

Skript zur Vorlesung:

Einführung in die Informatik: Systeme und Anwendungen

Sommersemester 2008

Kapitel 3: Datenbanksysteme

Vorlesung: Prof. Dr. Christian Böhm

Übungen: Annahita Oswald, Bianca Wackersreuther

Skript © 2004 Christian Böhm, Peer Kröger

<http://www.dbs.ifi.lmu.de/Lehre/InfoNF>



Überblick

3.1 Einleitung

3.2 Das Relationale Modell

3.3 Die Relationale Algebra

3.4 Mehr zu SQL

3.5 Das E/R-Modell

3.6 Normalformen

Charakteristika

- Die Tabelle (Relation) ist das ausschließliche Strukturierungsmittel des relationalen Datenmodells
- Edgar F. Codd, 1970.
- Grundlage vieler kommerzieller DBS:

ORACLE®

 Informix

Microsoft
SQL Server

Domain

- Ein Wertebereich (oder Typ)
- Logisch zusammengehörige Menge von Werten
- Beispiele:
 - $D_1 = \text{Integer}$ (ganze Zahlen $\{\dots, -2, -1, 0, 1, 2, \dots\}$)
 - $D_2 = \text{String}$ (Zeichenketten, z.B. „Hallo“)
 - $D_3 = \text{Date}$ (Datum, typischerweise: TTMMYYYY)
 - $D_4 = \{\text{rot, gelb, grün, blau}\}$
 - $D_5 = \{1, 2, 3\}$
- Kann **endliche** oder **unendliche** Kardinalität haben

Kartesisches Produkt

- Bedeutung des kartesisches Produkts (Kreuzprodukts) von k Domains:

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 ($k = 3$):

$$D_1 = D_2 = D_3 = \mathcal{N}$$

$$D_1 \times D_2 \times D_3 = \{(1, 1, 1), (1, 1, 2), (1, 1, 3), \dots, (1, 2, 1), \dots\}$$

Relation

- Mathematische Definition:

Relation R ist Teilmenge des kartesischen Produktes von k Domains D_1, D_2, \dots, D_k

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

- Beispiel ($k = 2$):

$$D_1 = \{1, 2, 3\}, D_2 = \{a, b\}$$

$$R_1 = \{\} \text{ (leere Menge)}$$

$$R_2 = \{(1, a), (2, b)\}$$

$$R_3 = \{(1, a), (2, a), (3, a)\}$$

$$R_4 = D_1 \times D_2 = \{(1, a), (1, b), (2, a), (2, b), (3, a), (3, b)\}$$

Relation

- Weiteres Beispiel:

$$D_1 = D_2 = \mathcal{N}$$

$$\text{Relation } R_1 = \{(1,1), (1,2), (1,3), \dots, (2,2), (2,3), \dots, \\ (3,3), (3,4), \dots, (4,4), (4,5), (4,6), \dots\}$$

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

Relation

- Die einzelnen Domains lassen sich als **Spalten einer Tabelle** verstehen und werden als **Attribute** bezeichnet
- Für $R \subseteq D_1 \times \dots \times D_k$ 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:
 $\{(a,0), (b,1)\} \neq \{(0,a), (1,b)\}$

Relation

- Relationen sind die mathematische Formalisierung der ***Tabelle***
- Jedes Tupel steht für einen Datensatz
- Die einzelnen Informationen sind in den Attributwerten der Tupel gespeichert
- Die Attribute können auch benannt sein:
D₁ = Name: String
D₂ = Vorname: String
- Zeichenketten und Zahlen sind die häufigsten Domains

Relationen-Schema

Alternative Definition:

Relation ist Ausprägung eines **Relationen-Schemas**

- Geordnetes Relationenschema:
 - k -Tupel aus Domains (Attribute)
 - Attribute können benannt sein
 - Attribute werden anhand ihrer **Position** im Tupel ref.

