

Vorlesung aus dem WS21/22

Datenbanksysteme

Prof. Dr. C. B.

geT_EXt von Ningh

Contents

1	Einführung	2
2	Relation	3
2.1	Das Relationale Modell	3
2.2	Die Relationale Algebra	5
2.3	Realtionen-Kalkül	6
2.3.1	Tupelkalkül	6
2.3.2	Bereichskalkül	6
3	SQL	7
3.1	Tabelldefinition in SQL	7
3.2	Anfragen	8
3.3	Quantoren und Subqueries in SQL	11

1 Einführung

TODO: Later

2 Relation

2.1 Das Relationale Modell

DEFINITION 2.1 (Domain). Domain ist ein Wertbereich, der endlich oder unendlich sein kann.

DEFINITION 2.2 (Kartesisches Produkt). Kartesisches Produkt von k Menge ist Menge von allen möglichen Kombinationen der Elemente der Menge.

DEFINITION 2.3 (Relation). Relation R ist Teilmenge des kartesischen Produktes von k Domains D_1, \dots, D_k

$$R \subseteq D_1 \times D_2 \times \dots \times D_k$$

REMARK 2.4. • Die Relation kann endlich oder unendlich sein. Aber in Datenbanksysteme haben wir nur endliche Relation.

- Die Anzahl der Tupel einer Relation heisst Kardinalität $|\cdot|$. (Tupel ist Elemente der Relation.)
- Die einzelnen Domains lassen sich als Spalten einer Tabelle verstehen und werden als Attribute bezeichnet.
- Die Reihenfolge der Tupel spielt keine Rolle. Reihenfolge der Attribute ist von Bedeutung.

DEFINITION 2.5 (Relations-Schema(Alternative Definition in DBS)). Relation ist Ausprägung eines Relation-Schemas.

- Geordnetes Relationenschema

$$R = (A_1 : D_1, \dots, A_k : D_k)$$

- Domänen-Abbildung (ungeordnetes Rel.-Sch.)

$$R = \{A_1, \dots, A_k\} \text{ mit } \text{dom}(A_i) = D_i, i \leq k$$

REMARK 2.6. Vor: (geordnetes RS) Prägnanter aufzuschreiben.

Nach: (geordnetes RS) Einschränkungen bei logischer Datenunabhängigkeit: Einfügung neuer Attribute ist nur am Ende möglich.

Ungeordnetes RS: Reihenfolge der Spalte(Attribute) ist irrelevant.

REMARK 2.7. Begriff:

- Relation: Ausprägung eines Relationenschema.
- Datenbankschema: Menge von Relationenschema.
- Menge von Relation(Ausprägungen).

REMARK 2.8. Relation sind Menge von Tupel. Dann ist Reihenfolge der Tupel irrelevant. Und Duplikate kann nicht auftreten.

DEFINITION 2.9 (Schlüssel). Ein Schlüssel dient in einer relationalen Datenbank dazu, die Tupel einer Relation eindeutig zu identifizieren, sie zu nummern.

REMARK 2.10. • Ein/mehrere Attribute als Schlüssel kennzeichnen.

- Oft ist ein einzelnes Attribut nicht ausreichend, um die Tupel eindeutig zu identifizieren.
- Das muss eindeutig sein.

DEFINITION 2.11 (Schlüssel, formale). Eine Teilmenge S der Attribute eines Relationschemas R ($S \subseteq R$) heisst Schlüssel, wenn gilt:

Eindeutigkeit: Keine Ausprägung von R kann zwei verschiedene Tupel enthalten, die sich in allen Attributen von S gleichen. (\forall Ausprägung $r \forall t_1, t_2 \in r : t_1 \neq t_2 \Rightarrow t_1[S] \neq t_2[S]$.)

Minimalität: Es existiert keine echte Teilmenge $T \subsetneq S$, die die Bedingung der Eindeutigkeit erfüllt. (\forall Attributmenge T , die Eindeutigkeit erfüllen, gilt: $T \subseteq S \Rightarrow T = S$.)

REMARK 2.12. • Eine Menge $S \subseteq R$ heisst Superschlüssel, wenn sie die Eindeutigkeitseigenschaft erfüllt.

