

SQL-Übung Wintersemester 03/04

Übungsleiter: Dipl.-Inform. Tom Gelhausen



Operationen der Relationalen Algebra (Übersicht) 2

Grundoperationen

σ	Selektion	relation \times bedingung \rightarrow relation
π	Projektion	relation \times attributfolge \rightarrow relation
\times	Kartesisches Prod.	relation \times relation \rightarrow relation
\cup	Vereinigung	relation \times relation \rightarrow relation
\setminus	Differenz	relation \times relation \rightarrow relation

Makrooperationen

\cap	Durchschnitt	relation \times relation \rightarrow relation
\bowtie_{θ}	Theta-Verbindung	relation \times bedingung \times relation \rightarrow relation
\bowtie	nat. Verbindung	relation \times relation \rightarrow relation
\div	Division	relation \times relation \rightarrow relation



Joins in SQL 3

- Grundlage der Verbindungsoperationen in der relationalen Algebra ist das kartesische Produkt: $\rho_1 \bowtie_{\theta} \rho_2 = \sigma_{\theta}(\rho_1 \times \rho_2)$
- Kartesisches Produkt ($\rho_1 \times \rho_2$) in SQL:

```
SELECT *
FROM Rel1, Rel2, Rel3;
```

- Die SELECT-Klausel „*“
 - entspricht der Projektionsfunktion $\pi(\rho) = \rho$,
 - es werden also alle Tupelkomponenten zurückgegeben
 - in der Reihenfolge, in der sie in *Rel1* ◦ *Rel2* ◦ *Rel3* stehen
 - mit den Namen mit denen sie in *Rel1* ◦ *Rel2* ◦ *Rel3* stehen (bei Namensgleichheit durch *Rel.-name* ◦ „.“ ◦ *Attrib.-name* qualifiziert)
- Die FROM-Klausel enthält eine Liste der Relationen, aus denen das kartesische Produkt gebildet werden soll
- Die WHERE-Klausel ist implizit wahr und darf als optionaler Bestandteil weggelassen werden.



Joins in SQL 7

- Grundlage der Verbindungsoperationen in der relationalen Algebra ist das kartesische Produkt: $\rho_1 \bowtie \rho_2 = \sigma_{\Theta}(\rho_1 \times \rho_2)$
- Entsprechend wird der Theta-Join in SQL wie folgt ausgedrückt:

```

\rho_1 \bowtie \rho_2 = \text{SELECT } *
\text{FROM } (\text{SELECT } *
\text{FROM } \rho_1, \rho_2)
\text{WHERE } \Theta;

```

Die gebräuchlichere Form ist allerdings:

```

\text{SELECT } *
\text{FROM } \rho_1, \rho_2
\text{WHERE } \Theta;

```

Feststellungen:

- Das Ergebnis einer Anfrage ist selbst wieder eine gültige Relation.
- Die Ergebnisrelation ist allerdings nur temporär und wird nach der Ergebnisverarbeitung wieder verworfen.
- Diese Eigenschaft ermöglicht so genannte **„geschichtete Anfragen“**.

Übung Informatik I WS 03/04 – Dipl.-Inform. Tom Gelhausen SQL

Joins in SQL: Beispiel 8

- Zu welcher Kategorie gehört der Artikel mit der Artikelnummer 4711?

```

\text{SELECT } k.Name
\text{FROM } \text{Artikel } a, \text{Kategorien } k
\text{WHERE } a.Kategorie=k.KatNr
\text{AND } a.ANr=4711;

```

- Aber woher weiß man, dass es „Bestellungen“, „VersandUeber“, „BestellNr“, etc. heißt? Und dass man „VersandUeber“ und „FirmenNr“ vergleichen muss?

⇒ Schema!

Übung Informatik I WS 03/04 – Dipl.-Inform. Tom Gelhausen SQL

Joins in SQL (Beispiel): Schema 9

- Schema:

```

\text{Kategorien}
\text{KatNr}
\text{Name}
\text{Beschreibung}

\text{Artikel}
\text{ANr}
\text{Bezeichnung}
\text{Kategorie}
\text{LNr}
\text{EKPreis}

\text{Lieferant}
\text{LNr}
\text{Name}
\text{Adresse}
\text{Tel}

```

- Relationen werden als Kästen dargestellt
- Attribute, die miteinander verbunden werden „könnten“ sind durch Linien miteinander verbunden
- Was bedeuten „1“ und „∞“ an diesen Linien?

