2.5 Relationale Algebra

2.5.1 Überblick

Codd-vollständige relationale Sprachen

- Relationale Algebra
  Abfragen werden durch exakte Angabe der auf den Relationen durchzuführenden Operationen formuliert
- Relationenkalküle
  Abfrageergebnis wird durch seine Eigenschaften beschrieben ohne Angabe einer Berechnungsprozedur

Codd hat gezeigt, dass Relationale Algebra, Tupelkalkül und Domänenkalkül gleich ausdrucksstark sind

"Leistungsmaßstab" ist Relationale Algebra:
mindestens so mächtige Sprache ist *Codd-vollständig*

2.5.1 Überblick RA (2)

Wie verhält sich SQL dazu?

Beispiel Algebra-Elemente:

```
SELECT nr, name FROM produkt
  WHERE hnr IN (SELECT nr FROM hersteller);
```

Beispiel Kalkül-Element:

```
SELECT nr, name FROM produkt
  WHERE EXISTS (SELECT ' ' FROM hersteller
    WHERE hersteller.nr=produkt.hnr);
```

- SQL basiert hauptsächlich auf Relationaler Algebra, enthält aber einzelne Elemente des Relationalen Kalküls.
- SQL ist nicht nur Codd-vollständig, sondern *mächtiger* als die Relationale Algebra
- SQL kann zwar optional Relationen (Mengen) erzeugen (*distinct*), arbeitet aber ansonsten mit Bags (ungeordneten Listen)

2.5.1 Überblick RA (3)

Bedeutung der Relationalen Algebra

- Formale Sprache für den Berechnungsweg von Abfrageergebnissen
- mathematische Rechenregeln ermöglichen Abfrageoptimierung durch algebraische Umformung
- auch geeignet zur Formulierung von Integrity Constraints

2.5.1 Überblick RA (4)

Was ist eine Algebra?

- besteht aus atomaren *Operanden* und *Operatoren*
- Regelwerk um daraus *Ausdrücke* zu bilden; dabei im allgemeinen Gruppierung durch Klammern nötig

Beipiel: Algebra der Arithmetik

- Operanden: Variablen (z.B. x) und Zahlenkonstanten (z.B. 42)
- Operatoren: arithemtische Operatoren (+, -, *, /)
- Beipiele für Ausdrücke in dieser Algebra:

```
(x + 2*y) * z
(x + 5)/(y - 2) + x
```

2.5.1 Überblick RA (5)

Operanden der Relationalen Algebra

- ∘ Relationen-Variablen
- ∘ Relationen-Constanten

Operatoren der Relationalen Algebra

- ∘ übliche Mengenoperationen (Vereinigung, Schnitt, Differenz)
- ∘ Extraktionsoperatoren "Selektion" und "Projektion"
- ∘ Kombinationsoperatoren "Cartesisches Produkt" und "Joins"
- ∘ Umbenennung ("Rename")

Ausdrücke der RA heißen *Queries* (Abfragen)
Relationenänderung durch Zuweisung, z.B.

$A := A \setminus \sigma_{nr\, =\, 'H2'}(A)$

5

2.5.1 Überblick RA (6)

Themen zur Relationalen Algebra:

- □ Operatoren und Rechenregeln
  Welche Basis-Operatoren werden benötigt für Abfragen?
  Unterschied *Relationen* versus *Bags*
- □ Dreiwertige Logik
  Behandlung von *NULL*-Werten
- □ Eigenschaften der Operatoren
  Berechnungskomplexität, begrenzte Ausdrucksstärke
- □ Algebraische Umformungen
  Äquivalenz algebraischer Ausdrücke, Abfrageoptimierung
- □ Formulierung von Constraints mit RA

6

2.5.2 RA Operatoren (1)

Mengen versus Bags

- □ Relationales Modell ursprünglich (Codd 1970) mit Relationen als *Mengen* definiert
  - ∘ keine Doubletten zulässig
  - ∘ Operatoren inperformant (warum?)
- □ Moderne Implementierungen bevorzugen *Bags* (ungeordnete Listen)
  - ∘ Tupel ungeordnet, aber Doubletten möglich
  - ∘ Operatoren performant implementierbar

