Teil II

Relationale Datenbanken – Daten als Tabellen

Relationale Datenbanken – Daten als Tabellen

1. Relationen für tabellarische Daten
2. SQL-Datendefinition
3. Grundoperationen: Die Relationenalgebra
4. SQL als Anfragesprache
5. Änderungsoperationen in SQL
6. Anwendungsbeispiel

Lernziele für heute ...

- Grundverständnis zur Struktur relationaler Datenbanken
- Kenntnis der Basisoperationen relationaler Anfragesprachen
- elementare Fähigkeiten in der Anwendung von SQL

Relationen für tabellarische Daten

Relationenmodell

Konzeptuell: Datenbank = **Menge von Tabellen (= Relationen)**

WEINE

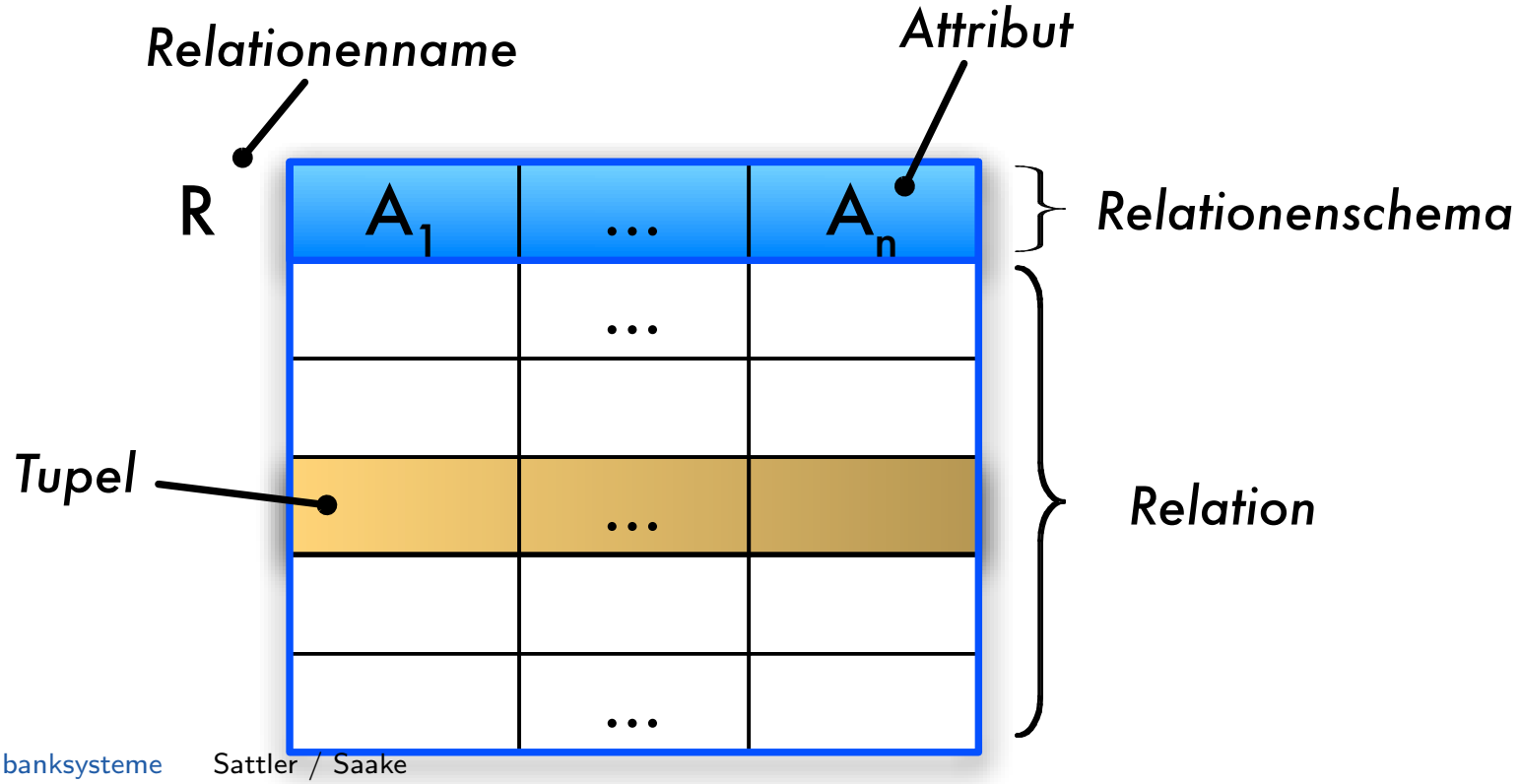
WeinID	Name	Farbe	Jahrgang	Weingut
1042	La Rose Grand Cru	Rot	1998	Château La Rose
2168	Creek Shiraz	Rot	2003	Creek
3456	Zinfandel	Rot	2004	Helena
2171	Pinot Noir	Rot	2001	Creek
3478	Pinot Noir	Rot	1999	Helena
4711	Riesling Reserve	Weiß	1999	Müller
4961	Chardonnay	Weiß	2002	Bighorn