Übung Informatik I WS 03/04 – Dipl.-Inform. Tom Gelhausen SQL

Joins in SQL (Beispiel): Schema 10

▪ Schema:

▪ Dem DBMS vorgegebene Konsistenzbedingung, die die Kopplung zweier Relationen p_1 und p_2 ermöglicht, so dass

- die in p_1 unter der Attributfolge fk auftretenden Werte
- in p_2 unter der Attributfolge pk auftreten,

also: $\pi_{fk}(p_1) \subseteq \pi_{pk}(p_2)$

▪ Es besteht **referentielle Konsistenz** von p_1 , fk nach p_2 , pk .

▪ Ist pk Schlüssel von p_2 , dann nennt man fk **Fremdschlüssel (foreign key)** in p_1 .

Übung Informatik I WS 03/04 – Dipl.-Inform. Tom Gelhausen SQL

Joins in SQL (Beispiel): Schema 11

▪ Schema:

▪ „KatNr“ ist Primärschlüssel in „Kategorien“ und charakterisiert alle Kategorien eindeutig. Jede „KatNr“ gibt es in der Relation „Kategorien“ nur ein mal.

▪ „ANr“ ist Primärschlüssel in „Artikel“ und charakterisiert alle Artikel eindeutig. Jede „ANr“ gibt es in der Relation „Artikel“ nur ein mal.

▪ „LNr“ ist Primärschlüssel in „Lieferanten“ und charakterisiert alle Lieferanten eindeutig. Jede „LNr“ gibt es in der Relation „Lieferanten“ nur ein mal.

Übung Informatik I WS 03/04 – Dipl.-Inform. Tom Gelhausen SQL

Joins in SQL (Beispiel): Schema 12

▪ Schema:

▪ „Kategorie“ ist Fremdschlüssel in „Artikel“ und legt somit fest, dass in dieser Spalte nur Werte stehen dürfen, die auch in der Spalte „KNr“ der Relation „Kategorien“ vorkommen. In der Spalte „Kategorie“ der Relation „Artikel“ dürfen die Werte aber beliebig oft vorkommen.

▪ „LNr“ ist Fremdschlüssel in „Artikel“ und legt somit fest, dass in dieser Spalte nur Werte stehen dürfen, die auch in der Spalte „LNr“ der Relation „Lieferant“ vorkommen. In der Spalte „LNr“ der Relation „Artikel“ dürfen die Werte aber beliebig oft vorkommen.

Übung Informatik I WS 03/04 – Dipl.-Inform. Tom Gelhausen SQL

Gruppierung 22

- Welche Firma hat wie viel Frachtkosten bei uns verdient?
- Erster Ansatz:

Northwind: SQL: Query Results

Firma	Frachtkosten
Federal Shipping	\$32.38
Speedy Express	\$11.61
United Package	\$65.83
Speedy Express	\$41.34
United Package	\$51.30
United Package	\$58.17
United Package	\$22.98
Federal Shipping	\$148.33
United Package	\$13.97
Federal Shipping	\$81.91
Speedy Express	\$140.51
Federal Shipping	\$3.25
Speedy Express	\$5.509.00
United Package	\$395.00
Federal Shipping	\$48.29
Federal Shipping	\$14.606.00
Federal Shipping	\$3.67
Speedy Express	\$55.28
Federal Shipping	\$25.73
Speedy Express	\$208.58
Federal Shipping	\$66.29

Übung 1

Gruppierung 23

- Welche Firma hat wie viel Frachtkosten bei uns verdient?
- Zweiter Ansatz:

Northwind: SQL: Query Results

Firma	Frachtkosten
Federal Shipping	\$32.38
Federal Shipping	\$148.33
Federal Shipping	\$81.91
Federal Shipping	\$3.25
Federal Shipping	\$48.29
Federal Shipping	\$14.606.00
Federal Shipping	\$3.67
Federal Shipping	\$25.73
Federal Shipping	\$66.29
Federal Shipping	\$7.607.00
Federal Shipping	\$13.94
Federal Shipping	\$125.77
Federal Shipping	\$84.81
Federal Shipping	\$229.24
Federal Shipping	\$12.76
Federal Shipping	\$22.77
Federal Shipping	\$21.18
Federal Shipping	\$257.62
Federal Shipping	\$7.56
Federal Shipping	\$1.61
Federal Shipping	\$24.69
Federal Shipping	\$159.15
Federal Shipping	\$54.58