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

- 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

$$R = \{A_1, \dots, A_k\} \text{ mit } \text{dom}(A_i) = D_i, 1 \leq i \leq k$$

Relationen-Schema

- Beispiel: Städte-Relation

| Städte | Name | Einwohner | Land |
|--------|---------|-----------|--------|
| | München | 1.211.617 | Bayern |
| | Bremen | 535.058 | Bremen |
| | Passau | 49.800 | Bayern |

- Als geordnetes Relationenschema:

Schema: Städte = (Name: String, Einwohner: Integer, Land: String)

Ausprägung: {(München, 1.211.617, Bayern), (Bremen, 535.058, Bremen), (Passau, 49.800, Bayern)}

- Als Relationenschema mit Domänenabbildung:

Schema: Städte = {Name, Einwohner, Land}

mit $\text{dom}(\text{Name}) = \text{String}$, $\text{dom}(\text{Einwohner}) = \text{Integer}$, ...

Ausprägung: $\{t_1, t_2, t_3\}$

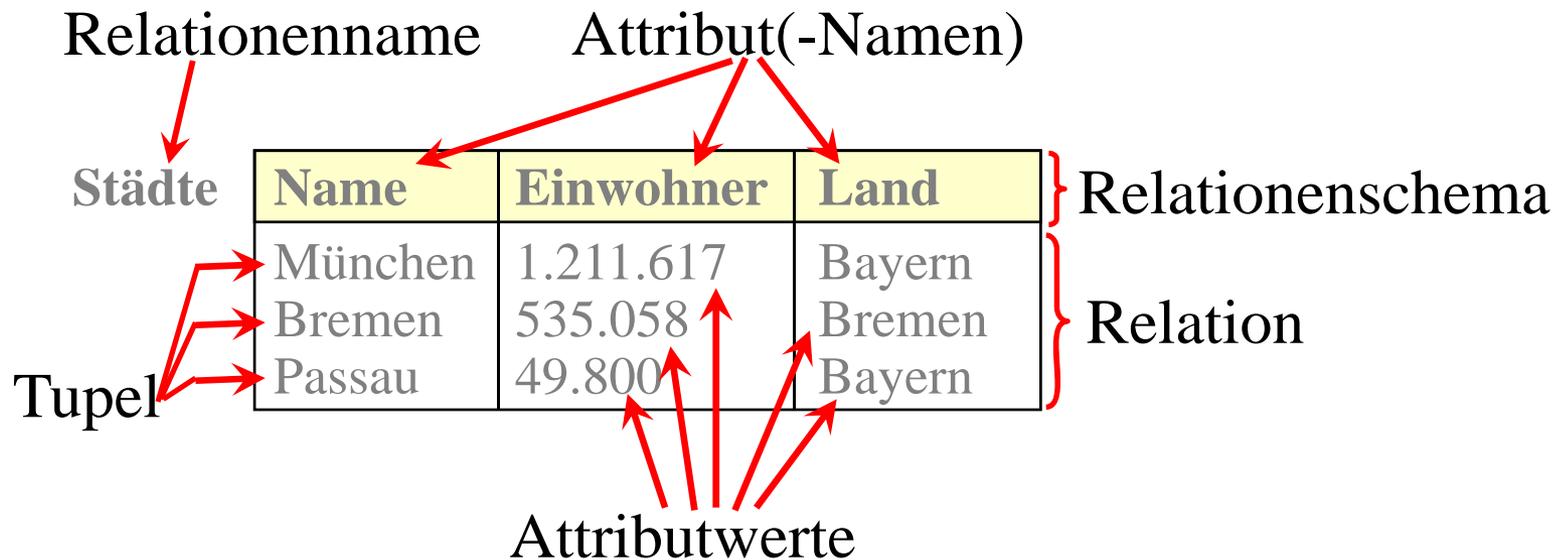
mit $t_1(\text{Name}) = \text{München}$, $t_1(\text{Einwohner}) = 1.211.617, \dots$