ERZEUGER

Weingut	Anbaugebiet	Region
Creek	Barossa Valley	Südaustralien
Helena	Napa Valley	Kalifornien
Château La Rose	Saint-Emilion	Bordeaux
Château La Pointe	Pomerol	Bordeaux
Müller	Rheingau	Hessen
Bighorn	Napa Valley	Kalifornien

Darstellung von Relationen und Begriffe

- „Tabellenkopf“: **Relationenschema**
- Eine Zeile der Tabelle: **Tupel**; Menge aller Einträge: **Relation**
- Eine Spaltenüberschrift: **Attribut**
- Ein Eintrag: **Attributwert**



Integritätsbedingungen: Schlüssel

- Attribute einer Spalte **identifizieren** eindeutig gespeicherte Tupel: **Schlüsseleigenschaft**
- etwa **Weingut** für Tabelle **ERZEUGER**

ERZEUGER

Weingut	Anbaugebiet	Region
Creek	Barossa Valley	Südaustralien
Helena	Napa Valley	Kalifornien
Château La Rose	Saint-Emilion	Bordeaux
Château La Pointe	Pomerol	Bordeaux
Müller	Rheingau	Hessen
Bighorn	Napa Valley	Kalifornien

- auch Attributkombinationen können Schlüssel sein!
- Schlüssel können durch Unterstreichen gekennzeichnet werden

Integritätsbedingungen: Fremdschlüssel

- Schlüssel einer Tabelle können in einer anderen (oder derselben!) Tabelle als eindeutige Verweise genutzt werden: **Fremdschlüssel**, **referenzielle Integrität**
- etwa **Weingut** als Verweise auf **ERZEUGER**
- ein Fremdschlüssel ist ein **Schlüssel in einer „fremden“ Tabelle**

Fremdschlüssel /2

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

WEINE

WeinID	Name	Farbe	Jahrgang	Weingut → ERZEUGER
1042	La Rose ...	Rot	1998	Château La Rose
2168	Creek Shiraz	Rot	2003	Creek
3456	Zinfandel	Rot	2004	Helena
2171	Pinot Noir	Rot	2001	Creek
3478	Pinot Noir	Rot	1999	Helena
4711	Riesling ...	Weiß	1999	Müller
4961	Chardonnay	Weiß	2002	Bighorn

ERZEUGER

Weingut	Anbaugebiet	Region
Creek	Barossa Valley	Südaustralien
Helena	Napa Valley	Kalifornien
Château La Rose	Saint-Emilion	Bordeaux
Château La Pointe	Pomerol	Bordeaux
Müller	Rheingau	Hessen
Bighorn	Napa Valley	Kalifornien

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

SQL-Datendefinition

Die Anweisung `create table`

```
create table basisrelationenname (
             spaltenname_1 wertebereich_1 [not null],
             ...
             spaltenname_k wertebereich_k [not null])
```

- Wirkung dieses Kommandos ist sowohl
  - die Ablage des **Relationenschemas** im Data Dictionary, als auch
  - die Vorbereitung einer „**leeren Basisrelation**“ in der Datenbank

Löschen einer Tabelle: `drop table`

- komplettes Löschen einer Tabelle (Inhalt und Eintrag im Data Dictionary)

```
drop table basisrelationenname
```