Übung 1

Gruppierung 24

- Lösung: „GROUP BY“ + Aggregationsfunktion
- „GROUP BY“ teilt die Tupel der Ergebnisrelation in Gruppen ein, so dass sich eine Äquivalenzrelation bzgl. der angegebenen Attributmenge ergibt
- „GROUP BY“ ist optional und steht zwischen „WHERE“ und „ORDER BY“
- Aggregationsfunktionen
 - count (*) – Zählen der Tupel der Ergebnisrelation
 - min (NameNichtGruppiertenAttributes) – gib jeweils das Minimum der Elemente in der angegebenen Spalte der Tupel jeder Gruppe zurück
 - max (NameNichtGruppiertenAttributes) – genau so
 - avg (NameNichtGruppiertenAttributes) – genau so
 - sum (NameNichtGruppiertenAttributes) – genau so
 - count ([distinct] NameNichtGruppiertenAttributes) – Zählen der Tupel pro Gruppe. Mit optionalem distinct werden zuerst die Duplikate bezüglich des angegebenen Attributes eliminiert.

Übung Informatik | WS 03/04 – Dipl.-Inform. Tom Gelhausen SQL

Gruppierung 25

Northwind: SQL: Query Results

Firma	sum
Federal Shipping	\$159.610,48
Speedy Express	\$147.780,09
United Package	\$352.484,70

3 row(s)
[Back](#) | [Create report](#)

phpPgAdmin - SQL - Mozilla Fir...

SQL Find

Database: Northwind

```
SELECT v."Firma", sum(b."Frachtkosten")
FROM "Bestellungen" b, "Versandfirmen" v
WHERE b."Versandueber"=v."FirmenNr"
GROUP BY v."Firma"
ORDER BY v."Firma";
```

Go Explain Reset

Übung Informatik I WS 03/04 – Dipl.-Inform. Tom Gelhausen SQL

Beispiel: Relationale Algebra 26

- SQL-Anfrage:


```
SELECT m.Nachname
FROM mitarbeiter m, projekt p
WHERE m.p_id = p.p_id
AND p.Budget < 40.000
```
- Ausdruck in relationaler Algebra:

$$\pi_{m.Nachname}(\sigma_{(m.p_id=p.p_id) \wedge (p.Budget < 40.000)}(m \times p))$$
- Anfragebaum:


```

      graph TD
      A((π  
m.Nachname)) --- B((σ  
(m.p_id=p.p_id) ∧ (p.Budget < 40.000)))
      B --- C((×))
      C --- D[mitarbeiter]
      C --- E[projekte]
      
```

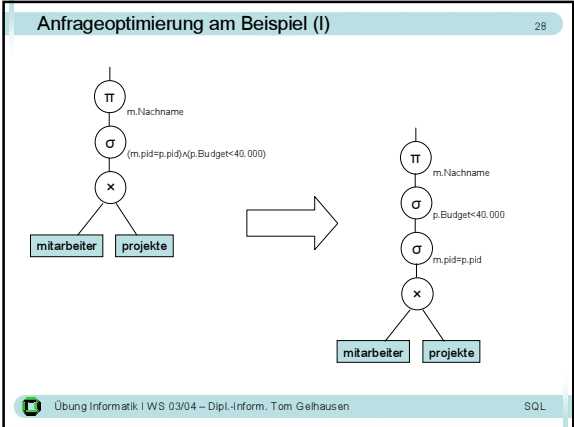
nach Felix Naumann: Crashkurs Informationssysteme, siehe <http://www.informatik.uni-bonn.de/stm/inf/ws03/04/PCBMC.ppt>

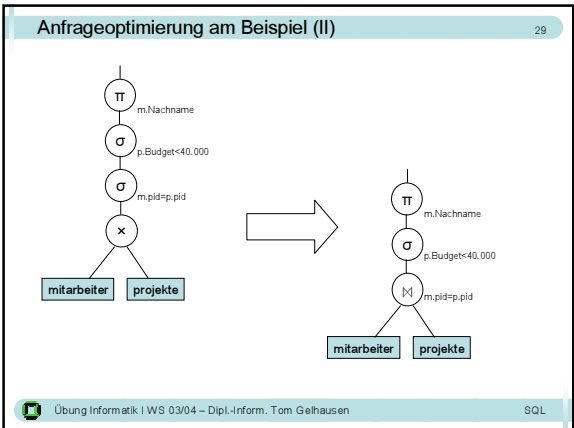
Übung Informatik I WS 03/04 – Dipl.-Inform. Tom Gelhausen SQL