- In der Mathematik wird allgemein eine Menge M als minimale Menge bezüglich einer Eigenschaft B bezeichnet, wenn es keine echte Teilmenge von M gibt, die ebenfalls B erfüllt.
- Damit können wir auch definieren: Ein Schlüssel ist ein minimaler Superschlüssel.
- Minimalität bedeutet nicht: Schlüssel mit den wenigsten Attributen, sondern keine überflüssigen Attribute sind enthalten.
- Manchmal gibt es mehrere verschiedene Schlüssel.

Man wählt einen dieser Kandidaten aus als sogenannter Primärschlüssel.

DEFINITION 2.13 (Fremdschlüssel). Attribute, die auf einen Schlüssel einer anderen Relation verweist, heisst Fremdschlüssel.

REMARK 2.14. Die Eindeutigkeit bezieht sich nicht auf die aktuelle Ausprägung einer Relation. Sondern immer auf die Semantik der realen Welt.

2.2 Die Relationale Algebra

Wichtigste Beispiele:

- Relationale Algebra
- Relationale-Kalkül.

5 Grundoperationen der Relationalen Algebra:

- Vereinigung: $R = S \cup T$
- Differenz: $R = S - T$
- Kartesisches Produkt: $R = S \times T$
- Selektion: $R = \sigma_F(S)$
- Projektion $\pi_{A,B,\dots}(S)$

REMARK 2.15. Die Kombinationen aus Selektion und kartesischen Produkt heiss Join.

Eine Reihe nützlicher Operationen lassen sich mit Hilfe der 5 Grundoperationen ausdrücken:

- Durchschnitt: $R = S \cap T$
- Quotient $R = S \div T$
- Join $R = S \bowtie T$
 - Thera-Join $R \bowtie_{A\Theta B} S, \Theta \in \{=, \leq, <, \geq, >, \neq\}$ (Denn: $R \bowtie_{A\Theta B} S = \sigma_{A\Theta B}(R \times S)$)
 - Equi-Join $R \bowtie_{A=B}$
 - Natural Join

2.3 Relationen-Kalkül

- Relationale Algebra ist prozedurale Sprache:
Ausdruck gibt an, unter Benutzung welcher Operation das Ergebnis berechnet werden soll
- Relationen-Kalkül ist deklarative Sprache:
Ausdruck beschreibt, welche Eigenschaft die Tupel der Ergebnisrelation haben müssen ohne eine Berechnungsprozedur dafür anzugeben.
- Es gibt zwei verschiedene Ansätze:
 - Tupelkalkül: Variablen sind vom Typ Tupel.
 - Bereichskalkül: Variablen haben einfachen Typ.

REMARK 2.16. Hier gibt es eine relativ lange Teil über Mathematik. Ich habe gar kein Interesse daran. Etwas über Formel, Syntax, Semantik, Interpretation (Belegung von Variablen)

2.3.1 Tupelkalkül

Man arbeitet mit

- Tupelvariablen: t
- Formeln: $\varphi(t)$
- Ausdrücken: $\{t \mid \varphi(t)\}$

DEFINITION 2.17 (Definites Schema von Tupelvariablen). • $\text{Schema}(T) = (A_1 : D_1, A_2 : D_2, \dots)$

- $\text{Schema}(T) = R$.

REMARK 2.18. $t[A]$ oder $t.A$ für einen Attributnamen $A \in \text{Schema}(t)$.

2.3.2 Bereichskalkül

DEFINITION 2.19. Ein Ausdruck hat die Form:

$$\{x_1, x_2, \dots, \mid \varphi(x_1, x_2, \dots)\}$$

Atome haben die Form:

- $R_1(x_1, x_2, \dots)$: Tupel (x_1, x_2, \dots) tritt in Relation R auf.
- $x \Theta y$: x, y Bereichsvariablen bzw. Konstant. $\Theta \in \{=, <, \leq, >, \geq, \neq\}$

3 SQL

3.1 Tableddefinition in SQL

```
CREATE TABLE name          // name Name der Relation
(
    a1 d1 c1,                // definition des ersten Attributs
    a2 d2 c2,
    ....
    ak dk ck                 // definition des Attributs Nr. k
)
```

Hierbei bedeuten

- ai der name des Attributs Nr. i
- di der Typ (die Domain) des Attributs
- ci ein optionaler Constraint für das Attribut

Wirkung: Definition eines Realtionschemas mit einer leeren Realtion als Ausprägung.
Der SQL-Standard kennt u.a. folgende Datentypen:

- integer oder auch integer4, integer
- smallint oder integer2
- float(p) oder auch float
- decimal(p,q) und numeric(p,q) mit p Stellen, davon q Nachkommast.
- character(n), char(n) für Strings fester Länge n
- character varying(n), varchar(n): variable Strings
- date, time, timestamp für Datum und Zeit

Einfache Zusätze (Integritätsbedingungen) können unmittelbar hinter einer Attributdefinition stehen:

- not null
- primary key
- unique
- references t1(a1)
- default w1: Wert w1 ist Default, wenn unbesetzt.
- check f

Zusätze, die keinem einzelnen Attribut zugeordnet sind, stehen mit Komma abgetrennt in extra Zeilen

- primary key(A1,A2,...)

- unique (A1,A2,...)
- foreign key (A1,A2,...) references t1 (B1,B2,...)
- check f

REMARK 3.1. SQL ist case-insensitiv.

Lösche eines Tupels in B Referenzen nicht möglich. Es gibt aber verschiedene Zusätze:
foreign key(a5,a6) references B(b1,b2)

- on delete cascade
- on update cascade
- on delete set null

FIXME:

Weitere DDL(Data Definition Language):

```
drop table n1 // RS n1 wird mit allen evtl.
               / /vorhandenen Tupeln gelöscht
alter table n1 add (a1 d1 c1, a2 d2 c2, ...)
alter table drop (a1, a2, ...)
alter table modify (a1 d1 c1, a2 d2 c2, ...)
```

3.2 Anfragen

Grundform einer Anfrage:

```
Projektion -> SELECT <List von Attributnamen bzw. *>
Kreuzprodukt -> FROM <ein oder mehrere Relationennamen>
Selektion -> [WHERE <Bedingung>]
```

Mengenoperationen:

```
SELECT ... FROM ... WHERE ...
UNION
SELECT ... FROM ... WHERE ...
```

REMARK 3.2. • SQL ist relational vollständig.

- Duplikatelimination nur, wenn durch das Schlüsselwort DISTINCT explizit verlangt:

```
SELECT * FROM ... -- Keine Projektion
SELECT A1, A2, ... FROM ... -- Projektion ohne
                             -- Duplikatelimination
SELECT DISTINCT A1, A2, ... -- Projektion mit
                             -- Duplikatelimination
```

- Man kann Schreibarbeit sparen, indem man den Relationen lokal kurze Namen zuweist(Alias-Namen):


```
SELECT m.Name, a.Name
FROM Mitarbeiter m, Abteilung a
WHERE ...
```

- Where

- Vergleichsoperatoren: =, <, ≤, >, ≥, <>
- Test auf Wert undefiniert: A IS NULL / IS NOT NULL
- Inexakter Stringvergleich: A LIKE 'Datenbank%'
 - * % steht für einen beliebig belegbaren Teilstring
 - * _ steht für genau ein einzelnes frei belegbares Zeichen
- A1 IN (2, 3, 4, 5, 12, 13)

Join: Normalerweise wird der Join wie bei der relationalen Algebra als Selektionsbedingung über dem kartesischen Produkt formuliert. Folgende Anfrage sind möglich in SQL:

```
SELECT * FROM Mitarbeiter m, Abteilung a WHERE m.ANr = a.ANr
SELECT * FROM Mitarbeiter m JOIN Abteilung a on a.ANr = m.ANr
SELECT * FROM Mitarbeiter JOIN Abteilung using (ANr)
SELECT * FROM Mitarbeiter natural JOIN Abteilung
```

TODO: OUTER JOIN. bu ru du yi ba, qi mo kao bu hui kao outer join.

Änderungs-Operationen:

Grundsätzlich unterscheiden wir:

```
INSERT: Einfuegen von Tupeln in eine Relation
DELETE: Loeschen von Tupeln aus einer Relation
UPDATE: Aedern von Tupeln einer Relation
```

REMARK 3.3. Usage:

- UPDATE:

```
UPDATE Relation
SET attribut1 = ausdruck1
  [, ...,
  attributn = ausdruckn]
[WHERE bedingung]
```

- UPDATE-Operationen können zur Verletzung von Integritätsbedingungen führen: Abbruch der Operation mit Fehlermeldung.

- DELETE:

```
DELETE FROM relation
[WHERE bedingung]
```

- Löscht alle Tupel, die die Bedingung erfüllen

- Ist keine Bedingung angegeben, werden alle Tupel gelöscht
- Abbruch der Operation, falls eine Integritätsbedingung verletzt würde (z.B. Fremdschlüssel ohne cascade)

- INSERT:

```
INSERT INTO relation (attribut1, attribut2,...)
VALUES (konstante1, konstante2, ...)
or
INSERT INTO relation
VALUES (konstante1, konstante2, ...)
or
INSERT INTO relation [(attribut1 , ...)]
( SELECT ... FROM ... WHERE ... )
```

- Ist die optionale Attributliste hinter dem Relationennamen angegeben, dann
 - * können unvollständige Tupel eingefügt werden: Nicht aufgeführte Attribute werden mit NULL belegt.
 - * werden die Werte durch die Reihenfolge in der Attributliste zugeordnet.
- Ist die optionale Attributliste hinter dem Relationennamen angegeben, dann
 - * können unvollständige Tupel eingefügt werden: Nicht aufgeführte Attribute werden mit NULL belegt
 - * werden die Werte durch die Reihenfolge in der Attributliste zugeordnet. (mangelnde Datenunabhängigkeit!)
- Wirkung zum Einfügen berechneter Tupel:
 - * Alle Tupel des Ergebnisses der SELECT-Anweisung werden in die Relation eingefügt.
 - * Die optionale Attributliste hat dieselbe Bedeutung wie bei der entsprechenden Ein-Tupel-Operation.
 - * Bei Verletzung von Integritätsbedingungen (z.B. Fremdschlüssel nicht vorhanden) wird die Operation nicht ausgeführt (Fehlermeldung).

3.3 Quantoren und Subqueries in SQL

EXAMPLE 3.4 (Subquery).

```
SELECT * FROM Kunde
WHERE EXISTS (SELECT ... FROM ... WHERE)
```

REMARK 3.5. • In where-Klausel der Subquery auch Zugriff auf Relationen/Attribute der Hauptquery.

- Eindeutigkeit ggf. durch Aliasnamen für Relation.

```
SELECT * FROM kunde konstante1
WHERE EXISTS ( SELECT *
                FROM Kunde k2
                WHERE k1.Adr = K2.Adr AND ...
            )
```

- Existenz-Quantor ist realisiert mit dem Schlüsselwort EXISTS.
- Term TRUE gdw. Ergebnis der Subquery nicht leer.

Es gibt keine direkte Unterstützung in SQL von Allquantor. Aber es ist äquivalent $\forall x : \varphi(x) \Leftrightarrow \neg \exists : \neg \varphi(x)$. Also Notation in SQL:

```
... WHERE NOT EXISTS (SELECT ... FROM ... WHERE NOT ...)
```

Direkt Subquery:

An jeder Stelle in der select- und where-Klausel, an der ein konstanter Wert stehen kann, kann auch eine Subquery (select...from...where...) stehen.

EXAMPLE 3.6.

```
SELECT Preis,
Preis * (SELECT Kurs FROM Devisen
          WHERE DName = 'US$' ) as USPreis
FROM Waren Where ...
```

REMARK 3.7. Es gibt weitere Quantoren bei Standard-Vergleichung in WHERE.

- $A_i = \text{ALL} (\text{SELECT } \dots \text{ FROM } \dots \text{ WHERE } \dots)$ // Allquantor
- $A_i = \text{ANY} (\text{SELECT } \dots \text{ FROM } \dots \text{ WHERE } \dots)$ // Ex.quantor
- $A_i = \text{SOME} (\text{SELECT } \dots \text{ FROM } \dots \text{ WHERE } \dots)$ // Ex.quantor

Es bedeutet $A_i \Theta \text{all} (\text{Subquery}) \equiv \{ \dots \mid \forall t \in \text{Subquery} : A_i \Theta t \}$, $\Theta \in \{=, <, \leq, >, \geq, <>\}$

REMARK 3.8. Es kann Subquery mit IN sein.

```
A_i [NOT] in ...
```

EXAMPLE 3.9. Typische Form der Subquery:

```
SELECT ... FROM ... WHERE EXISTS (SELECT * FROM ...)
SELECT ... FROM ... WHERE A <= ALL (SELECT B FROM ...)
... WHERE A <= (SELECT B FROM ... WHERE Schlüssel = ...)
```