Teil IV

Datenbankentwurf

Datenbankentwurf

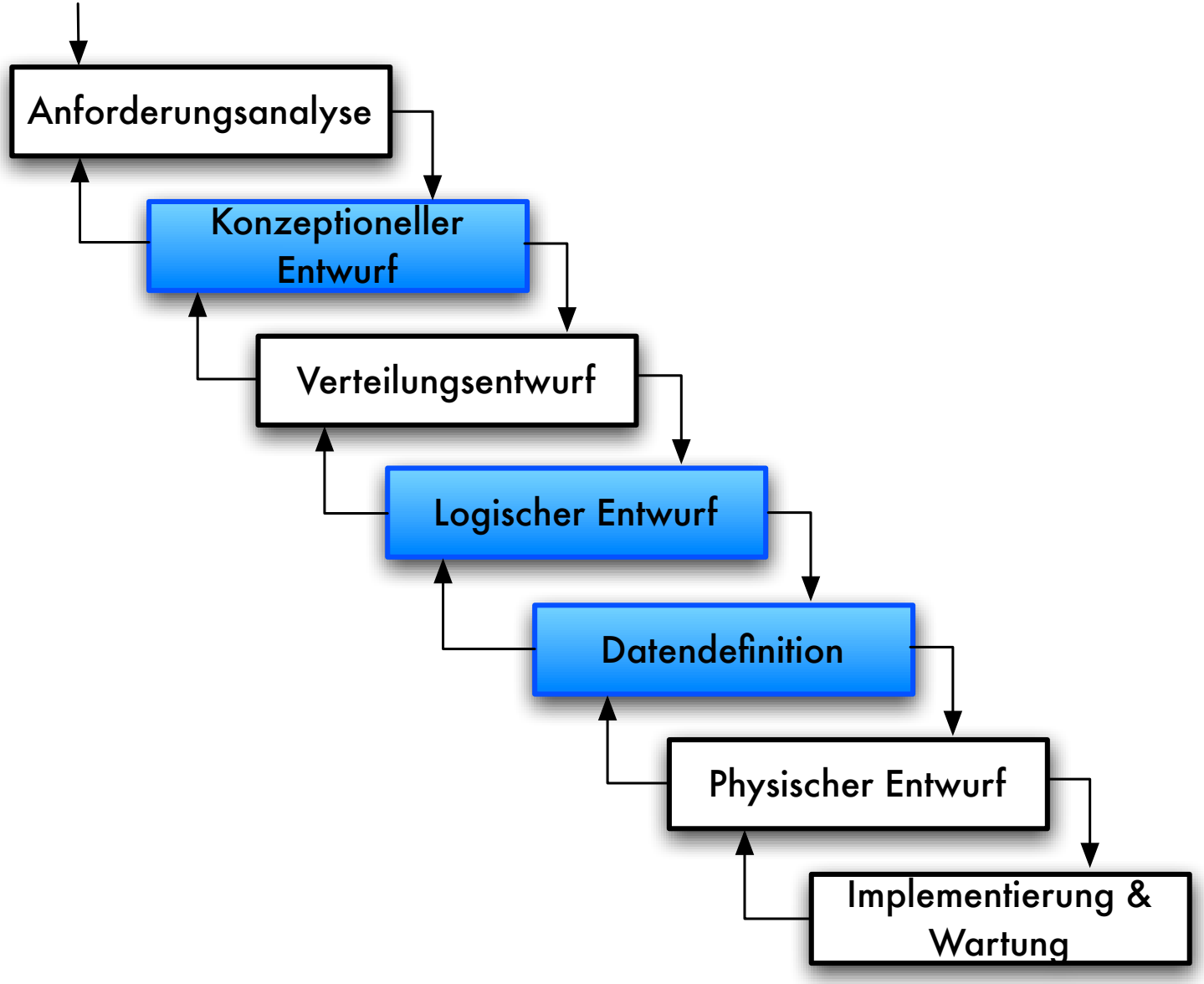
Lernziele für heute ...

- Kenntnisse über Ziele und Ablauf des Datenbankentwurfsprozesses
- Kenntnisse der Regeln zur Abbildung von ER-Schemata auf Relationenschemata

Entwurfsaufgabe

- Datenhaltung für mehrere Anwendungssysteme und mehrere Jahre
- daher: besondere Bedeutung
- Anforderungen an Entwurf
  - Anwendungsdaten jeder Anwendung sollen aus Daten der Datenbank ableitbar sein (und zwar möglichst effizient)
  - nur „vernünftige“ (wirklich benötigte) Daten sollen gespeichert werden
  - nicht-redundante Speicherung