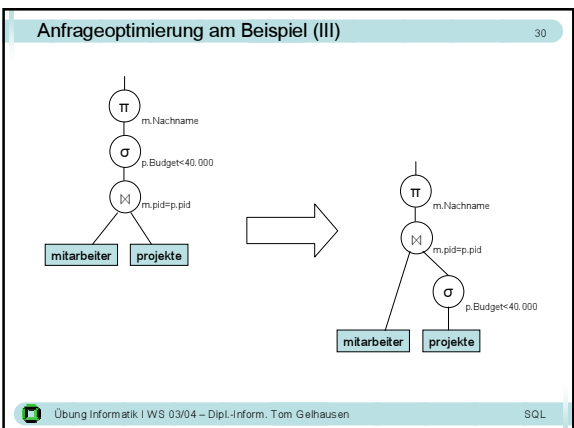
Anfrageoptimierung 27

- Transformation der internen Darstellung
 - zur effizienteren Berechnung
 - ohne Semantik zu verändern: suche nach äquivalenten Ausdrücken, die
 - kleine Zwischenergebnisse erfordern.
- Beispiele für anwendbare Transformationen
 - Kommutativität von σ und \times : $\sigma_{E}(\sigma_{F}(p)) = \sigma_{F \wedge E}(p) = \sigma_{E}(\sigma_{F}(p))$
 - Assoziativität von σ und \times : $(p_1 \times p_2) \times p_3 = p_1 \times (p_2 \times p_3)$
 - Projektionskaskaden: $\pi_{L_1}(\pi_{L_2}(p)) = \pi_{L_1}(p)$, wenn $L_1 \subseteq L_2$
 - Zusammenfassung von σ und \times zu Join-Operation (entspr. Def.)
 - π über σ ziehen: $\pi_{m.Nachname}(\sigma_{m.p_id=p.p_id}(m \times p)) = \pi_{m.Nachname}(\sigma_{m.p_id=p.p_id}(\pi_{m.Nachname, m.p_id, p.p_id}(m \times p)))$
 - π über \times ziehen: $\pi_{m.Nachname}(m \times \pi_{m.p_id=p.p_id}(p)) = \pi_{m.Nachname}(\pi_{m.Nachname, m.p_id}(m) \times \pi_{p.p_id}(p))$
 - σ über \times ziehen: $\sigma_{p.Budget < 40.000}(p \times m) = (\sigma_{p.Budget < 40.000}(p)) \times m$
- Beweise: Übung!

Übung Informatik I WS 03/04 – Dipl.-Inform. Tom Gelhausen SQL







Anfrageoptimierung am Beispiel (IV) 31

Übung Informatik I WS 03/04 – Dipl.-Inform. Tom Gelhausen SQL

Anfrageoptimierung am Beispiel (V) 32

Übung Informatik I WS 03/04 – Dipl.-Inform. Tom Gelhausen SQL

Zusammenfassung (bisher) 33

- SQL arbeitet auf der theoretischen Grundlage relationaler Algebra
- SQL arbeitet jedoch mit Vielfachmengen, genauer gesagt mit Listen:
 - ⇒ Duplikate können in (Quell- und Ergebnis-) Relationen enthalten sein (Hinweis: `DISTINCT`)
 - ⇒ Die Tupel einer Relation haben eine Reihenfolge, die man beeinflussen kann (`ORDER BY`)
- SQL ist eine deklarative Sprache, man sagt dem DBMS also was man haben will, nicht, wie das DBMS das Ergebnis bauen soll
 - ⇒ Daraus ergeben sich, wie eben gesehen, Optimierungsmöglichkeiten für das DBMS
- Zentrales Konzept der SQL ist der JOIN
 - ⇒ Lernen, verstehen, üben, üben, üben!

Übung Informatik I WS 03/04 – Dipl.-Inform. Tom Gelhausen SQL
