

Übung 4

Relationale Algebra und SQL

2.6 Relationale Algebra

SQL

Datendefinition

Datenmanipulation

Beispiele



- Grundlage (Kalkül) der relationalen Datenbanken
- Untersuchung endlicher n-stelliger Relationen und Verknüpfungen solcher Relationen

Bestandteile und Aufbau der relationalen Algebra:

- Ausgangspunkt: endliche Menge U von **Wertebereichen**, d.h. Mengen D_1, \dots, D_m , also $U = \{D_j\}$
- n-stellige Relationen $\rho \subseteq D_{i_1} \times \dots \times D_{i_n}$
 - sind Teilmengen des kartesischen Produkts mehrerer D_i
 - jedem D_j wird ein eindeutiger **Attributname** a_j zugeordnet
 - falls Tupel $(v_1, \dots, v_n) \in \rho$, so ist v_j der **Wert des Attributs** a_j
 - Beschreibung der Relation ρ durch das **relationale Schema**
 $\mathcal{A} = (a_1:D_{i_1}, \dots, a_n:D_{i_n})$

Endliche Menge von relationalen Schemata heißt **Datenbankschema**

Endliche Menge von Relationen zu einem DB-Schema heißt **Datenbank**



Alternative Bezeichnung eines Elements aus dem Schema:

- über die Position: j -tes Element v_j aus (v_1, \dots, v_n)
- über den Namen: Wert des Attributs a_j in der Menge $(a_1:v_1, \dots, a_n:v_n)$

- $\mathcal{A}(\rho)$ bezeichnet das Schema einer Relation ρ
 $\rho(\mathcal{A})$ bezeichnet eine Relation zum Schema \mathcal{A}

Kompatibel:

Zwei Schemata $\mathcal{A} = (a_1:D_1, \dots, a_n:D_n)$ und $\mathcal{B} = (b_1:E_1, \dots, b_m:E_m)$ heißen kompatibel, wenn es eine Nummerierung der Attribute so gibt, dass $n = m$ und $D_i = E_i$ für $i = 1, \dots, n$

Schlüsselmenge:

Eine Teilmenge M der Attributnamen $N_{\mathcal{A}}$, deren Wertebelegung ein Tupel in der Relation eindeutig kennzeichnet.



▪ **Vereinigung** \cup und **Differenz** \setminus :

geg.: zwei Relationen ρ, σ mit kompatiblen Schemata $\mathcal{A}(\rho)$ und $\mathcal{B}(\sigma)$:

- $\rho \cup \sigma = \{t \mid t \in \rho \text{ oder } t \in \sigma\}$
- $\rho \setminus \sigma = \{t \mid t \in \rho, t \notin \sigma\}$

$\rho \cap \sigma$

A	B	C
12	xy	1

▪ **Durchschnitt** \cap :

Läßt sich mit Hilfe der Differenz \setminus ausdrücken

- $\rho \cap \sigma = \rho \setminus (\rho \setminus \sigma)$

$\rho \cup \sigma$:

Beispiel:

ρ :

A	B	C
12	xy	1
2	ba	7
5	fr	2
1	g	1

σ :

A	B	C
7	as	6
12	xy	1
5	bg	2

A	B	C
12	xy	1
2	ba	7
5	fr	2
1	g	1
7	as	6
5	bg	2

$\rho \setminus \sigma$:

A	B	C
2	ba	7
5	fr	2
1	g	1



- $\mathcal{A} \cap \mathcal{B}$, $\mathcal{A} \cup \mathcal{B}$, $\mathcal{A} \setminus \mathcal{B}$ wie angegeben als DB-Operationen definiert
- **Unterschema $\mathcal{B} \subseteq \mathcal{A}$**
 - $\mathcal{B} = \{a_{i_1}:D_{i_1}, \dots, a_{i_k}:D_{i_k}\} \subseteq \{a_1:D_1, \dots, a_n:D_n\} = \mathcal{A}$, $k \leq n$
- **Projektion $P_{\mathcal{B}}: \rho(\mathcal{A}) \rightarrow \rho(\mathcal{B})$**
 - $\rho(\mathcal{A})$ als Restriktion auf Schema \mathcal{B} , für das $\mathcal{B} \subseteq \mathcal{A}$ gilt, definiert
 - $\rho(\mathcal{B}) = \{(a_{i_1}:v_{i_1}, \dots, a_{i_k}:v_{i_k}) \mid \text{es gibt } (a_1:v_1, \dots, a_n:v_n) \in \rho(\mathcal{A}) \text{ so, daß}$
 $(a_{i_1}:v_{i_1}, \dots, a_{i_k}:v_{i_k}) \subseteq (a_1:v_1, \dots, a_n:v_n) \}$
- **Kartesisches Produkt $\rho \times \sigma$** zweier Relationen $\rho = \rho(\mathcal{A})$, $\sigma = \sigma(\mathcal{B})$
 - Jedes n-Tupel aus ρ wird mit jedem m-Tupel aus σ kombiniert
 - $\rho \times \sigma = \{(v_1, \dots, v_{n-1}, v_n, v_{n+1}, \dots, v_{n+m}) \mid (v_1, \dots, v_n) \in \rho,$
 $(v_{n+1}, \dots, v_{n+m}) \in \sigma\}$



A	B	C
7	as	6
12	xy	1
7	bg	6

↓ Projektion

A	C
7	6
12	1

ρ

A	B	C
1	4	5
2	6	7

$\rho \times \sigma$:

A	B	C
1	4	5
1	6	7
2	4	5
2	6	7



- **Selektion $\text{sel}_\sigma(\rho)$** zweier Relationen $\rho(\mathcal{A})$, $\sigma(\mathcal{B})$ mit $\mathcal{B} \subseteq \mathcal{A}$
 - $\text{sel}_\sigma(\rho) = \{t \in \rho \mid P_{\mathcal{B}}(t) \in \sigma\}$
 - Relation zum Schema \mathcal{A} , die nur solche Tupel enthält, deren Projektion auf $\sigma(\mathcal{B})$ auch in σ vorkommt
 - Spezialfall:
 - Statt expliziter Angabe von σ werden Bedingungen formuliert, die zwischen bestimmten Feldern der Relation ρ gelten müssen
 - Beispiel: $\text{sel}_{a=b}(\rho)$ bezeichnet die Menge aller Tupel mit $v_a = v_b$

- Selektion über drei Relationen $\rho(\mathcal{A})$, $\sigma(\mathcal{B})$, $\tau(\mathcal{C})$ mit $\mathcal{B}, \mathcal{C} \subseteq \mathcal{A}$
 - $\text{sel}_\sigma(\text{sel}_\tau(\rho)) = \{t \in \rho \mid P_{\mathcal{B}}(t) \in \sigma \text{ und } P_{\mathcal{C}}(t) \in \tau\}$
 - Kommutativität: $\text{sel}_\sigma(\text{sel}_\tau(\rho)) = \text{sel}_\tau(\text{sel}_\sigma(\rho))$



σ $\frac{A}{1}$ ρ

A	B
1	xy
2	bg
3	z

$$\text{sel}_{\sigma}(\rho) = \{t \in \rho \mid \mathcal{P}_A(t) \in \sigma\}$$

$$= \{(1, xy)\} =$$

$$\cdot \text{sel}_{A=1}(\rho)$$



- **Verbund** (engl. *join*):
Wichtigste auf 2 Relationen $\rho = \rho(\mathcal{A})$, $\sigma = \sigma(\mathcal{B})$ anwendbare Operation

- **natürlicher Verbund** $\rho \bowtie_{\varphi} \sigma$:
 - φ ist ein Unterschema zu \mathcal{A} und \mathcal{B} ($\varphi \subseteq \mathcal{A}$, $\varphi \subseteq \mathcal{B}$)
 - Idee: nimm alle Tupel aus $\rho \times \sigma$, die in φ übereinstimmen
 - $\rho \bowtie_{\varphi} \sigma = \{ z \mid z \in P_{\mathcal{A} \cup \varphi \cup \mathcal{B}}(\rho \times \sigma),$
 $P_{\mathcal{A}}(z) \in \rho, P_{\mathcal{B}}(z) \in \sigma,$
 $P_{\varphi}(P_{\mathcal{A}}(z)) = P_{\varphi}(P_{\mathcal{B}}(z)) \}$
 - Eigenschaften: kommutativ, assoziativ, idempotent

- **Θ -Verbund** $\rho[X \Theta Y]\sigma$ (engl. *theta-join*):
 - ρ und σ sind Relationen
 - X aus ρ und Y aus σ sind Attribute über demselben Wertebereich
 - Θ -Verbund sind die Tupel aus $\rho \times \sigma$, die Bedingung $X \Theta Y$ erfüllen:
 $\rho[X \Theta Y]\sigma := \text{sel}_{X \Theta Y}(\rho \times \sigma)$



P	X	Y
	5	3
	2	4

Q	Y	Z
	4	6
	1	7

$$\varphi = \{Y\}$$

$$Y_A \leq Y_B$$

~~P~~ × Q

X	Y	Y	Z
2	4	4	6
5	3	4	6
2	4	1	7
5	3	1	7

Arrows from the left point to the second and third rows of the table above.



▪ Division ρ / σ :

- $\rho = \rho(\mathcal{A})$, $\sigma = \sigma(\mathcal{B})$, $\mathcal{B} \subseteq \mathcal{A}$ und $\mathcal{C} = \mathcal{A} \setminus \mathcal{B}$

Forderung: Es kann nur eine Relation eines Schema durch eine Relation eines Unterschemas geteilt werden

- $\rho / \sigma = P_{\mathcal{C}}(\rho) \setminus P_{\mathcal{A}}((P_{\mathcal{C}}(\rho) \times \sigma) \setminus \rho)$
Klammerausdruck: alle Tupel t mit der Eigenschaft
 - $t \notin \rho$, $P_{\mathcal{B}}(t) \in \sigma$
 - Ergebnis der Division: alle Tupel $P_{\mathcal{C}}(t)$ mit $t \in \rho$, $P_{\mathcal{B}}(t) \in \sigma$
- Gleichwertige Definition:
 $\rho / \sigma = P_{\mathcal{C}}(\rho) \setminus P_{\mathcal{C}}((P_{\mathcal{C}}(\rho) \bowtie \sigma) \setminus \rho)$



P	x	y	z	G	X
1	4	7			1
2	5	8			5
3	6	9			

P/G	y	z
	4	7



- SQL ist ein ANSI-Standard
 - theoretisch von fast jeder Datenbank unterstützt
 - praktisch spielen Erweiterungen und nicht standardisierte Bereiche eine große Rolle
- Neuer Sprachgebrauch
 - Eine Wertemenge heißt **Typ**
 - Ein Element eines Tupels heißt **Feld**
 - Ein Tupel heißt Zeile oder **Record**
 - Eine Relation heißt **Tabelle**
 - Erfreulich: Schemata heißen weiterhin Schemata



- Teilsprachen von SQL
 - **Datendefinitionssprache**
 - Tabellen anlegen
 - Tabellen ändern
 - Tabellen löschen
 - **Datenmanipulationssprache**
 - Einfügen von Daten
 - Ändern von Daten
 - Abfragen von Daten
 - Löschen von Daten



- Definition eines Datenbankschemas:
 - **CREATE TABLE** $r(a_1D_1, \dots, a_nD_n)$
mit Relationenname r ,
Feldnamen a_i
Feldtypen D_i
- Problem
 - Grundlegende Feldtypen in fast allen Datenbanken gleich
 - Namen der Typen leider nicht standardisiert
 - Unterschiede zwischen MySQL, MSQL, Access, ...
 - Hier: Namen aus MySQL
- Beispiele für Typen
 - **CHAR** ein einzelnes Zeichen
 - **CHAR(n)** Zeichenkette mit fester Länge n
 - **VARCHAR(n)** Zeichenkette mit Maximallänge n
 - **INTEGER** ganze Zahl mit Vorzeichen
 - **FLOAT** Gleitkommazahl



▪ Aus Sicht der Universität hat ein Student folgende Eigenschaften

- Matrikelnummer
- Name
- Vorname
- Semesteranschrift
- Heimanschrift
- Fachrichtung
- Fachsemester
- Studiensemester

▪ Modellierung als Tabelle

```
CREATE TABLE
Studenten
( Matr_No INTEGER,
  Name VARCHAR(255),
  Vorname VARCHAR(255),
  S_Anschrift " ",
  H_Anschrift " ",
  Fachrichtung " ",
  INTEGER,
  " ")
PRIMARY KEY (Matr_No)
```



- Beobachtung
 - Heim- und Studienadresse sind beides Adressen
 - Studienfächer sind sehr redundant, alternative Schreibweisen möglich
- Idee: Auslagerung in eigene Tabellen
- Zusätze zu Typen nötig:
 - Spalte nicht leer: **NOT NULL**
 - Spalte eindeutig: **UNIQUE**
 - Zugriff beschleunigen: **INDEX**
 - Primärschlüssel: **PRIMARY KEY**
(NOT NULL, UNIQUE, INDEX implizit)
 - Werte automatisch erzeugen: **AUTO_INCREMENT**



- Tabellen für Studenten, Fachrichtungen, Adressen

```
CREATE TABLE Fachrichtungen  
(key INTEGER NOT NULL UNIQUE,  
Fach VARCHAR(255),  
PRIMARY KEY (key))
```

```
CREATE TABLE Adressen  
(key INTEGER NOT NULL UNIQUE,  
Adresse Str VARCHAR(255),  
PRIMARY KEY (key))
```

- Schemata ändern:
 - **ALTER TABLE r ADD a T [FIRST | AFTER b]**
 - mit Relation r, neuem Feldnamen a, neuem Typ T,
 - ggf existierendem Feldnamen b
 - **ALTER TABLE r CHANGE a a' T [FIRST | AFTER b]**
 - mit Relation r, altem Feldnamen a, neuem Feldnamen a', neuem Typ T,
 - ggf existierendem Feldnamen b
 - **ALTER TABLE r DROP a**
 - mit Relation r, zu löschendem Feldnamen a



- Ändern des Studentenschemas

```
ALTER TABLE Studenten CHANGE  
S_Anschrift S_Anschrift INTEGER  
AFTER Vorname
```

```
ALTER TABLE Studenten CHANGE  
H_Anschrift H_Anschrift INTEGER  
AFTER S_Anschrift
```

```
ALTER TABLE Studenten CHANGE  
Fachrichtung Fachrichtung INTEGER  
AFTER H_Anschrift
```



- Eintragen von Zeilen
- Abfragen
- Ändern von Zeilen
- Löschen von Zeilen



- **INSERT INTO r (a₁,...,a_n) VALUES (v₁,..., v_n) , ...**
mit Relation r, Feldern a_i und Werten v_i
(evtl. mehrere Wertetupel)
- Typ der v_j muß mit den Wertebereichen der a_j
übereinstimmen
- Fehlende Werte werden zu NULL
(Ausnahme: AUTO_INCREMENT)
- Einfügen kann fehlschlagen aufgrund von
 - NOT NULL
 - UNIQUE
 - PRIMARY_KEY



- Student Peter Müller, Matrikelnummer 12345, wohnhaft Beispielstr. 12, 76543 Karlsruhe (Heim- und Studienadresse), studiert Informatik im 1. Fachsemester, aber 3. Semester (vorher Germanist)

```
INSERT INTO Adressen (key, Adresse)
VALUES (1, "Beispielstr. 12, 76 543
Karlsruhe")
```

```
INSERT INTO Fachrichtungen (key, Fachrichtung)
VALUES (1, "Informatik")
```

```
INSERT INTO Studenten (MatriNr, Name, Vorname,
S-Anschrift, H-Anschrift, Fachrichtung, Fachsemester,
Studiensemester) VALUES (12345, "Müller", "Peter",
1, 1, 1, 1, 3)
```



- **SELECT** $r_1.f_1$ **AS** $l_1, \dots, r_n.f_n$ **AS** l_n
FROM r'_1 **AS** s_1, \dots, r'_k **AS** s_k
WHERE cond.
ORDER BY l'_1, \dots, l'_n
GROUP BY l''_1, \dots, l''_n
(Teile auch optional)
- **SELECT**: r_i Relationen, f_i Felder, l_i Aliase für die Felder
(manuelle Auflösung von Namenskollisionen)
- **FROM**: r'_1, \dots, r'_k , n Quelltabellen, s_i Umbenennungen
Entspricht Theta-Joins (Umbenennungen = Rollen)
- **WHERE** cond Ausdruck mit Vergleichs- und Boole'schen Operatoren
Entspricht Vergleichsoperationen in Theta-Joins, Selektion
- **ORDER BY** Sortieren
- **GROUP BY** Gruppieren (für Fortgeschrittene)



- Matrikelnummer und Nachname aller Studenten

```
SELECT Studenten.Matr.Nr, Studenten.Name  
FROM Studenten
```

Bsp. (1 2 3 4 5, Müller)

- Wie heißt der Student mit Matrikelnummer 12345?

```
SELECT Studenten.Vorname, Studenten.Nachname  
FROM Studenten  
WHERE Studenten.Matr.Nr = 12345
```

- Welche Matrikelnummern haben den Nachnamen Müller?

```
SELECT Studenten.Matr.Nr  
FROM Studenten  
WHERE Studenten.Name = Müller
```

- Welche Heimadresse hat Peter Müller?

```
SELECT Adressen. Adresse  
FROM Adressen, Studenten  
WHERE Adressen.key = Studenten.H-Auschr.  
AND Studenten.Name = "Müller" AND  
Studenten.Vorname = "Peter"
```

- Welche Studenten wohnen im Klosterweg 18?



- Schema: CREATE TABLE autor (person INTEGER, buch INTEGER)
- Wer ist Koautor von X?
- Ansatz 1: eine Tabelle

```
SELECT autor.person  
FROM autor  
WHERE autor.person = X
```

- Ansatz 2: eine Tabelle in zwei Rollen
(Veröffentlichungen von X, Autoren der Veröffentlichungen)

```
SELECT autor2.person  
FROM   autor, autor AS autor2  
WHERE  autor.person=X AND  
        autor.buch=autor2.buch
```

Problem: Duplikate -> SELECT DISTINCT