Phasenmodell



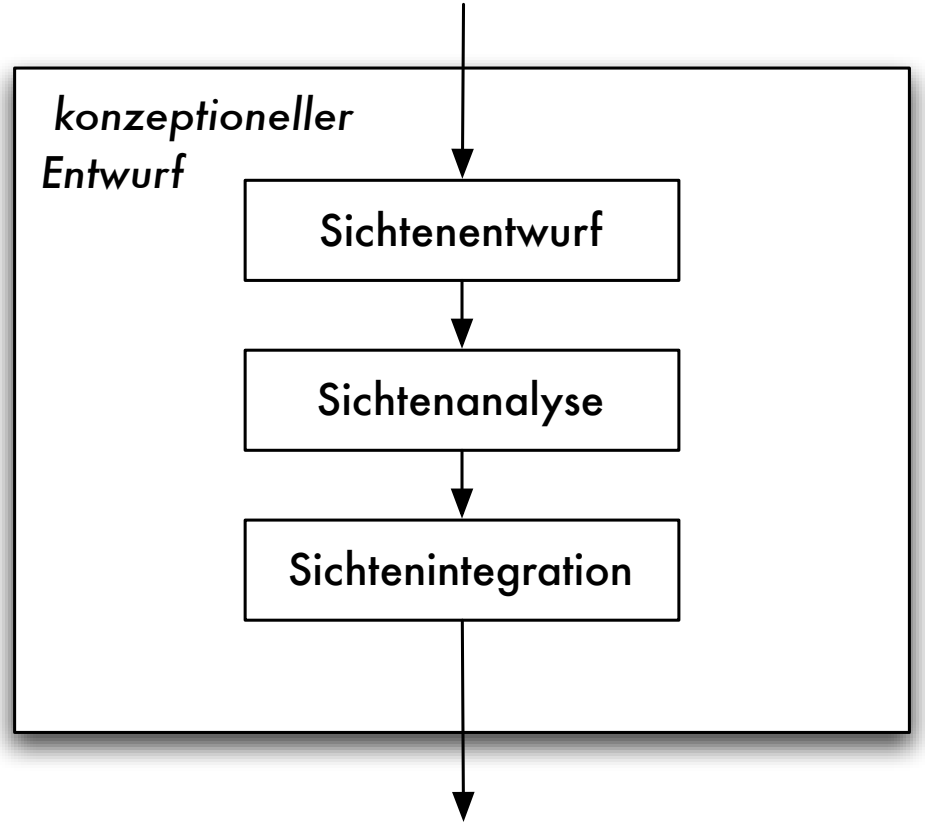
Anforderungsanalyse

- Vorgehensweise: Sammlung des Informationsbedarfs in den Fachabteilungen
- Ergebnis:
  - informale Beschreibung (Texte, tabellarische Aufstellungen, Formblätter, usw.) des Fachproblems
  - Trennen der Information über Daten (Datenanalyse) von den Information über Funktionen (Funktionsanalyse)
- „Klassischer“ DB-Entwurf:
  - nur Datenanalyse und Folgeschritte
- Funktionsentwurf:
  - siehe Methoden des Software Engineering
- **genauer: Softwaretechnik-Vorlesungen wie Requirements Engineering**

Konzeptioneller Entwurf

- erste formale Beschreibung des Fachproblems
- Sprachmittel: semantisches Datenmodell
- Vorgehensweise:
  - Modellierung von Sichten z.B. für verschiedene Fachabteilungen
  - Analyse der vorliegenden Sichten in Bezug auf Konflikte
  - Integration der Sichten in ein Gesamtschema
- Ergebnis: konzeptuelles Gesamtschema, z.B. ER-Diagramm
- **siehe erste Übung**