Mögliche Wertebereiche in SQL

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

- `integer` (oder auch `integer4`, `int`),
- `smallint` (oder auch `integer2`),
- `float`(p) (oder auch kurz `float`),
- `decimal`(p,q) und `numeric`(p,q) mit jeweils q Nachkommastellen,
- `character`(n) (oder kurz `char`(n), bei $n = 1$ auch `char`) für Zeichenketten (Strings) fester Länge n,
- `character varying`(n) (oder kurz `varchar`(n) für Strings variabler Länge bis zur Maximallänge n,
- `bit`(n) oder `bit varying`(n) analog für Bitfolgen, und
- `date`, `time` bzw. `datetime` für Datums-, Zeit- und kombinierte Datums-Zeit-Angaben

Beispiel für `create table`

```
create table WEINE (
   WeinID int primary key,
   Name varchar(20) not null,
   Farbe varchar(10),
   Jahrgang int,
   Weingut varchar(20))
```

- `primary key` kennzeichnet Spalte als **Schlüsselattribut**

`create table` mit Fremdschlüssel

```
create table WEINE (
   WeinID int,
   Name varchar(20) not null,
   Farbe varchar(10),
   Jahrgang int,
   Weingut varchar(20),
   primary key(WeinID),
   foreign key(Weingut)
      references ERZEUGER(Weingut))
```

- `foreign key` kennzeichnet Spalte als **Fremdschlüssel**

Nullwerte

- `not null` schließt in bestimmten Spalten **Nullwerte** als Attributwerte aus
- Kennzeichnung von Nullwerte in SQL durch `null`; hier $\perp$
- `null` repräsentiert die Bedeutung „*Wert unbekannt*“, „*Wert nicht anwendbar*“ oder „*Wert existiert nicht*“, gehört aber zu keinem Wertebereich
- `null` kann in allen Spalten auftauchen, außer in Schlüsselattributen und den mit `not null` gekennzeichneten

