Skript zur Vorlesung

Informationssysteme

Wintersemester 2018/19

Kapitel 2: Das Relationale Modell

Charakteristika

- Einführungskapitel:
  Viele Informationen darstellbar als Tabelle
- Die Tabelle (Relation) ist das ausschließliche Strukturierungsmittel des relationalen Datenmodells
- Edgar F. Codd, 1970.
  A relational model of data for large shared data banks. Comm. of the ACM 13.06.1970
- Grundlage vieler kommerzieller und freier DBS:

ORACLE
IBM DB2 / Informix
Microsoft SQL Server
MySQL
PostgreSQL

Domain

- Ein Wertebereich (oder Typ)
- Logisch zusammengehörige Menge von Werten
- Beispiele:
  - D_1 = Integer
  - D_2 = String
  - D_3 = Date
  - $D_4 = \{rot, gelb, grün, blau\}$
  - $D_5 = \{1, 2, 3\}$
- Kann *endliche* oder *unendliche* Kardinalität |…| haben:
  - $|D_4| = 4$; $|D_5| = 3$;
  - $|D_1|$ = *unendlich*; ebenso $|D_2|$ und $|D_3|$.

Kartesisches Produkt

- Bedeutung: kartesisches Produkt (Kreuzprodukt) von k Mengen?

  Menge von allen möglichen Kombinationen der Elemente der Mengen

- Beispiel ($k = 2$):
  $D_1 = \{1, 2, 3\}$, $D_2 = \{a,b\}$
  $D_1 \times D_2 =$

  $\{(1,a), (1,b), (2,a), (2,b), (3,a), (3,b)\}$

- Beispiel ($k = 3$):
  $D_1 = D_2 = D_3 = \mathcal{N}$
  $D_1 \times D_2 \times D_3 = \{(1,1,1),(1,1,2),(1,1,3),\ldots,(1,2,1),\ldots\}$

Relation in der Mathematik

- Mathematische Definition:
  Relation R ist Teilmenge des kartesischen Produktes von k Domains $D_1, D_2, \ldots, D_k$

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

- Beispiel ($k = 2$):
  $D_1 = \{1, 2, 3\}$, $D_2 = \{a,b\}$

$R_1 = \{\}$ (leere Menge)
$R_2 = \{(1,a), (2,b)\}$
$R_3 = \{(1,a), (2,a), (3,a)\}$
$R_4 = D_1 \times D_2 = \{(1,a),(1,b),(2,a),(2,b),(3,a),(3,b)\}$

Relation in der Mathematik

- Weiteres Beispiel:
  $D_1 = D_2 = \mathcal{N}$
  Relation $R_1 =$ {(1,1),(1,2),(1,3),...,(2,2),(2,3),....,
  (3,3),(3,4),...(4,4),(4,5),(4,6),....}

  Wie heißt diese mathematische Relation?

$$\leq \quad R_1 = \{(x, y) \in \mathcal{N} \times \mathcal{N} \mid x \leq y\}$$

- Es gibt endliche und unendliche Relationen
  (wenn mindestens eine Domain unendlich ist).
- In Datenbanksystemen: Nur endliche Relationen
  Unendlich: Nicht darstellbar .
- Die Anzahl der Tupel einer Relation heißt *Kardinalität* |…|

Relation in der Mathematik

- Die einzelnen Domains lassen sich als Spalten einer Tabelle verstehen und werden als Attribute bezeichnet
- Für $R \subseteq D_1 \times \ldots \times D_k$ ist k der Grad (Stelligkeit)
- Die Elemente der Relation heißen Tupel: (1,a), (2,a), (3,a) sind drei Tupel vom Grad $k = 2$
- Relation ist Menge von Tupeln d.h. die Reihenfolge der Tupel spielt keine Rolle: $\{(0,a), (1,b)\} = \{(1,b), (0,a)\}$
- Reihenfolge der Attribute ist von Bedeutung: $\{(a,0), (b,1)\} \neq \{(0,a), (1,b)\}$

Christian-Albrechts-Universität zu Kiel

Relationen-Schema

Alternative Definition in DBS:
Relation ist Ausprägung eines Relationen-Schemas.