Phasen des konzeptionellen Entwurf

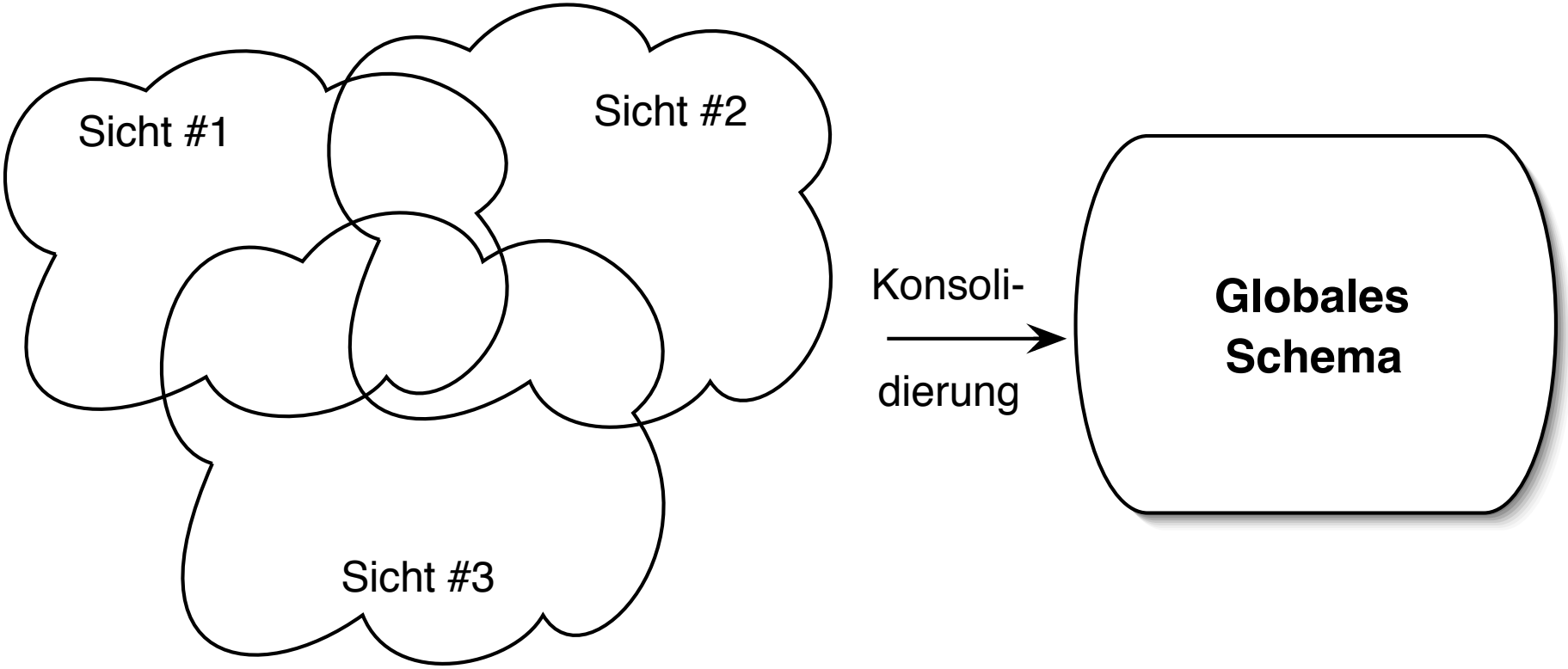


Weiteres Vorgehen beim Entwurf

- ER-Modellierung von verschiedenen Sichten auf Gesamtinformation, z.B. für verschiedene Fachabteilungen eines Unternehmens $\rightsquigarrow$ konzeptioneller Entwurf
  - Analyse und Integration der Sichten
  - Ergebnis: konzeptuelles Gesamtschema
- Verteilungsentwurf bei verteilter Speicherung
- Abbildung auf konkretes Implementierungsmodell (z.B. Relationenmodell) $\rightsquigarrow$ logischer Entwurf
- Datendefinition, Implementierung und Wartung $\rightsquigarrow$ physischer Entwurf

Sichtenintegration

- Analyse der vorliegenden Sichten in Bezug auf Konflikte
- Integration der Sichten in ein Gesamtschema



Integrationskonflikte

- **Namenskonflikte**: Homonyme / Synonyme
  - Homonyme: Schloss; Kunde
  - Synonyme: Auto, KFZ, Fahrzeug
- **Typkonflikte**: verschiedene Strukturen für das gleiche Element
- **Wertebereichskonflikte**: verschiedene Wertebereiche für ein Element
- **Bedingungskonflikte**: z.B. verschiedene Schlüssel für ein Element
- **Strukturkonflikte**: gleicher Sachverhalt durch unterschiedliche Konstrukte ausgedrückt

genauer: Data Science im Bachelor, GDBF im Master

Verteilungsentwurf

- sollen Daten auf mehreren Rechnern verteilt vorliegen, muss Art und Weise der verteilten Speicherung festgelegt werden
- z.B. bei einer Relation
  `KUNDE (KNr, Name, Adresse, PLZ, Konto)`
  - horizontale Verteilung:
    ```
    KUNDE_1 (KNr, Name, Adresse, PLZ, Konto)
    where PLZ < 50.000
    KUNDE_2 (KNr, Name, Adresse, PLZ, Konto)
    where PLZ >= 50.000
    ```
  - vertikale Verteilung (Verbindung über `KNr` Attribut):
    ```
    KUNDE_Adr (KNr, Name, Adresse, PLZ)
    KUNDE_Konto (KNr, Konto)
    ```

genauer: Datenbanken III im Master

Logischer Entwurf

- Sprachmittel: Datenmodell des ausgewählten „Realisierungs“-DBMS z.B. relationales Modell
- Vorgehensweise:
  1. (automatische) Transformation des konzeptuellen Schemas z.B. ER $\rightarrow$ relationales Modell
  2. Verbesserung des relationalen Schemas anhand von Gütekriterien (Normalisierung, siehe Kapitel 5):
     Entwurfsziele: Redundanzvermeidung, ...
- Ergebnis: logisches Schema, z.B. Sammlung von Relationenschemata

siehe zweite Übung für Schritt 1 und dritte Übung für Schritt 2

Datendefinition

- Umsetzung des logischen Schemas in ein konkretes Schema
- Sprachmittel: DDL und DML eines DBMS z.B. Oracle, DB2, SQL Server
  - Datenbankdeklaration in der DDL des DBMS
  - Realisierung der Integritätssicherung
  - Definition der Benutzersichten