Operatoren der Relationalen Algebra haben
eine Mengen- und eine Bag-Version

7

2.5.2 RA Operatoren (2)

a) Extraktionsoperatoren

- □ Projektion

  $\pi_{a_1,\ldots,a_n}(R) \quad = \quad$ Auswahl der Spalten $a_1, \ldots, a_n$

- □ Selektion

  $\sigma_{\text{Bedingung}}(R) \quad = \quad$ Auswahl Tupel, die *Bedingung* erfüllen

$\pi_{name,\ preis}$

$\sigma_{preis\, >\, 3}$

pnr#	*name*	*preis*	*hnr*
P1	Pritt	2.50	H1
P2	Füller	12.98	H2
P3	Tinte	3.20	H2

8

2.5.2 RA Operatoren (3)

Bemerkungen:

- Verhalten des Selektionsoperators σ für Bags und Mengen identisch (Warum?)
- Projektionsoperator π für Bags und Mengen verschieden

Wie verhält sich SQL-Befehl *select* dazu?

- Kombination von Selektion und Projektion:

```
SELECT name FROM produkt WHERE preis > 3;
```

entspricht

$\pi_{name}\ (\ \sigma_{preis > 3}\ (\ produkt\ )\ )$

- normalerweise Bagoperation, mit Modifier *distinct* aber Mengenoperation

2.5.2 RA Operatoren (4)

b) Mengenoperatoren

- Vereinigung: $R \cup S$
- Schnitt: $R \cap S$
- Differenz: $R \setminus S$

Bemerkungen:

- Voraussetzung: identische Attributnamen in R und S
- Schnittoperator eigentlich überflüssig wegen

$R \cap S\ =\ (R \cup S)\ \setminus\ (\,(R \setminus S)\ \cup\ (S \setminus R)\,)$

2.5.2 RA Operatoren (5)

Definition von $\cup$, $\cap$ und $\setminus$ für Mengen altbekannt.
Aber was ist bei Bags?

R enthalte Tupel t n-mal
S enthalte dasselbe Tupel t m-mal

- $R \cup S$ enthält Tupel t dann $(n+m)$-mal
- $R \cap S$ enthält Tupel t dann min (n, m)-mal
- $R \setminus S$ enthält Tupel t dann max $(0, n\text{-}m)$-mal

R

a	b
1	2
1	1

S

a	b
1	2
1	2

$R \cup S$

a	b
1	2
1	1
1	2
1	2

$R \cap S$

a	b
1	2

$R \setminus S$

a	b
1	1

$S \setminus R$

a	b
1	2

2.5.2 RA Operatoren (6)

Achtung:

Rechenregeln für Mengen- und Bag-Version können unterschiedlich sein:

- $R \cup S = S \cup R$ gilt für Mengen und Bags
- $(R \cup S) \setminus T = (R \setminus T) \cup (S \setminus T)$ gilt für Mengen, aber nicht Bags

Mengenoperationen in SQL

- *union, intersect* und *except* für $\cup$, $\cap$ und $\setminus$ nur möglich bei gleichen Rückgabetypen der verknüpften *select*-Statements
- normalerweise Mengenoperation (!), Bagoperation mit Modifier *all*, z.B. *select ... except all select ... ;*

2.5.2 RA Operatoren (7)

c) Umbenennung

Rename-Operator:

$$\varrho_{S(B_1,\ldots,B_n)}(R) \quad = \quad \text{Kopie von } R \text{ mit anderen Namen}$$

Wozu braucht man das?

- um zwei Relation kompatibel für Mengenoperationen $\cup$, $\cap$ oder $\setminus$ zu machen
- Vermeidung doppelter Attributnamen im Cartesischen Produkt

2.5.2 RA Operatoren (8)

d) Kombinationsoperatoren

Cartesisches Produkt (Kreuzprodukt):

$R \times S \quad = \quad$ Relation Q mit allen Attributen aus R und S, wobei jedes Tupel $\in R$ mit jedem Tupel $\in S$ kombiniert wird

R

a	b
1	2
3	4

S

c	d
x	y
x	z

$R \times S$

a	b	c	d
1	2	x	y
1	2	x	z
3	4	x	y
3	4	x	z

Tupelanzahl: $|R \times S| = |R| \cdot |S|$

2.5.2 RA Operatoren (9)

Join = Selektion auf Kartesischem Produkt

$$R \bowtie_{\text{Bed}} S \quad = \quad \sigma_{\text{Bed}}(R \times S)$$

Spezialfall "Natural Join":

Bei Bedingung der Form *a=b* entstehen Spalten, die in allen Werten identisch sind
=> Vereinfachungskonvention *Natural Join*