- Geordnetes Relationenschema:
  - k-Tupel aus Domains (Attribute)
  - Attribute werden anhand ihrer Position im Tupel referenziert
  - Attribute können zusätzlich einen Attributnamen haben

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

- Domänen-Abbildung (ungeordnetes Rel.-Sch.):
  - Relationenschema R ist Menge von Attributnamen:
  - Jedem Attributnamen A_i ist Domäne D_i zugeordnet:
  - Attribute werden anhand ihres Namens referenziert

  $R = \{A_1, \ldots.\ A_k\}$ mit $\mathrm{dom}(A_i) = D_i,\ 1 \leq i \leq k$

Relationen-Schema

- Beispiel: Städte-Relation

Städte

München	1.211.617	Bayern
Bremen	535.058	Bremen
Passau	49.800	Bayern

- Als geordnetes Relationenschema:

Schema: R = (Name: String, Einwohner: Integer, Land: String)
Ausprägung: r = {(München, 1.211.617, Bayern), (Bremen, 535.058, Bremen), (Passau, 49.800, Bayern)}

- Als Relationenschema mit Domänenabbildung:

Schema: R = {Name, Einwohner, Land}
mit dom(Name) = String, dom(Einwohner) = Integer, ...
Ausprägung: $r = \{t_1, t_2, t_3\}$
mit t_1(Name) = München, t_1(Einwohner) = 1.211.617,...

Diskussion

- Vorteil von geordnetem Relationenschema:
  - Prägnanter aufzuschreiben.
    Wichtig z.B. beim Einfügen neuer Tupel:
    t_3 = (Passau, 49.800, Bayern)
    vergleiche: t_3 (Name) = Passau; t_3 (Einwohner) = ...
- Nachteil von geordnetem Relationenschema:
  - Einschränkungen bei logischer Datenunabhängigkeit:
    Applikationen sensibel bzgl. Einfügung neuer Attribute (nur am Ende!)
- Definitionen prinzipiell gleichwertig
- Wir verwenden beide Ansätze

Begriffe

- Relation: Ausprägung eines Relationenschemas
- Datenbankschema: Menge von Relationenschemata
- Datenbank: Menge von Relationen (Ausprägungen)

Relationenname

Attribut(-Namen)

Städte

Name	**Einwohner**	**Land**
München	1.211.617	Bayern
Bremen	535.058	Bremen
Passau	49.800	Bayern

Relationenschema

Ausprägung der Relation

Tupel

Attributwerte

Duplikate

- Relationen sind Mengen von Tupeln.
  Konsequenzen:
  - Reihenfolge der Tupel irrelevant (wie bei math. Def)
  - Es gibt keine Duplikate (gleiche Tupel) in Relationen:
    {(0,a), (0,a), (0,a), (1,b)} = {(0,a), (1,b)}

- Frage: Gilt dies auch für die Spalten beim ungeordneten Relationenschema $R = \{A_1,\ldots,A_k\}$?
  - Reihenfolge der Spalten ist irrelevant
    (das ist gerade das besondere am ungeordneten RS)
  - Duplikate treten nicht auf, weil alle Attribut-Namen verschieden sein müssen

Schlüssel

- Tupel müssen eindeutig identifiziert werden
- Warum? Z.B. für Verweise:

Mitarbeiter

PNr	Name	Vorname	Abteilung
001	Huber	Erwin	
002	Mayer	Hugo	
003	Müller	Anton	

Abteilungen

ANr	Abteilungsname
01	Buchhaltung
02	Produktion
03	Marketing

- Objektidentifikation in Java:
  Mit Referenz (Adresse im Speicher)
- Im relationalen Modell werden Tupel anhand von Attributwerten identifiziert
- Ein/mehrere Attribute als Schlüssel kennzeichnen
- Konvention: Schlüsselattribut(e) unterstreichen!

Schlüssel

Beispiel: **PNr** und **ANr** werden Primärschlüssel:

Mitarbeiter

PNr	Name	Vorname	Abteilung
001	Huber	Erwin	
002	Mayer	Hugo	
003	Müller	Anton	

Abteilungen

ANr	Abteilungsname
01	Buchhaltung
02	Produktion
03	Marketing

- Damit müssen diese Attributswerte eindeutig sein!
- Verweis durch Wert dieses Schlüsselattributs:

Mitarbeiter

PNr	Name	Vorname	Abteilung
001	Huber	Erwin	01
002	Mayer	Hugo	01
003	Müller	Anton	02

Abteilungen

ANr	Abteilungsname
01	Buchhaltung
02	Produktion
03	Marketing

Zusammengesetzter Schlüssel

- Oft ist ein einzelnes Attribut nicht ausreichend, um die Tupel eindeutig zu identifizieren
- Beispiel:

Lehrveranstaltung

VNr		Semester
012	Einführung in die Informatik	WS 2001/02
012	Einführung in die Informatik	WS 2002/03
013	Medizinische Informationssyst.	WS 2001/02
...	...	...

