

## 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

# Relationenmodell

- Konzeptuell ist die Datenbank eine **Menge von Tabellen**

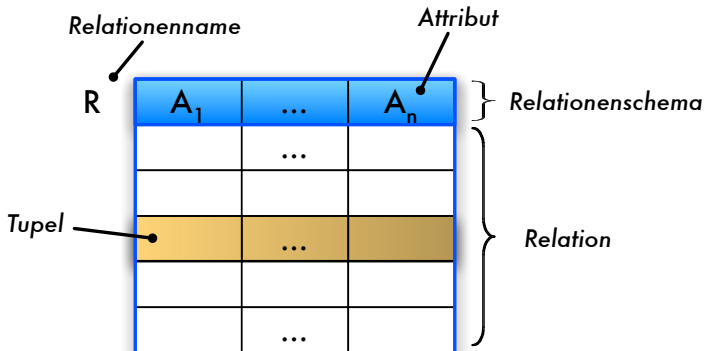
WEINE	WeinID	Name	Farbe	Jahrgang	Weingut
	1042	La Rose Grand Cru	Rot	1998	Château La Rose
	2168	Creek Shiraz	Rot	2003	Creek
	3456	Zinfandel	Rot	2004	Helena
	2171	Pinot Noir	Rot	2001	Creek
	3478	Pinot Noir	Rot	1999	Helena
	4711	Riesling Reserve	Weiß	1999	Müller
	4961	Chardonnay	Weiß	2002	Bighorn

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

- Tabelle = „**Relation**“

# Darstellung von Relationen und Begriffe

- **Fett** geschriebene Zeilen: **Relationenschema**
- Weitere Einträge in der Tabelle: **Relation**
- Eine Zeile der Tabelle: **Tupel**
- Eine Spaltenüberschrift: **Attribut**
- Ein Eintrag: **Attributwert**



## Integritätsbedingungen: Schlüssel

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

ERZEUGER	<u>Weingut</u>	Anbaugebiet	Region
	Creek	Barossa Valley	South Australia
	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

<b>WEINE</b>	<u>WeinID</u>	Name	Farbe	Jahrgang	Weingut → ERZEUGER
	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

<b>ERZEUGER</b>	<u>Weingut</u>	Anbaugebiet	Region
	Creek	Barossa Valley	South Australia
	Helena	Napa Valley	Kalifornien
	Château La Rose	Saint-Emilion	Bordeaux
	Château La Pointe	Pomerol	Bordeaux
	Müller	Rheingau	Hessen
	Bighorn	Napa Valley	Kalifornien

## Die Anweisung `create table`

```
create table basisrelationenname (  
    spaltenname1 wertebereich1 [not null],  
    ...  
    spaltennamek wertebereichk [not null])
```

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

## Mögliche Wertebereiche in SQL

- **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. **timestamp** für Datums-, Zeit- und kombinierte Datums-Zeit-Angaben

## Beispiel für `create table`

```
create table WEINE (  
  WeinID int not null,  
  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 WeinFarbe,  
    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

## Weiteres zur Datendefinition in SQL

- Neben Primär- und Fremdschlüsseln können in SQL angegeben werden:
  - ▶ mit der **default**-Klausel Defaultwerte für Attribute,
  - ▶ mit der **create domain**-Anweisung benutzerdefinierte Wertebereiche und
  - ▶ mit der **check**-Klausel weitere lokale Integritätsbedingungen innerhalb der zu definierenden Wertebereiche, Attribute und Relationenschemata

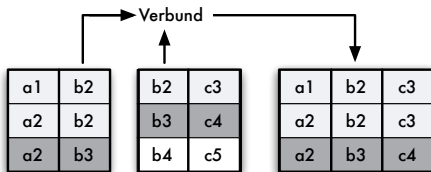
# Anfrageoperationen auf Tabellen

- **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  
→ für Datenbankabfragen 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 sogenannte relationale Algebra oder auch Relationenalgebra

# Relationenalgebra: Übersicht

Selektion


Projektion

## Selektion $\sigma$

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

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

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 $\pi$

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

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

<b>Region</b>
South Australia
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

### WEINE $\bowtie$ ERZEUGER

WeinID	Name	...	Weingut	Anbaugebiet	Region
1042	La Rose Grand Cru	...	Ch. La Rose	Saint-Emilion	Bordeaux
2168	Creek Shiraz	...	Creek	Barossa Valley	South Australia
3456	Zinfandel	...	Helena	Napa Valley	Kalifornien
2171	Pinot Noir	...	Creek	Barossa Valley	South Australia
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  
 $\rightsquigarrow$  **Tupel, die keinen Partner finden (*dangling tuples*), werden eliminiert**

# Kombination von Operationen

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

ergibt

Name	Farbe	Weingut
Zinfandel	Rot	Helena
Chardonnay	Weiß	Bighorn

# Umbenennung $\beta$

- 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:

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

# Mengenoperationen

- **Vereinigung**  $r_1 \cup r_2$  von zwei Relationen  $r_1$  und  $r_2$ : sammelt die Tupelmengen zweier Relationen unter einem gemeinsamen Schema auf
- Attributmengen beider Relationen müssen identisch sein

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

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

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

ergibt:

Name
------

Creek Shiraz
Zinfandel
Pinot Noir

## Mengenoperationen /3

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

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

liefert:

Name
La Rose Grand Cru
Riesling Reserve

# 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
```

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

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

# Kombination von Bedingungen

- Ausdruck in Relationenalgebra

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

- 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

- **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   attribut1 = ausdruck1
        ...
        attributn = ausdruckn
        [ where bedingung ]
```

## Beispiel für update

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

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

## Beispiel für **update**: neue Werte

<b>WEINE</b>	<b>WeinID</b>	<b>Name</b>	<b>Farbe</b>	<b>Jahrgang</b>	<b>Weingut</b>	<b>Preis</b>
	2168	Creek Shiraz	Rot	2003	Creek	7.99
	3456	Zinfandel	Rot	2004	Helena	5.99
	2171	Pinot Noir	Rot	2001	Creek	10.99
	3478	Pinot Noir	Rot	1999	Helena	21.99
	4711	Riesling Reserve	Weiß	1999	Müller	16.49
	4961	Chardonnay	Weiß	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

- 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  
    [ (attribut1, ..., attributn) ]  
values (konstante1, ..., konstanten)
```

- 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  
      [ (attribut1, ..., attributn) ]  
      SQL-anfrage
```

- Beispiel:

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

# Zusammenfassung

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