- ▹ Joinbedingung ist Gleichheit gleichnamiger Attribute
- ▹ identische Ergebnisspalten nur einmal aufgeführt
- ▹ Notation durch Weglassen der Bedingung

Beispiel: Natural Join von *R(a,b,c)* und *S(a,e,f)*:

$$R \bowtie S \quad = \quad \pi_{a,b,c,e,f}\left(R \bowtie_{R.a=S.a} S\right)$$

2.5.2 RA Operatoren (10)

Zusammenfassung

$\sigma_{Bedingung}$, $\pi_{Spaltenliste}$	Selektion, Projektion
$\cup$, $\cap$ und $\setminus$	Vereinigung, Schnitt und Differenz
$\varrho_{S(b1,\ldots,b_n)}$	Umbenennung
$\times$, $\bowtie_{Bedingung}$	Kreuzprodukt, Join

einige Rechenregeln

$\sigma_{Bed1}(\sigma_{Bed2}(R)) = \sigma_{Bed2}(\sigma_{Bed1}(R))$

$R \cup (S \cap T) = (R \cup S) \cap (R \cup T)$

$\pi_Y(\sigma_{Bed}(R)) = \sigma_{Bed}(\pi_Y(R))$

wenn in *Bed* nur Atrribute aus Y vorkommen

$\sigma_{Bed1}(R \bowtie_{Bed2} S) = \sigma_{Bed1}(R) \bowtie_{Bed2} \sigma_{Bed1}(S)$

wenn in *Bed1* nur Atrribute aus $attr(R) \cap attr(S)$ vorkommen

2.5.2 RA Operatoren (11)

Darstellung von Queries

"Was sind die Produkte des Herstellers 'H1',
die mehr als 3 Euro kosten?"

a) Verkettung der Operatoren

$$\pi_{pnr,\ name}\left(\sigma_{preis\,>\,3}(produkt)\ \cap\ \sigma_{hnr\,=\,'H1'}(produkt)\right)$$

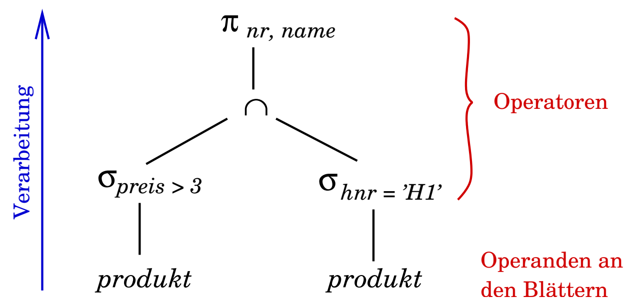
b) Definition temporärer Variablen

$$R\,(nr,\ name,\ hnr,\ preis) := \sigma_{preis\,>\,3}(produkt)$$
$$S\,(nr,\ name,\ hnr,\ preis) := \sigma_{hnr\,=\,'H1'}(produkt)$$
$$T := R \cap S$$
$$Ergebnis\ (nr,\ name) := \pi_{pnr,\ name}(T)$$

2.5.2 RA Operatoren (12)

"Was sind die Produkte des Herstellers 'H1',
die mehr als 3 Euro kosten?"

c) Baumdiagramm



2.5.3 Dreiwertige Logik (1)

Behandlung von NULL-Werten bei Vergleichs-
operatoren und logischen Verknüpfungen?

a) Vergleichsoperatoren

logischer Wert von (a ~ b) ist "unbekannt" (ω),
wenn a oder b den Wert NULL (unbekannt) hat

b) Logische Verknüpfungen

AND	F	ω	T
F	F	F	F
ω	F	ω	ω
T	F	ω	T

OR	F	ω	T
F	F	ω	T
ω	ω	ω	T
T	T	T	T

Not (F) = T
Not (T) = F
Not (ω) = ω

2.5.3 Dreiwertige Logik (2)

Beispiele:

`(a > NULL) AND (1 = 2)` ⟶ unwahr (F)
`(NULL = NULL) OR (1 = 2)` ⟶ unbekannt (ω)

Bei Duplikatslöschung (z.B. bei ∪ nötig) werden
NULL-Werte wie normale Werte behandelt

R

a	b
1	NULL
NULL	NULL

S

a	b
1	NULL
1	2

$S \cup R$

a	b
1	NULL
NULL	NULL
1	2

(ausführliche Diskussion in E.F. Codd: *Extending the Relational Model.* ACM Transactions on Database Systems (4) 1979, p 397-434)