Diskussion

- Vorteil von geordnetem Relationenschema:
 - Prägnanter aufzuschreiben.
Wichtig z.B. beim Einfügen neuer Tupel:
 $t_3 = (\text{Passau}, 49.800, \text{Bayern})$
vergleiche: $t_3(\text{Name}) = \text{Passau}$; $t_3(\text{Einwohner}) = \dots$
- Nachteil von geordnetem Relationenschema:
 - Einschränkungen bei logischer Datenunabhängigkeit:
Applikationen sensibel bzgl. Einfügung neuer Attribute (nur am Ende da sonst Position der vorhandenen Attribute verändert wird)
- Definitionen gleichwertig
(lassen sich ineinander überführen)
- Wir verwenden beide Ansätze

Begriffe

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



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 $R = \{A_1, \dots, A_k\}$?
 - Reihenfolge der Spalten ist irrelevant
(das ist gerade das besondere am ungeordneten RS)
 - Duplikate treten nicht auf, weil alle Attribut-Namen verschieden

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

| <u>PNr</u> | Name | Vorname | Abteilung |
|------------|--------|---------|-----------|
| 001 | Huber | Erwin | |
| 002 | Mayer | Hugo | |
| 003 | Müller | Anton | |

Abteilungen

| <u>ANr</u> | Abteilungsname |
|------------|----------------|
| 01 | Buchhaltung |
| 02 | Produktion |
| 03 | Marketing |

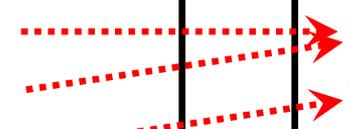
- Damit müssen diese Attributswerte eindeutig sein!
- Verweis durch Wert dieses Schlüsselattributs:

Mitarbeiter

| <u>PNr</u> | Name | Vorname | Abteilung |
|------------|--------|---------|-----------|
| 001 | Huber | Erwin | 01 |
| 002 | Mayer | Hugo | 01 |
| 003 | Müller | Anton | 02 |

Abteilungen

| <u>ANr</u> | Abteilungsname |
|------------|----------------|
| 01 | Buchhaltung |
| 02 | Produktion |
| 03 | Marketing |



Zusammengesetzter Schlüssel

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

Lehrveranstaltung

| <u>VNr</u> | Titel | <u>Semester</u> |
|------------|----------------------------------|-----------------|
| 012 | Einführung in die Informatik | WS 2001/02 |
| 012 | Einführung in die Informatik | WS 2002/03 |
| 013 | Medizinische Informationssysteme | WS 2001/02 |
| ... | ... | ... |

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

Schlüssel: Formale Definition

Definition:

- Eine Teilmenge S der Attribute eines Relationenschemas R heißt **Schlüssel**, wenn gilt:
 - **Eindeutigkeit**
Keine Ausprägung von R kann zwei verschiedene Tupel enthalten, die sich in **allen** Attributen von S gleichen.
 - **Minimalität**
Keine echte Teilmenge von S erfüllt bereits die Bedingung der Eindeutigkeit

Schlüssel: Formale Definition

In noch formalerer Notation:

Definition:

- Sei r eine Relation über dem Relationenschema R und $S \subseteq R$ eine Auswahl von Attributen.
- $t[S]$ bezeichne ein Tupel t eingeschränkt auf die Attribute aus S (alle anderen Attribute gestrichen)
- (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:
Für alle Attributmengen S und T , die (1) erfüllen, gilt:
 $T \subseteq S \Rightarrow T = S$

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

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 Informationssysteme | WS 2001/02 |
| ... | ... | ... | ... | ... |

- $\{LNr\}$ ist Schlüssel !!!

Eindeutigkeit: Alle $t_i[LNr]$ sind paarweise verschieden,
d.h. $t_1[LNr] \neq t_2[LNr]$, $t_1[LNr] \neq t_3[LNr]$, $t_2[LNr] \neq t_3[LNr]$

Minimalität: Trivial, weil 1 Attribut kürzeste Möglichkeit
- $\{LNr, VNr\}$ ist kein Schlüssel

Eindeutigkeit: Alle $t_i[LNr, VNr]$ paarweise verschieden.

Nicht minimal, da **echte** Teilmenge $\{LNr\} \subset \{LNr, VNr\}$ (\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 Informationssysteme | 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}
- Diese nennt man auch **Schlüsselkandidaten**
- Man wählt einen dieser Kandidaten aus als sog.
Primärschlüssel (oft den kürzesten, nicht immer)

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
```

```
(
```

```
  a1 d1 c1,
```

```
  a2 d2 c2,
```

```
  ...
```

```
  ak dk ck
```

```
)
```

← *n* Name der Relation

← Definition des ersten Attributs

← Definition des Attributs Nr. *k*

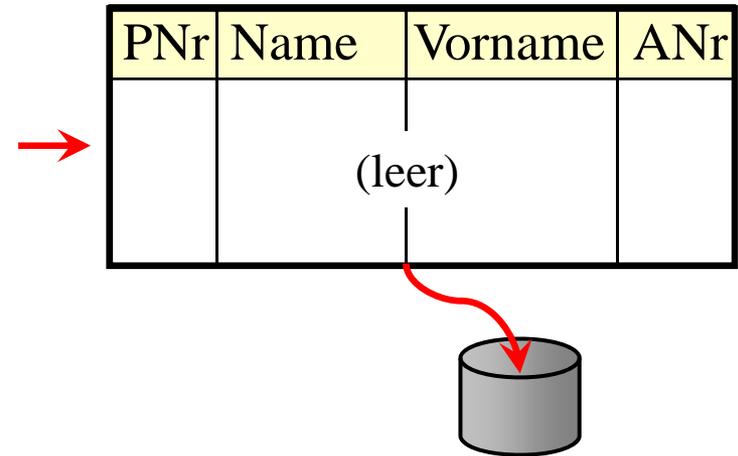
- hierbei bedeuten...

- *a*_{*i*} der Name des Attributs Nr. *i*
- *d*_{*i*} der Typ (die Domain) des Attributs
- *c*_{*i*} ein optionaler Constraint für das Attribut

Datenbank-Sprachen

- Was passiert dabei?
 - Es wird eine leere Tabelle angelegt
 - Beispiel:

```
CREATE TABLE Mitarbeiter
(
  PNr  NUMBER (3),
  Name  CHAR (20),
  Vorname CHAR (20),
  Anr  NUMBER (2)
)
```



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 (nur bei 1-Attribut-Schlüsseln)
 - **check f** :
Die Formel f wird bei jeder Einfügung überprüft
 - **references $t_1 (a_1)$** :
Ein Verweis auf Attribut a_1 von Tabelle t_1
Verschiedene Zusatzoptionen:
 - **on delete cascade**
 - **on delete set null**
 - **default w_1** : Wert w_1 ist Default, wenn unbesetzt.

Integritätsbedingungen

- Zusätze, die keinem einzelnen Attribut zugeordnet sind, stehen mit Komma abgetrennt in extra Zeilen
 - **primary key** (s_1, s_2, \dots):
Zusammengesetzter Primärschlüssel
 - **foreign key** (s_1, s_2, \dots) **references** t_1 (...)
Verweis auf zusammengesetzten Schlüssel in t_1
 - **check** f
- Anmerkung:
SQL ist case-insensitiv:
 - im Ggs. zu Java hat die Groß-/Kleinschreibung weder bei Schlüsselworten noch bei Bezeichnern Bedeutung

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** ($a_1 d_1 c_1, a_2 d_2 c_2, \dots$)
 - 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, \dots)
 - **alter table** n_1 **modify** ($a_1 d_1 c_1, a_2 d_2 c_2, \dots$)