Physischer Entwurf

- Ergänzen des physischen Entwurfs um Zugriffsunterstützung bzgl. Effizienzverbesserung, z.B. Definition von Indexen
- Index
  - Zugriffspfad: Datenstruktur für zusätzlichen, schlüsselbasierten Zugriff auf Tupel ($\langle$*Schlüsselattributwert, Tupeladresse*$\rangle$)
  - meist als B^+-Baum realisiert
- Sprachmittel: *Speicherstruktursprache* SSL

genauer: Bachelor-Vorlesung Datenbanken II

Indexe in SQL

```
create [ unique ] index indexname
   on relname (
      attrname [ asc | desc ],
      attrname [ asc | desc ],
   ...
)
```

- Beispiel

```
create index WeinIdx on WEINE (Name)
```

Notwendigkeit für Zugriffspfade

- Beispiel: Tabelle mit 100 GB Daten, Festplattentransferrate ca. 50 MB/s
- Operation: Suchen eines Tupels (Selektion)
- Implementierung: sequentielles Durchsuchen
- Aufwand: $102.400/50 = 2.048$ sec. ≈ 34 min.

Mit Zugriffspfad

- Möglichst nur das gesuchte Tupel in Hauptspeicher übertragen ($<$ ms)
- Dazu einige Blöcke mit wenigen MB übertragen: einige Blöcke Zugriffspfad, ein Block aus der gespeicherten Relation ($<$ s), üblicherweise insgesamt einstellige Anzahl von Blöcken

Implementierung und Wartung

- Phasen
  - der Wartung,
  - der weiteren Optimierung der physischen Ebene,
  - der Anpassung an neue Anforderungen und Systemplattformen,
  - der Portierung auf neue Datenbankmanagementsysteme
  - etc.

Kapazitätserhaltende Abbildung

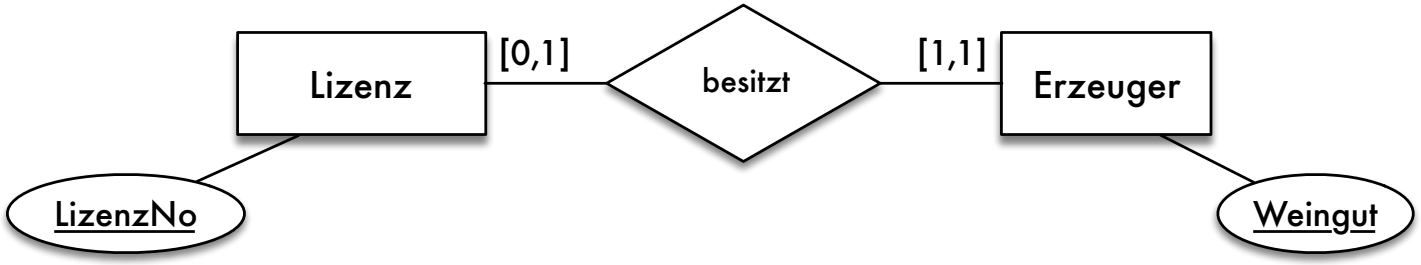
Wichtig bei Datenbankentwurfsschritten:

- Transformation von Datenbankbeschreibungen (Schemata) so, dass die Kapazität des Schemas erhalten bleibt
- Informationskapazität: Menge aller möglichen Instanzen zum Schema (im ER-Modell μ statt σ)
- Kapazitätserhaltung bei Transformation: im neuen Schema können genau so viele mögliche Instanzen dargestellt werden wie im alten
- Kapazitätserhöhung bei Transformation: im neuen Schema können mehr mögliche Instanzen dargestellt werden als im alten Schema
- Kapazitätsverminderung bei Transformation: im neuen Schema können weniger mögliche Instanzen dargestellt werden als im alten Schema

Unsere Forderung beim Datenbankentwurf:

- Kapazitätserhaltung
- Kapazitätserhöhung und Kapazitätsverminderung beim Datenbankentwurf nicht gewollt

Kapazitätserhöhende Abbildung



- Abbildung auf

$$R = \{\texttt{LizenzNo}, \texttt{Weingut}\}$$

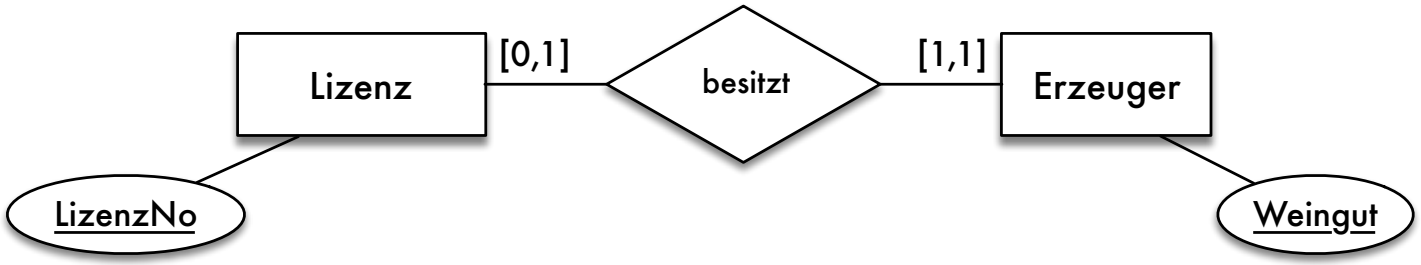
mit genau einem Schlüssel

$$K = \{\{\texttt{LizenzNo}\}\}$$

- mögliche ungültige Relation:

BESITZT	LizenzNo	Weingut
	007	Helena
	42	Helena

Kapazitätserhaltende Abbildung



- korrekte Ausprägung

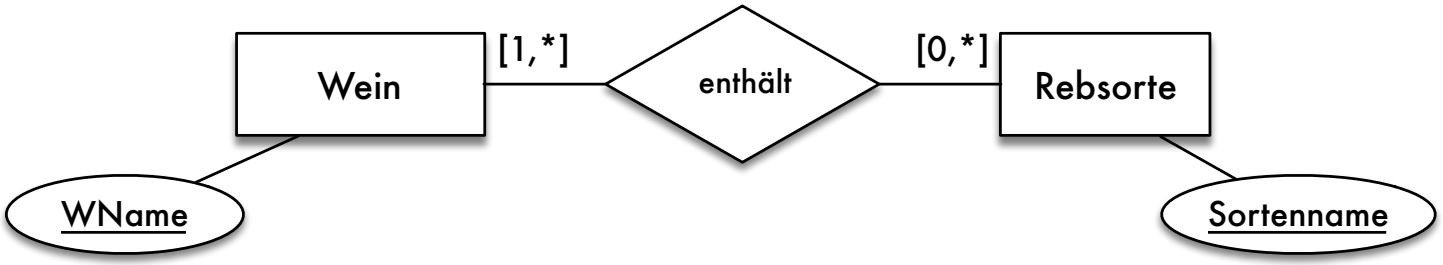
BESITZT

LizenzNo	Weingut
007	Helena
42	Müller

- korrekte Schlüsselmenge

$$K = \{\{\texttt{LizenzNo}\}, \{\texttt{Weingut}\}\}$$

Kapazitätsvermindernde Abbildung



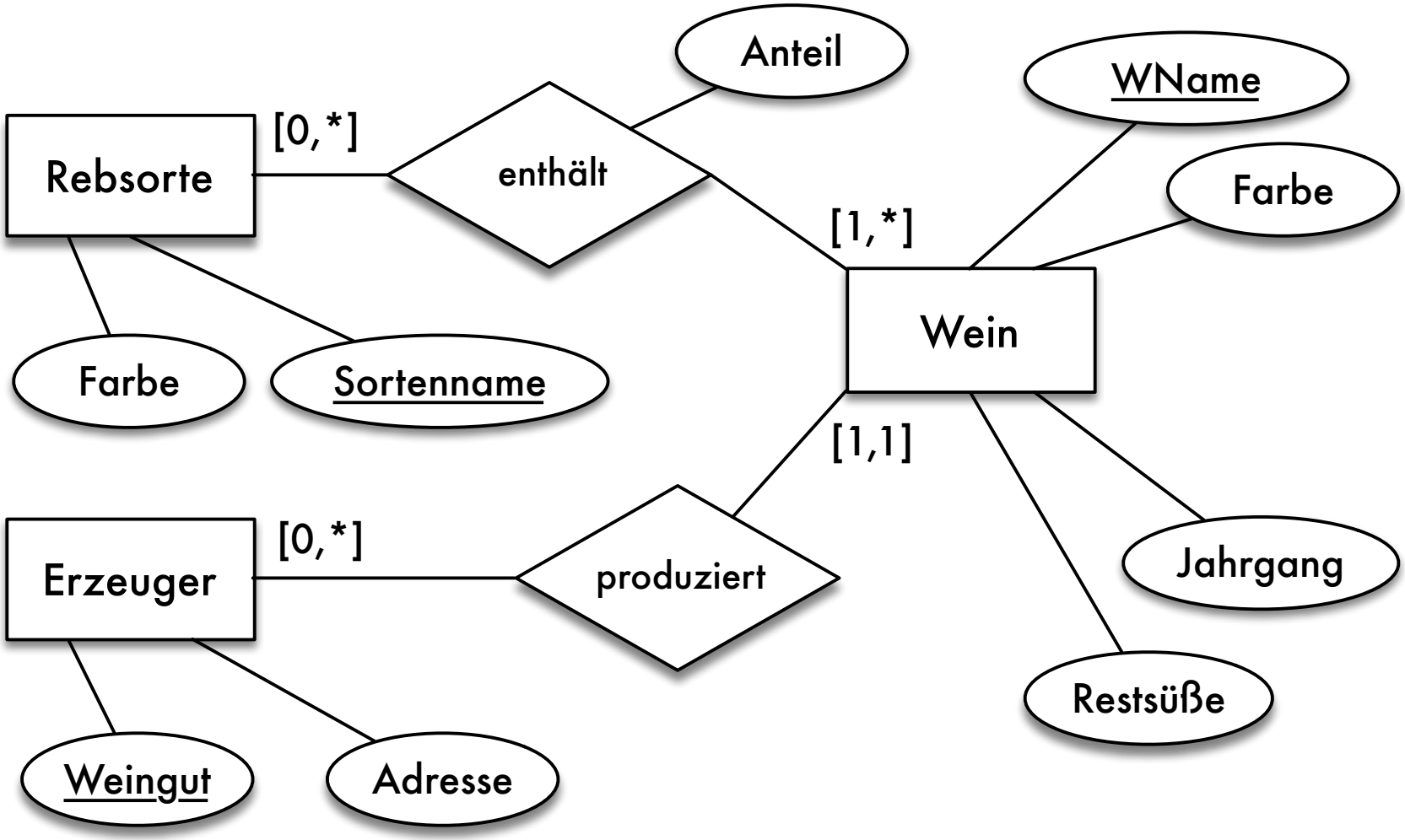
- Relationenschema mit einem Schlüssel $\{\texttt{WName}\}$
- als Ausprägung nicht mehr möglich:

ENTHÄLT	WName	Sortenname
	Zinfandel Red Blossom	Zinfandel
	Bordeaux Blanc	Cabernet Sauvignon
	Bordeaux Blanc	Muscadelle

- kapazitätserhaltend mit Schlüssel beider Entity-Typen im Relationenschema als neuer Schlüssel

$$K = \{\{\texttt{WName}, \texttt{Sortenname}\}\}$$

Beispielabbildung ER-RM: Eingabe



Beispielabbildung ER-RM: Ergebnis

1. $\texttt{REBSORTE} = \{\texttt{Farbe}, \texttt{Sortenname}\}$ mit $K_{\texttt{REBSORTE}} = \{\{\texttt{Sortenname}\}\}$
2. $\texttt{ENTHÄLT} = \{\texttt{Sortenname}, \texttt{WName}, \texttt{Anteil}\}$ mit $K_{\texttt{ENTHÄLT}} = \{\{\texttt{Sortenname}, \texttt{WName}\}\}$
3. $\texttt{WEIN} = \{\texttt{Farbe}, \texttt{WName}, \texttt{Jahrgang}, \texttt{Restsüße}\}$ mit $K_{\texttt{WEIN}} = \{\{\texttt{WName}\}\}$
4. $\texttt{PRODUZIERT} = \{\texttt{WName}, \texttt{Weingut}\}$ mit $K_{\texttt{PRODUZIERT}} = \{\{\texttt{WName}\}\}$
5. $\texttt{ERZEUGER} = \{\texttt{Weingut}, \texttt{Adresse}\}$ mit $K_{\texttt{ERZEUGER}} = \{\{\texttt{Weingut}\}\}$

ER-Abbildung auf Relationen

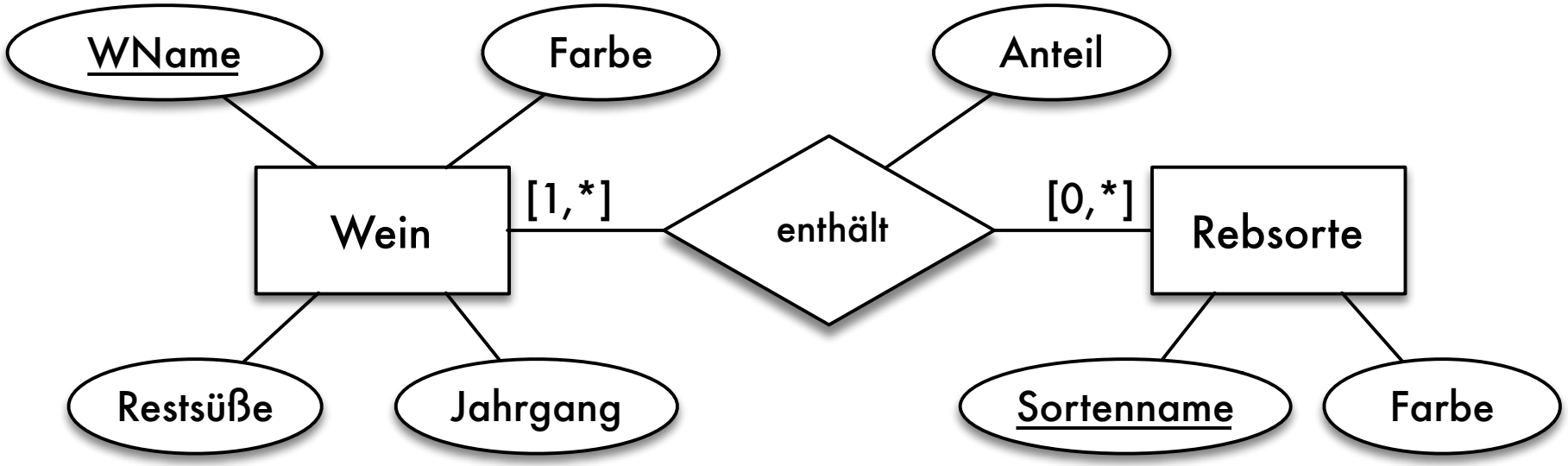
1. Entity-Typen und Beziehungstypen: jeweils auf Relationenschemata (7 Entity-Typen mit 9 Beziehungstypen ergeben 16 Relationenschemata)
2. Attribute:
   - bei Entity-Typen: Attribute und Schlüssel übernehmen
   - bei Beziehungstypen: Attribute übernehmen, Primärschlüssel der beteiligten Entity-Typen als Attribute des Relationenschemas übernehmen
3. Kardinalitäten der Beziehungen: durch Wahl der Schlüssel bei den zugehörigen Relationenschemata ausdrücken
4. in einigen Fällen: Verschmelzen der Relationenschemata von Entity- und Beziehungstypen
5. zwischen den verbleibenden Relationenschemata diverse Fremdschlüsselbedingungen einführen

Abbildung von Beziehungstypen

- neues Relationenschema mit allen Attributen des Beziehungstyps, zusätzlich Übernahme aller Primärschlüssel der beteiligten Entity-Typen
- **Festlegung der Schlüssel**:
  - m:n-Beziehung: $[\_, *][\_, *]$
    beide Primärschlüssel zusammen werden Schlüssel im neuen Relationenschema
  - 1:n-Beziehung: $[\_, *][\_, 1]$
    Primärschlüssel der n-Seite
    - bei der funktionalen Notation die Seite ohne Pfeilspitze
    - bei der Intervallnotation das Intervall mit $[\_, 1]$

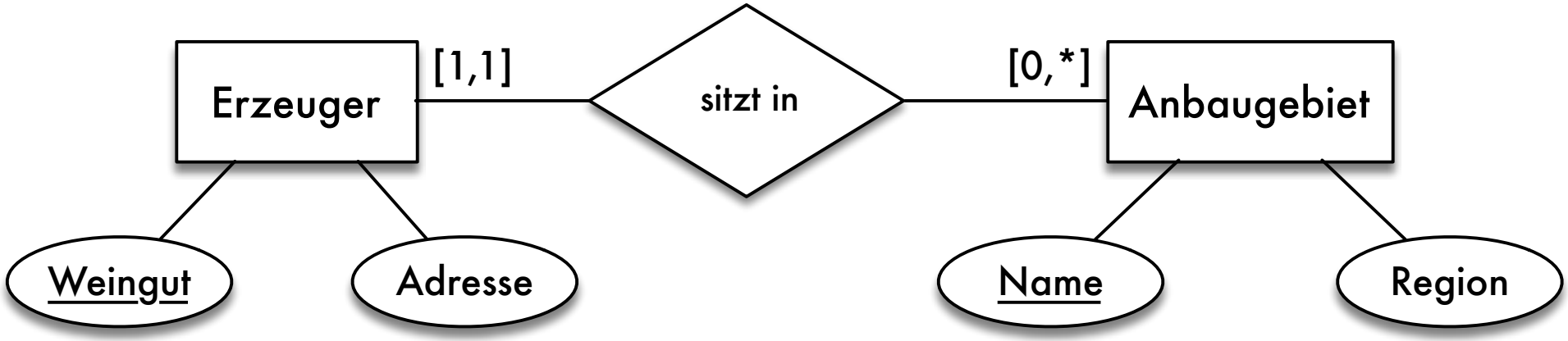
    wird Schlüssel im neuen Relationenschema
  - 1:1-Beziehung: $[\_, 1][\_, 1]$
    beide Primärschlüssel werden je ein Schlüssel im neuen Relationenschema, der Primärschlüssel wird dann aus diesen Schlüsseln gewählt

n:m-Beziehungen



- Umsetzung
  1. $\texttt{REBSORTE} = \{\texttt{Farbe}, \texttt{Sortenname}\}$ mit $K_{\texttt{REBSORTE}} = \{\{\texttt{Sortenname}\}\}$
  2. $\texttt{ENTHÄLT} = \{\texttt{Sortenname}, \texttt{WName}, \texttt{Anteil}\}$ mit $K_{\texttt{ENTHÄLT}} = \{\{\texttt{Sortenname}, \texttt{WName}\}\}$
  3. $\texttt{WEIN} = \{\texttt{Farbe}, \texttt{WName}, \texttt{Jahrgang}, \texttt{Restsüße}\}$ mit $K_{\texttt{WEIN}} = \{\{\texttt{WName}\}\}$
- Attribute `Sortenname` und `WName` sind gemeinsam Schlüssel

1:n-Beziehungen

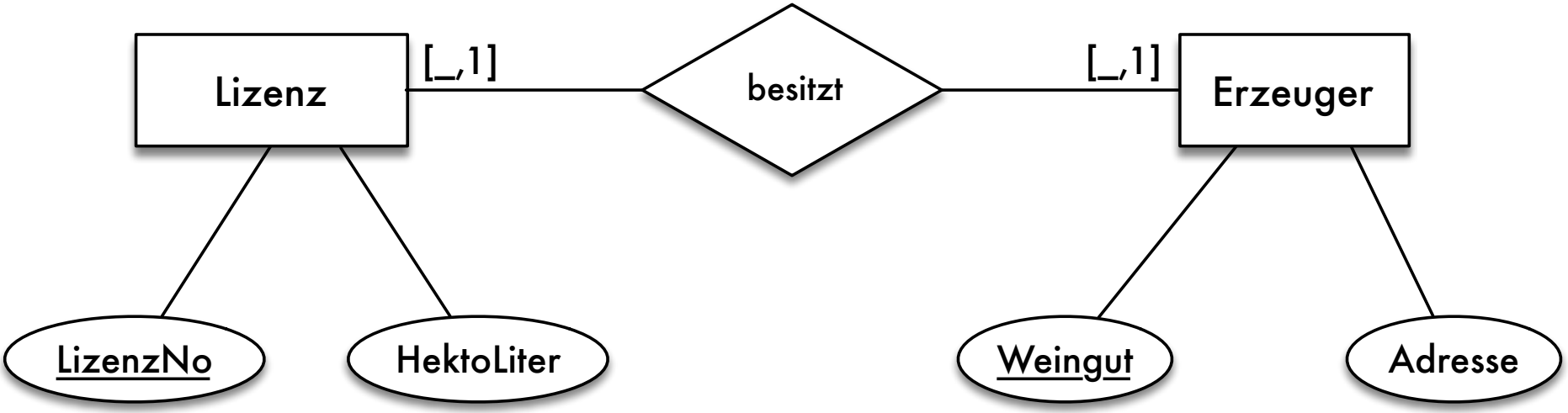


- Umsetzung (zunächst)
  - `ERZEUGER` mit den Attributen `Weingut` und `Adresse`,
  - `ANBAUGEBIET` mit den Attributen `Name` und `Region` und
  - `SITZT_IN` mit den Attributen `Weingut` und `Name` und dem Primärschlüssel der n-Seite `Weingut` als Primärschlüssel dieses Schemas.

Mögliche Verschmelzungen

- optionale Beziehungen ($[0, 1]$ oder $[0, *]$) nicht verschmelzen
- bei Kardinalitäten $[1, 1]$ oder $[1, *]$ (zwingende Beziehungen) Verschmelzung möglich:
  - 1:n-Beziehung: das Entity-Relationenschema der n-Seite kann in das Relationenschema der Beziehung integriert werden
  - 1:1-Beziehung: beide Entity-Relationenschemata können in das Relationenschema der Beziehung integriert werden
- **also immer genau die Relationen zu Entity-Typ und Beziehungstyp verschmelzen, die mit $[1, 1]$-Intervall verbunden sind**
- NICHT bei $[1, *]$ und $[0, *]$: Redundanzen entstehen, Normalformen verletzt (siehe nächstes Kapitel)
- NICHT bei $[0, 1]$ und $[0, *]$: Nullwerte werden in Fremdschlüssel oder sogar Schlüssel eingeschleppt
  - Nullwert bei Fremdschlüsseln unerwünscht = Nullwert-Anomalie
  - Nullwert bei Schlüsseln verboten

1:1-Beziehungen



- