Vorlesung aus dem WS21/22

Datenbanksysteme

Prof. Dr. C. B.

geTEXt 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
2.4 Sortieren, Gruppieren und Views in SQL . . . 7

3 SQL **8**
3.1 Tabelledefinition in SQL . . . 8
3.2 Anfragen . . . 9
3.3 Quantoren udn Subqueries in SQL . . . 12
3.4 Sortieren, Gruppieren und Views in SQL . . . 13
3.4.1 Sortieren . . . 13
3.4.2 Aggregation . . . 13
3.4.3 Gruppierung . . . 13
3.4.4 View . . . 13
3.4.5 Rechtvergabe . . . 14

1 Einführung

TODO: Later

2 Relation

2.1 Das Relationale Modell

Definition 2.1 (Domain). Domain ist ein Wertbereich, der endliche 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 $D_1, \cdots, D_k$

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

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 is 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 Realtion-Schemas.

- Geordnetes Relationenschema

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

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

$$R = \{A_1, \cdots, A_k\} \text{ mit } \mathrm{dom}(A_i) = D_i, i \leqslant i \leqslant k$$

Remark 2.6.
Vor: (geordnetes RS) Prägnanter aufzuschreiben.
Nach: (geordnetes RS) Einschränkungen bei logischer Datenunabhänigigkeit: Einfügung neuer Attribute ist nur am Ende möglich.
Ungeordnetes RS: Reihenfolge der Spalte(Attribute) ist irrelevant.

Remark 2.7. Begriff:

- Relation: Ausprägung eines Relationenschema.
- Datenbankscheme: 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 kennzeichen.
- Oft ist ein einzelnes Attribut nicht ausreichend, um die Tupel eindeutig zu identifizieren.
- Das muss eindeutig sein.

Definition 2.11 (Schlüssel, formale). Eine Telimeng S der Attribute eines Relationschemas R ($S \subset R$) heisst Schulüssel, wenn gilt:

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

Minimalität: Ex existiert keine echte Teilmenge $T \subsetneq S$, die breit die Bedigung der Eindeutigkeit erfüllt. ($\forall$ Attributmenge T, die Eindeutigkeit erfüllen, gilt: $T \subset S \Rightarrow T = S$.)

Remark 2.12.
- Eine Menge $S \subseteq R$ heisst Superschlüssel, wenn sie die Eindeutigkeitseingenschaft 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 definiren: Ein Schlüsseln 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 wählte einen dieser Kandidaten aus als sogenannter Primärschlüssel.

Definition 2.13 (Fremdschlüssel). Attribute das auf einen Schlüssel einer anderen Relation verweist, heisst Fremdschlüssel.

Remark 2.14. Die Eindeutigkeit bezieht sicht nicht auf die aktuelle Ausprägung eine Relation. Sondern immer auf die Semantik der rearlen 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 $\pi_{A,B,\ldots}(S)$

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 $R \bowtie_{A \Theta B} S, \Theta \in \{=, \leqslant, <, \geqslant, >, \neq\}$ (Denn: $R \bowtie_{A \Theta B} S = \sigma_{A \Theta B}(R \times S)$)
  - Equi-Join $R \bowtie_{A=B}$
  - Natural Join

2.3 Realtionen-Kalkül

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

Remark 2.16. Hier gibt es eine relativ lange Teil über Mahtematik. Ich habe gar kein Interesse daran. Etwas über Formel, Sytax, Sematik, Interpretation(Belegung von Variablen)

2.3.1 Tupelkalkül

Man arbeitet mit

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

Definition 2.17 (Definitertes Schema von Tupelvaribalen).

- Schema(T) = $(A_1 : D_1, A_2 : D_2, \cdots)$
- Schema(T) = R.

Remark 2.18. $t[A]$ oder $t.A$ für einen Attributnamen $A \in$Schema(t).

2.3.2 Bereichskalkül

Definition 2.19. Ein Ausdruck hat die Form:

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

Atome haben die Form:

- $R_1(x_1, x_2, \cdots)$: Tupel $(x_1, x_2, \cdots)$ tritt in Relation R auf.
- $x\Theta$: x, y Bereichsvariblen bzw. Konstant.$\Theta \in \{=, <, \leqslant, >, \geqslant, \neq\}$

2.4 Sortieren, Gruppieren und Views in SQL

Alles hier ist nur über SQL.

Und folgend ist E/R-Modell und Normalform. Ich bin der Meinung, dass es ganz sinnlos hier die zu wiederholen ist. 28.12.2021

3 SQL

3.1 Tabelledefinition 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 geloescht
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 Attributsnamene bzw. *>
Kreuzprodkt -> FROM   <ein oder mehrere Relationennamen>
Selektion   -> [WHERE <Bedigung>]
```

Mengenoperationen:

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

Remark 3.2.
- SQL ist relational vollständig.
- Duplikatelimination nur, wenn durch das Schlüsselwort DISTINCT explizit verlang:

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

- Mann 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: $=, <, \leqslant, >, \geqslant, <>$
  - Test auf Wert undefinirt: 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 Realtion
SET attribut1 = ausdruck1
    [,...,
    attributn = ausdruckn]
[WHERE bedingung]
```

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

- DELTEE:

```
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 Attributsliste 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 Attributsliste 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 udn 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 aud Relationen/Attribute der Hauptquery.
- Eindeutigkeit ggf. durch Aliasnamen für Relation.

```
SELECT * FROM kunde konstante1
WHERE EXISTS ( SLEECT *
                 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 direkt Unterstützung in SQL von Allquantor. Aber es ist äquivalent $\forall x : \varphi(x) \Leftrightarrow \neg\exists : \neg\neg\varphi(x)$. 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.

```
- Ai = ALL (SELECT ... FROM ... WHERE ...)  // Allquantor
- Ai = ANY (SELECT ... FROM ... WHERE ...)  // Ex.quantor
- Ai = SOME (SELECT ... FROM ... WHERE ...) // Ex.quantor
```

Es bedeutute :$A_i\Theta$ all (Subquery) $\equiv \{... \mid \forall t \in$ Subquery:$A_i\Theta t\}, \Theta \in \{=, <, \leqslant, >, \geqslant, <>\}$

Remark 3.8. Es kann Subquery mit IN sein.

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

Example 3.9. Typische Fomr der Subquery:

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

3.4 Sortieren, Gruppieren und Views in SQL

3.4.1 Sortieren

In SQL mit ORDER BY A1, A2, ...
Nach Attribut kann man ASC für aufsteigend(Default) order DESC für absteigend angeben.

3.4.2 Aggregation

Aggregation berechnet Eigenschaft ganzer Tupel-Menge. Es gibt folgendes Aggregatfunktionen in SQL:

- count
- sum
- avg
- max
- min

REMARK 3.10.
- NULL-Werte werden ignoriert(auch bei count).
- Eine Duplikatelimination kann erzwungen werden
  - count(distinct KNname) zählt verschiedene Kunden
  - count(all KName) zählt alle Eintäge (ausser NULL)
  - count(KName) ist identisch mit count(all KName)
  - count(*) zählt die Tupel des Anfrageergebnisses

3.4.3 Gruppierung

Syntax in SQL:

```
SELECT            ...
FROM              ...
[WHERE            ...]
[group by A1,A2,...
 [HAVING          ...]]
[ORDER BY         ...]
```

3.4.4 View

View ist eine Virtuelle Relation, die für den Benutzer aus wie eine Relation sieht. Aber die Relation ist nicht real existent/gespeichert. Definition und Löschen in SQL:

```
CREATE [OR REPLACE] view VName [(A1,A2,...)] AS SELECT ...
drop view VName
```

REMARK 3.11.
- Automatisch sind in dieser View alle Tupel der Basisrelation, die die Selecktionsbedingung erfüllen.

- In Wirklichkeit wird lediglich die View-definition in die Anfrage eingesetzt und dann ausgewertet.
- Bei Updates in der Basisrelation ändert sich auch die virtuelle Relation. Umgekehrt können (mit Einschränkungen) auch Änderungen an der View durchgeführt werden, die sich auf die Basisrelation auswirken.
- Views sind ein wichtiges Strukturierungsmittel für Anfragen und die gesamte Datenbank.
- Selektion, Projektion, Kreuzprodkt, Join, Vereinigung, Differenz, Schnitt, Gruppieren, Aggregation und Subqueries sind erlaubt in VIEW.
- Sortieren ist nicht erlaubt.

FIXME: Zhe yi duan shi sha ya .

3.4.5 Rechtvergabe

Syntax:

```
GRANT Rechteliste
ON Relation          //bzw. View
TO Benutzerliste
[with grant option]
```

Remark 3.12. Möglichkeit:

- Rechteliste:
  - ALL [privileges]
  - SELECT, INSERT, DELETE (mit Kommas sep.)
  - UPDATE (optional in Klammern: Attributnamen)
- Benutzerliste:
  - Benutzernamen (mit Passwort identifizert)
  - TO PUBLIC
- Rücknahme von Rechten:

```
REVOKE Rechteliste
ON Relation
FROM Benutzerliste
[RESTRICT] Abbruch, falls Recht bereits weitergegeben
[CASCADE] ggf. Propagierung der Revoke-Anweisung
```