen[FiName, Stadt, Umsatz]
Kunden[KuName, Strasse, Ort]
Konten[KoNr, FiName, Guthaben]
Kredite[KrNr, FiName, Betrag]
Kontoinhaber[KuName, KoNr]
Kreditnehmer[KuName, KrNo]
```

- Füge folgende Information in die Datenbank ein: Kunde Smith eröffnet ein neues Konto mit Nummer A-973 auf der Domplatz Filiale und legt 1200 EUR ein.

$Konten \leftarrow Konten \cup \{['A-973', 'Domplatz', 1200]\}$

$Kontoinhaber \leftarrow Kontoinhaber \cup \{['Smith', 'A-973']\}$

# Beispiel: Einfügen/2

Filialen[FiName, Stadt, Umsatz]  
 Kunden[KuName, Strasse, Ort]  
 Konten[KoNr, FiName, Guthaben]  
 Kredite[KrNr, FiName, Betrag]  
 Kontoinhaber[KuName, KoNr]  
 Kreditnehmer[KuName, KrNo]

- Alle Kreditnehmer der Domplatz Filiale erhalten ein Konto mit 200 EUR Guthaben geschenkt, wobei die Kontonummer des neuen Kontos identisch mit der jeweiligen Kreditnummer ist.

$$R_1 \leftarrow \sigma_{FiName='Domplatz'}(Kreditnehmer \bowtie_{KrNo=KrNr} Kredite)$$

$$Konten \leftarrow Konten \cup \rho_{KoNr, FiName, Guthaben}(\pi_{KrNr, FiName}(R_1) \times \{[200]\})$$

$$Kontoinhaber \leftarrow Kontoinhaber \cup \rho_{KuName, KoNr}(\pi_{KuName, KrNr}(R_1))$$

# Änderung

- **Änderungen** erlauben, einzelne Werte eines Tupels zu ändern, ohne alle Werte ändern zu müssen.
- Kann durch **Löschen und Einfügen** ausgedrückt werden.
  - in realen Systemen ist die Änderungsoperation jedoch oft viel schneller
  - deshalb gibt es einen eigenen Operator
- In relationaler Algebra werden **Änderungen** in der Relation  $R$  durch **Ersetzen der Relation  $R$  durch einen relationalen Ausdruck  $E$**  ausgedrückt:

$$R \leftarrow E$$

- Oft ist  $E$  eine **erweiterte Projektion** über  $R[A_1, A_2, \dots, A_n]$ :

$$R \leftarrow \rho_{[A_1, A_2, \dots, A_n]} \pi_{F_1, F_2, \dots, F_n}(R)$$

wobei  $F_i$

- entweder  $A_i$  ist, falls Attribut  $A_i$  nicht geändert werden soll
- oder eine Funktion, die einen neuen Wert für  $A_i$  festlegt.

# Beispiel: Änderung

- Auszahlung der Zinsen von 5% auf alle Konten:

$$Konten \leftarrow \rho_{[KoNr, FiName, Guthaben]} \pi_{KoNr, FiName, Guthaben * 1.05} (Konten)$$

- Zahle 6% Zinsen für alle Konten mit mehr als 10.000 EUR Guthaben und 5% für alle anderen Konten:

$$Konten \leftarrow$$

$$\rho_{[KoNr, FiName, Guthaben]} \left( \pi_{KoNr, FiName, Guthaben * 1.06} \left( \sigma_{Guthaben > 10000} (Konten) \right) \right)$$

U

$$\rho_{[KoNr, FiName, Guthaben]} \left( \pi_{KoNr, FiName, Guthaben * 1.05} \left( \sigma_{Guthaben \leq 10000} (Konten) \right) \right)$$

# Zusammenfassung

- Relationale Manipulationssprache
  - Löschen, Einfügen, Ändern
  - Wird durch Zuweisungsoperator ( $\leftarrow$ ) und Ausdrücken der relationalen Algebra ausgedrückt.



# Zusammenfassung

## Relationale Algebra:

- **Elementare Operatoren:** notwendig
- **Zusätzliche Operatoren:** redundant (können durch elementare Operatoren ausgedrückt werden)
- **Erweiterte Operatoren:** erhöhen die Ausdruckskraft
- **Manipulationssprache:** Zuweisungsoperator und relationale Algebra