- Schlüssel: **(VNr, Semester)**
- Anmerkung: Warum ist dies ein schlechtes DB-Design?

Nicht redundanzfrei:
Der Titel ist mehrfach in der Datenbank gespeichert.
→ hierzu mehr in Kapitel 6+7

Schlüssel: Formale Definition

Definition:

- Eine Teilmenge S der Attribute eines Relationenschemas R ($S \subseteq R$) heißt **Schlüssel**, wenn gilt:
  1) **Eindeutigkeit**
     Keine Ausprägung von R kann zwei verschiedene Tupel enthalten, die sich in allen Attributen von S gleichen.
  2) **Minimalität**
     Es existiert keine echte Teilmenge $T \subsetneqq S$, die bereits die Bedingung der Eindeutigkeit erfüllt.

Anm.: Der Teilmengenbegriff umfasst die Menge selbst, also jede Menge ist Teilmenge von sich selbst. Eine Teilmenge einer Menge S, die ungleich S ist, heißt *echte* Teilmenge. In Symbolen: $T \subsetneqq S \Leftrightarrow T \subseteq S \wedge T \neq S$

Schlüssel: Formale Definition

Manche Lehrbücher definieren in noch formalerer Notation:

1) Eindeutigkeit:
   $\forall$ möglichen Ausprägungen r und Tupel t_1, $t_2 \in r$ gilt:
   $t_1 \neq t_2 \Rightarrow t_1[S] \neq t_2[S]$.
2) Minimalität:
   $\forall$ Attributmengen T, die (1) erfüllen, gilt:
   $T \subseteq S \Rightarrow T = S$.

Hierbei bezeichne $t[S]$ ein Tupel t eingeschränkt auf die Attribute aus S (alle anderen Attribute gestrichen).

Wir schreiben später auch $\pi_S(t)$ für $t[S]$ (*Projektion*, s. Kap. 3)

Superschlüssel / Minimale Menge

- Eine Menge $S \subseteq R$ heißt Superschlüssel (oder Oberschlüssel, engl. Superkey), wenn sie die Eindeutigkeitseigenschaft erfüllt
- Der Begriff des Superschlüssels impliziert keine Aussage über die Minimalität
- 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
  (minimale Menge $S \subseteq R$ mit Eindeutigkeits-Eigenschaft)

Schlüssel: Beispiele

- Gegeben sei die folgende Relation:

Lehrveranst	LNr	VNr	Titel	Semester
(t_1=)	1	012	Einführung in die Informatik	WS 2001/02
(t_2=)	2	012	Einführung in die Informatik	WS 2002/03
(t_3=)	3	013	Medizinische Informationssyst.	WS 2001/02
	...	...	...	...

- {VNr} ist kein Schlüssel
  Nicht eindeutig: $t_1 \neq t_2$ **aber** $t_1[\text{VNr}] = t_2[\text{VNr}] = 012$
- {Titel} ist kein Schlüssel
  (gleiche Begründung)
- {Semester} ist kein Schlüssel
  Nicht eindeutig: $t_1 \neq t_3$ **aber** $t_1[\text{Semester}] = t_3[\text{Semester}]$

Schlüssel: Beispiele

Lehrveranst	LNr	VNr	Titel	Semester
(t_1=)	1	012	Einführung in die Informatik	WS 2001/02
(t_2=)	2	012	Einführung in die Informatik	WS 2002/03
(t_3=)	3	013	Medizinische Informationssyst.	WS 2001/02
	...	...	...	...

- {LNr} ist Schlüssel !!!
  Eindeutigkeit: Alle t_i[LNr] sind paarweise verschieden,
  d.h. $t_1[\text{LNr}] \neq t_2[\text{LNr}]$, $t_1[\text{LNr}] \neq t_3[\text{LNr}]$, $t_2[\text{LNr}] \neq t_3[\text{LNr}]$
  Minimalität: Trivial, weil 1 Attribut kürzeste Möglichkeit

- {LNr, VNr} ist kein Schlüssel (aber Superschlüssel)
  Eindeutigkeit: Alle t_i[LNr,VNr] paarweise verschieden.
  Nicht minimal, da echte Teilmenge $\{\text{LNr}\} \subset \{\text{LNr, VNr}\}$ ($\neq$) die Eindeutigkeit bereits gewährleistet, s.o.

Schlüssel: Beispiele

Lehrveranst	LNr	VNr	Titel	Semester
(t_1=)	1	012	Einführung in die Informatik	WS 2001/02
(t_2=)	2	012	Einführung in die Informatik	WS 2002/03
(t_3=)	3	013	Medizinische Informationssyst.	WS 2001/02
	...	...	...	...

- {VNr, Semester} ist Schlüssel !!!

  Eindeutigkeit: Alle t_i[VNr, Semester] paarw. verschieden:
  - t_1 [VNr, Semester] = (012, WS 2001/02)
  - t_2 [VNr, Semester] = (012, WS 2002/03)
  - t_3 [VNr, Semester] = (013, WS 2001/02)

  ≠ ≠ ≠

  Minimalität:
  Weder {VNr} noch {Semester} gewährleisten Eindeutigkeit (siehe vorher). Dies sind alle echten Teilmengen.

Primärschlüssel

- Minimalität bedeutet **nicht**:
  Schlüssel mit den wenigsten Attributen
- Sondern Minimalität bedeutet:
  Keine überflüssigen Attribute sind enthalten
  (d.h. solche, die zur Eindeutigkeit nichts beitragen)
- Manchmal gibt es mehrere verschiedene Schlüssel
  - {LNr}
  - {VNr, Semester}

  → **Schlüsselkandidat** (SQL: **unique**)
- Später ist wichtig, *alle* Schlüsselkandidaten zu ermitteln.
- Man wählt einen dieser Kandidaten aus als sogenannter **Primärschlüssel** (SQL: **primary key**)
- Attribut(e) das auf einen Schlüssel einer anderen Relation verweist, heißt **Fremdschlüssel** (SQL: **foreign key**)

Schlüssel: Semantische Eigenschaft

- Die Eindeutigkeit bezieht sich **nicht** auf die aktuelle Ausprägung einer Relation *r* !!!!
- Sondern immer auf die **Semantik** der realen Welt

Mitarbeiter

PNr	Name	Gehalt
001	Müller	1700 €
002	Mayer	2172 €
003	Huber	3189 €
004	Schulz	2171 €

  – Bei der aktuellen Relation wären sowohl {PNr} als auch {Name} und {Gehalt} eindeutig.
  – Aber es ist möglich, dass mehrere Mitarbeiter mit gleichem Namen und/oder Gehalt eingestellt werden
  – {PNr} ist **für jede mögliche** Ausprägung eindeutig

Tabellendefinition in SQL

- Definition eines Relationenschemas:

CREATE TABLE n ← n Name der Relation

(

$a_1\ d_1\ c_1$, ← Definition des ersten Attributs

$a_2\ d_2\ c_2$,

...

$a_k\ d_k\ c_k$ ← Definition des Attributs Nr. k

)

- hierbei bedeuten...
  - a_i der Name des Attributs Nr. i
  - d_i der Typ (die Domain) des Attributs
  - c_i ein optionaler Constraint (Integritätsbedignung) für das Attribut.
- Wirkung: Definition eines Relationenschemas mit einer leeren Relation als Ausprägung.

Basis-Typen in SQL

Der SQL-Standard kennt u.a. folgende Datentypen:

- **integer** oder auch **integer4**, **int**
- **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

Zusätze bei Attributdefinitionen

- Einfache Zusätze (Integritätsbedingungen) können unmittelbar hinter einer Attributdefinition stehen:
  - **not null**: Das Attribut darf nicht undefiniert sein
    in DBS: undefinierte Werte heissen **null**-Werte
  - **primary key**: Das Attribut ist Primärschlüssel
    (nicht bei zusammengesetzten Schlüsseln)
  - **unique**:
    Das Attribut ist Schlüsselkandidat
  - **references** t_1 (a_1):
    Ein Verweis auf Attribut a_1 von Tabelle t_1
  - **default** w_1: Wert w_1 ist Default, wenn unbesetzt.
  - **check** f:
    Die Formel f wird bei jeder Einfügung überprüft, z.B.:
    check $A <= 100$

Integritätsbedingungen

- Zusätze, die keinem einzelnen Attribut zugeordnet sind, stehen mit Komma abgetrennt in extra Zeilen
  - **primary key** (A_1, A_2, ...):
    Zusammengesetzter Primärschlüssel
  - **unique** (A_1, A_2, ...):
    Zusammengesetzter Schlüsselkandidat
  - **foreign key** (A_1, A_2, ...) **references** t_1 (B_1, B_2, ...)
    Verweis auf zusammengesetzten Schlüssel in Rel. T_1
    Anmerkung: Fehlt die Angabe (B_1, B_2, ...) hinter t_1 so wird automatisch (A_1, A_2, ...) eingesetzt.
  - **check** f
- Anmerkung: SQL ist case-insensitiv:
  Im Ggs. zu Java hat die Groß-/Kleinschreibung weder bei Schlüsselworten noch bei Bezeichnern Bedeutung

Schlüssel-Definitionen

A

a_1	a_2	a_3	a_4	a_5	a_6	…
x	y	z	u	1	2	
y	z	x	v	1	2	
x	x	y	y	2	3	
~~z~~	~~z~~	~~v~~	~~x~~	~~4~~	~~5~~	

B

b_1	b_2	…
1	2	
2	3	
6	7	

(nicht möglich, wenn (a_5,a_6) Fremdschlüssel)

Anmerkung zu Fremdschlüssel (Verweise):

– Tupel in A ohne gültigen Partner in B nicht erlaubt

– Ohne weiteren Zusatz nicht möglich, Tupel in B, auf die durch Tupel in A verwiesen wird, zu löschen oder die Werte von b_1,b_2 zu verändern.

Schlüssel-Definitionen

A	a_1	a_2	a_3	a_4	a_5	a_6	…
	x	y	z	u	1	2	
	y	z	x	v	1	2	
	x	x	y	y	2	3	

B	b_1	b_2	…	
	1	2		(kann **nicht** gelöscht/verändert werden)
	2	3		(kann **nicht** gelöscht/verändert werden)
	6	7		(kann gelöscht und verändert werden)

Löschen eines Tupels in B mit Referenzen nicht möglich.

Es gibt aber verschiedene Zusätze:

- **foreign key** (a_5,a_6) **references** B (b_1,b_2)
  on delete cascade
  Löschen eines Tupels in B führt auch zum Löschen der entsprechenden Tupel in A
- **on update cascade**
  Verändern eines Tupels in B führt zum Verändern in A
- **on delete set null**
  „hängende Verweise“ werden ggf. auf **null** gesetzt.

Beispiel Tabellendefinition

- Zusammengesetzter Primärschlüssel {VNr, Semester}:

create table Lehrveranst
(
LNr **integer** **not null**,
VNr **integer** **not null**,
Titel **varchar**(50),
Semester **varchar**(20) **not null**,
primary key (VNr, Semester)
)

- Alternative mit einfachem Primärschlüssel {LNr}:

create table Lehrveranst2
(
LNr **integer** **primary key**,
VNr **integer** **not null**,
Titel **varchar**(50),
Semester **varchar**(20) **not null**
)

Beispiel Tabellendefinition

- Tabelle für Dozenten:

create table Dozenten
(
DNr **integer** **primary key**,
Name **varchar**(50),
Geburt **date**,
)

- Verwendung von Fremdschlüsseln:

create table Haelt
(
Dozent **integer** **references** Dozenten (DNr)
on delete cascade,
VNr **integer** **not null,**
Semester **varchar**(20) **not null,**
primary key (Dozent, VNr, Semester),
foreign key (VNr, Semester) **references** Lehrveranst
)

Beispiel Tabellendefinition

- Das Schlüsselwort **on delete cascade** in *Haelt* führt dazu, dass bei Löschen eines *Dozenten* auch entsprechende Tupel in *Haelt* gelöscht werden
- Weitere Konstrukte der Data Definition Language:
  - **drop table** n_1
    Relationen-Schema n_1 wird mit allen evtl. vorhandenen Tupeln gelöscht.
  - **alter table** n_1 **add** $(a_1\ d_1\ c_1,\ a_2\ d_2\ c_2, \ldots)$
    - Zusätzliche Attribute oder Integritätsbedingungen werden (rechts) an die Tabelle angehängt
    - Bei allen vorhandenen Tupeln Null-Werte
  - **alter table** n_1 **drop** $(a_1, a_2, \ldots)$
  - **alter table** n_1 **modify** $(a_1\ d_1\ c_1,\ a_2\ d_2\ c_2, \ldots)$