OTTO VON GUERICKE
UNIVERSITÄT
MAGDEBURG

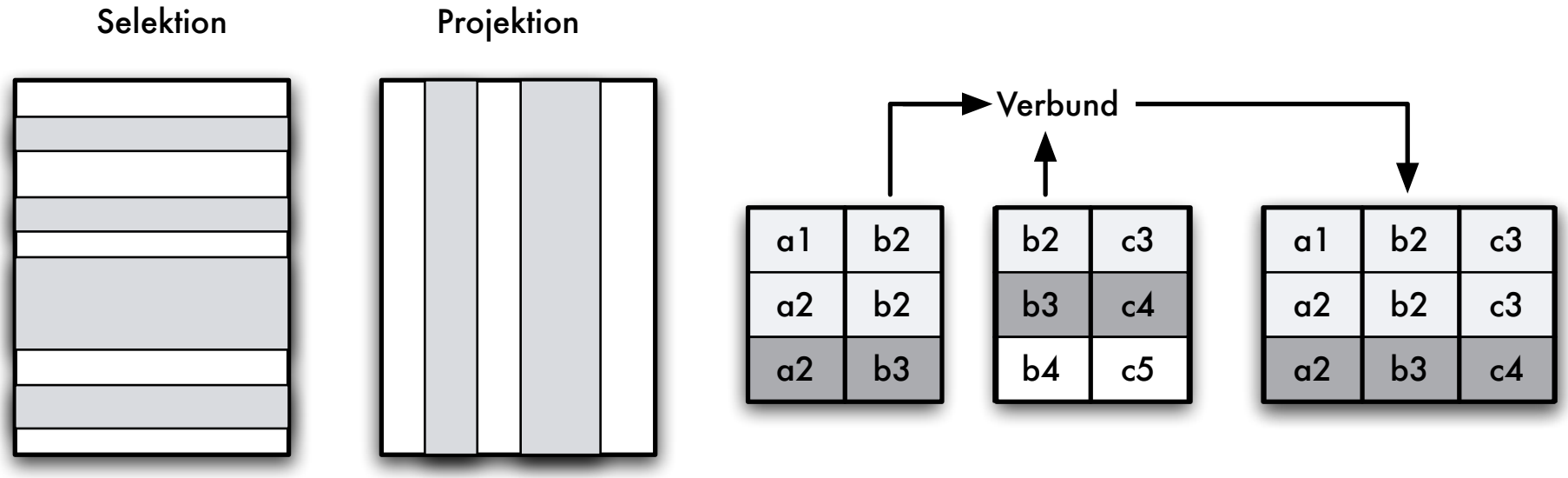
Grundoperationen: Die Relationenalgebra

Anfrageoperationen auf Tabellen

- **Basisoperationen** auf Tabellen, die die Berechnung von neuen Ergebnistabellen aus gespeicherten Datenbanktabellen erlauben
- Operationen werden zur sogenannten **Relationenalgebra** zusammengefasst
- Mathematik: Algebra ist definiert durch Wertebereich sowie darauf definierten Operationen

  $\rightarrow$ für Datenbankanfragen entsprechen die Inhalte der Datenbank den Werten, Operationen sind dagegen **Funktionen zum Berechnen der Anfrageergebnisse**
- Anfrageoperationen sind **beliebig kombinierbar** und bilden eine Algebra zum „Rechnen mit Tabellen“ – die **Relationenalgebra**

Relationenalgebra: Übersicht



Selektion σ

- **Selektion**: Auswahl von Zeilen einer Tabelle anhand eines Selektionsprädikats

$$\sigma_{\texttt{Jahrgang}>2000}(\texttt{WEINE})$$

WeinID	Name	Farbe	Jahrgang	Weingut
2168	Creek Shiraz	Rot	2003	Creek
3456	Zinfandel	Rot	2004	Helena
2171	Pinot Noir	Rot	2001	Creek
4961	Chardonnay	Weiß	2002	Bighorn

Projektion π

- **Projektion**: Auswahl von Spalten durch Angabe einer Attributliste

$$\pi_{\texttt{Region}}(\texttt{ERZEUGER})$$

Region
Südaustralien
Kalifornien
Bordeaux
Hessen

- **Die Projektion entfernt doppelte Tupel.**

Natürlicher Verbund $\bowtie$

- **Verbund** (engl. *join*): verknüpft Tabellen über **gleichbenannte Spalten**, indem er jeweils zwei Tupel verschmilzt, falls sie dort **gleiche Werte** aufweisen

Natürlicher Verbund: Beispiel

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

WEINE ⋈ ERZEUGER

WeinID	Name	...	Weingut	Anbaugebiet	Region
1042	La Rose Grand Cru	...	Ch. La Rose	Saint-Emilion	Bordeaux
2168	Creek Shiraz	...	Creek	Barossa Valley	Südaustralien
3456	Zinfandel	...	Helena	Napa Valley	Kalifornien
2171	Pinot Noir	...	Creek	Barossa Valley	Südaustralien
3478	Pinot Noir	...	Helena	Napa Valley	Kalifornien
4711	Riesling Reserve	...	Müller	Rheingau	Hessen
4961	Chardonnay	...	Bighorn	Napa Valley	Kalifornien

- Das Weingut „Château La Pointe“ ist im Ergebnis verschwunden ⇝ **Tupel, die keinen Partner finden (dangling tuples), werden eliminiert**

Kombination von Operationen

$$\pi_{\text{Name,Farbe,Weingut}}\big(\sigma_{\text{Jahrgang}>2000}(\texttt{WEINE}) \bowtie \sigma_{\text{Region}=\text{'Kalifornien'}}(\texttt{ERZEUGER})\big)$$

ergibt

Name	Farbe	Weingut
Zinfandel	Rot	Helena
Chardonnay	Weiß	Bighorn

Umbenennung β

- Anpassung von Attributnamen mittels **Umbenennung**:

WEINLISTE

Name
La Rose Grand Cru
Creek Shiraz
Zinfandel
Pinot Noir
Riesling Reserve

EMPFEHLUNG

Wein
La Rose Grand Cru
Riesling Reserve
Merlot Selection
Sauvignon Blanc

- Angleichen durch:

$$\beta_{\text{Name}\leftarrow\text{Wein}}\ (\texttt{EMPFEHLUNG})$$

Mengenoperationen

- **Vereinigung** $r_1 \cup r_2$ von zwei Relationen r_1 und r_2: Gesamtheit der beiden Tupelmengen
- Attributmengen beider Relationen müssen identisch sein

$$\texttt{WEINLISTE} \cup \beta_{\texttt{Name} \leftarrow \texttt{Wein}}(\texttt{EMPFEHLUNG})$$