2.5.4 Eigenschaften der RA

An der Tafel vorgeführt:

- Komplexität der Operatoren
  - Operationen *effizient* (vgl. THI, 4. Semester) durchführbar
  - Bagoperatoren geringere Komplexität als Mengenoperatoren
- Grenzen der Relationalen Algebra
  - es gibt Abfragen, die nicht in RA möglich sind (z.B. Berechnung der *transitiven Hülle*)
  - Grund: RA ist *nicht Turing-vollständig* (vgl. THI)

21

2.5.5 Algebraische Optimierung (1)

Beispiel:

Lieferant (A)
lnr#
lieferant

Produkt (B)
pnr#
produkt
preis

Lieferung (C)
nr#
lnr
pnr
menge

Die Abfrage

$$\pi_{\textit{lieferant, produkt}}\left(\sigma_{\textit{lieferant = 'Henkel'}}(A \bowtie B \bowtie C)\right) \qquad (1)$$

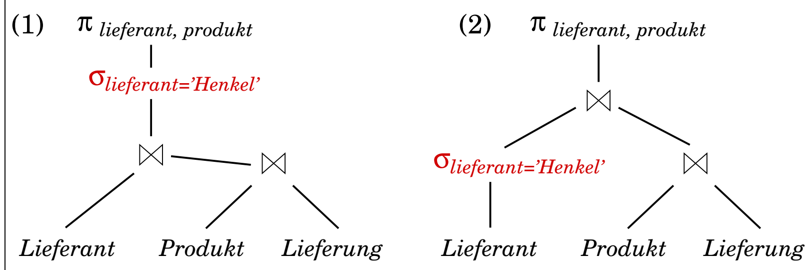
kann wegen $A \bowtie B = B \bowtie A$, $\sigma_C(R \bowtie S) = \sigma_C(R) \bowtie S$ und $(A \bowtie B) \bowtie C = A \bowtie (B \bowtie C)$ umgeformt werden in

$$\pi_{\textit{lieferant, produkt}}\left(C \bowtie (\sigma_{\textit{lieferant = 'Henkel'}}(A) \bowtie B)\right) \qquad (2)$$

22

2.5.5 Algebraische Optimierung (2)

Ausdrücke (1) und (2) sind *äquivalent*, d.h. liefern dasselbe Ergebnis



Selektion wird in (2) früher durchgeführt
=> kleinere Zwischenergebnisse => performanter

23

2.5.5 Algebraische Optimierung (3)

Allgemeine Optimierungsstrategie

- Selektionen zu Blättern verschieben
  - Selektionen verkleinern Zwischenergebnisse => möglichst früh durchführen
  - Joins sind besonders teuer. Nutze Regel
    $\sigma_{Bed}(R \sim S) = \sigma_{Bed}(R) \sim S$ für $\sim \in \{\times, \bowtie, \cap\}$
    wenn R alle Attribute aus *Bed* enthält
- zusätzliche Projektionen einfügen
  - nicht benötigte Spalten können entfernt werden => nicht weniger Tupel, aber weniger Attribute

(siehe Garcia-Molina, Ullman, Widom: *Database Systems - The Complete Book*. Prentice Hall 2002. Kapitel 16.2)

24

2.5.6 RA für Constraints (1)

Integritätsbedingungen durch (Un-) Gleichungen der Relationalen Algebra formulierbar:

A1 = ∅ A1 = A2 A1 ⊆ A2

wobei A1, A2 Ausdrücke der Relationalen Algebra

Beispiel a)

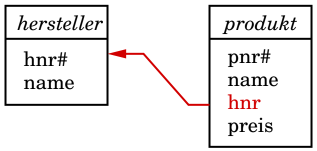
NOT NULL Constraint auf Attribut a in R:

$$\sigma_{a \text{ is } NULL}(R) = \emptyset$$

2.5.6 RA für Constraints (2)

Beispiel b)

Foreign Key Constraint



$$\pi_{hnr}(produkt) \subseteq \pi_{hnr}(hersteller)$$

oder $\pi_{hnr}(produkt) \setminus \pi_{hnr}(hersteller) = \emptyset$

Bemerkungen:

- ∘ Projektionen oben sind *Mengen*-Operationen (warum?)
- ∘ Ungleichung immer als Gleichung formulierbar wegen

$$R \subseteq S \iff R \setminus S = \emptyset$$