Name
La Rose Grand Cru
Creek Shiraz
Zinfandel
Pinot Noir
Riesling Reserve
Merlot Selection
Sauvignon Blanc

Mengenoperationen /2

- **Differenz** $r_1 - r_2$ eliminiert die Tupel aus der ersten Relation, die auch in der zweiten Relation vorkommen

$$\texttt{WEINLISTE} - \beta_{\texttt{Name} \leftarrow \texttt{Wein}}(\texttt{EMPFEHLUNG})$$

ergibt:

Name
Creek Shiraz
Zinfandel
Pinot Noir

Mengenoperationen /3

- **Durchschnitt** $r_1 \cap r_2$: ergibt die Tupel, die in beiden Relationen gemeinsam vorkommen

$$\texttt{WEINLISTE} \cap \beta_{\texttt{Name} \leftarrow \texttt{Wein}}(\texttt{EMPFEHLUNG})$$

liefert:

Name
La Rose Grand Cru
Riesling Reserve

OTTO VON GUERICKE
UNIVERSITÄT
MAGDEBURG

SQL als Anfragesprache

SQL-Anfrage als Standardsprache

- Anfrage an eine einzelne Tabelle

```
select Name, Farbe
from WEINE
where Jahrgang = 2002
```

- SQL hat **Multimengensemantik** — Duplikate in Tabellen werden in SQL nicht automatisch unterdrückt!
- Mengensemantik durch `distinct`

```
select distinct Name from WEINE
```

Verknüpfung von Tabellen

- Kreuzprodukt als Basisverknüpfung

```
select *
from WEINE, ERZEUGER
```

- Verbund durch Operator `natural join`

```
select *
from WEINE natural join ERZEUGER
```

Verknüpfung von Tabellen /2

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

- Verbund alternativ durch Angabe einer **Verbundbedingung**!

```
select *
from WEINE, ERZEUGER
where WEINE.Weingut = ERZEUGER.Weingut
```

Kombination von Bedingungen

- Ausdruck in Relationenalgebra

$$\pi_{\text{Name,Farbe,Weingut}}\big(\sigma_{\text{Jahrgang}>2000}(\texttt{WEINE}) \bowtie \sigma_{\text{Region}=\text{'Kalifornien'}}(\texttt{ERZEUGER})\big)$$

- Anfrage in SQL

```
select Name, Farbe, WEINE.Weingut
from WEINE, ERZEUGER
where Jahrgang > 2000 and
      Region = 'Kalifornien' and
      WEINE.Weingut = ERZEUGER.Weingut
```

Mengenoperationen in SQL

- Vereinigung in SQL explizit mit `union`
- Differenzbildung durch geschachtelte Anfragen

```
select *
from WINZER
where Name not in (
      select Nachname
      from KRITIKER)
```

Änderungsoperationen in SQL

Änderungsoperationen in SQL

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

- `insert`: **Einfügen** eines oder mehrerer Tupel in eine Basisrelation oder Sicht
- `update`: **Ändern** von einem oder mehreren Tupel in einer Basisrelation oder Sicht
- `delete`: **Löschen** eines oder mehrerer Tupel aus einer Basisrelation oder Sicht
- **Lokale und globale Integritätsbedingungen müssen bei Änderungsoperationen automatisch vom System überprüft werden**

Die update-Anweisung

- Syntax:

```
update basisrelation
set    attribut_1 = ausdruck_1
       ...
       attribut_n = ausdruck_n
       [ where bedingung ]
```

Beispiel für update

WEINE

WeinID	Name	Jahrgang	Weingut	Preis
2168	Creek Shiraz	2003	Creek	7.99
3456	Zinfandel	2004	Helena	5.99
2171	Pinot Noir	2001	Creek	10.99
3478	Pinot Noir	1999	Helena	19.99
4711	Riesling Reserve	1999	Müller	14.99
4961	Chardonnay	2002	Bighorn	9.90

```
update WEINE
set Preis = Preis * 1.10
where Jahrgang < 2000
```

Beispiel für update: neue Werte

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

WEINE

WeinID	Name	Jahrgang	Weingut	Preis
2168	Creek Shiraz	2003	Creek	7.99
3456	Zinfandel	2004	Helena	5.99
2171	Pinot Noir	2001	Creek	10.99
3478	Pinot Noir	1999	Helena	21.99
4711	Riesling Reserve	1999	Müller	16.49
4961	Chardonnay	2002	Bighorn	9.90

Weiteres zu update

- Realisierung von Eintupel-Operation mittels Primärschlüssel:

```
update WEINE
set Preis = 7.99
where WeinID = 3456
```

- Änderung der gesamten Relation:

```
update WEINE
set Preis = 11
```

Die delete-Anweisung

- Syntax:

```
delete
from basisrelation
[ where bedingung ]
```

- Löschen eines Tupels in der WEINE-Relation:

```
delete from WEINE
where WeinID = 4711
```

Weiteres zu `delete`

- Standardfall ist das Löschen mehrerer Tupel:

```
delete from WEINE
where Farbe = 'Weiß'
```

- Löschen der gesamten Relation:

```
delete from WEINE
```

Weiteres zu `delete` /2

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

- Löschoperationen können zur Verletzung von Integritätsbedingungen führen!
- Beispiel: Verletzung der Fremdschlüsseleigenschaft, falls es noch Weine von diesem Erzeuger gibt:

```
delete from ERZEUGER
where Anbaugebiet = 'Hessen'
```

Die `insert`-Anweisung

- Syntax:

```
insert
into basisrelation
   [ (attribut_1, ..., attribut_n) ]
values (konstante_1, ..., konstante_n)
```

- optionale Attributliste ermöglicht das Einfügen von unvollständigen Tupeln

insert-Beispiele

```
insert into ERZEUGER (Weingut, Region)
values ('Wairau Hills', 'Marlborough')
```

- nicht alle Attribute angegeben $\rightsquigarrow$ Wert des fehlenden Attribut `Land` wird `null`

```
insert into ERZEUGER
values ('Château Lafitte', 'Medoc', 'Bordeaux')
```

Einfügen von berechneten Daten

- Syntax:

```
insert
into basisrelation [ (attribut_1, ..., attribut_n) ]
   SQL-anfrage
```

- Beispiel:

```
insert into WEINE (
   select ProdID, ProdName, 'Rot', ProdJahr,
      'Château Lafitte'
   from LIEFERANT where LName = 'Wein-Kontor' )
```

OTTO VON GUERICKE
UNIVERSITÄT
MAGDEBURG

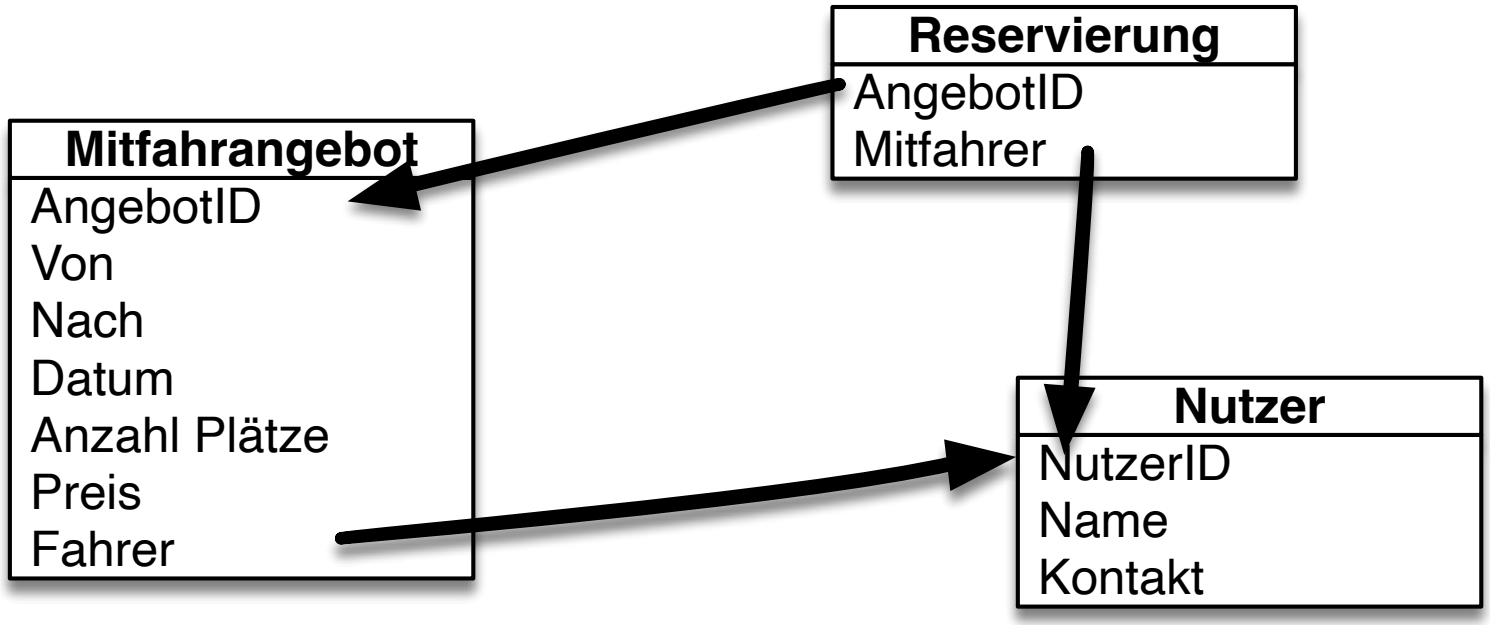
Anwendungsbeispiel

Mitfahrzentrale

- Welche Daten?
  - **Mitfahrangebote**: Wann? Von wo? Wohin? Wer? Plätze?
  - **Nutzer**: Anmeldung, Kontaktdaten
  - **Reservierung**: Wer? Welches Angebot?

CC-BY-2.0: Luo Shaoyang

Mitfahrzentrale: Datenbank



Mitfahrzentrale: Datenbank in SQL

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

```
create table Nutzer (
   NutzerID varchar(10) primary key,
   Name varchar(100),
   Kontakt varchar(500));
```

Mitfahrzentrale: Datenbank in SQL /2

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

```
create table Mitfahrangebot (
   AngebotID int primary key,
   Von varchar(100) not null,
   Nach varchar(100) not null,
   Datum date not null,
   AnzPlaetze int,
   Preis decimal,
   Fahrer varchar(10)
      references Nutzer(NutzerID));
```

Mitfahrzentrale: Datenbank in SQL /3

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

```
create table Reservierung (
   AngebotID int
      references Mitfahrangebot(AngebotID),
   Mitfahrer varchar(10)
      references Nutzer(NutzerID));
```

Mitfahrzentrale: Anfragen

- Welche Angebote gibt es heute von Ilmenau nach Erfurt?

```
select * from Mitfahrangebot
where Von = 'Ilmenau' and Nach = 'Erfurt'
   and Datum = date('now');
```

- Reservierung für eine bestimmte Mitfahrgelegenheit

```
insert into Reservierung values (1, 'holgi');
```

Mitfahrzentrale: Anfragen /2

- Wer will bei mir mitfahren?

```
select R.Mitfahrer
from Reservierung R, Mitfahrangebot M
where R.AngebotID = M.AngebotID
   and M.Fahrer = 'heike';
```

Zusammenfassung

- Relationenmodell: Datenbank als Sammlung von Tabellen
- Integritätsbedingungen im Relationenmodell
- Tabellendefinition in SQL
- Relationenalgebra: Anfrageoperatoren
- Grundkonzepte von SQL-Anfragen und -Änderungen

Kontrollfragen

- Was ist eine Relation?
- Was definiert die Relationenalgebra?
- Wie wird eine Realweltobjekt in einer relationalen Datenbank repräsentiert?
- Wie werden Tabellen in SQL definiert und manipuliert?
- Was sind Integritätsbedingungen?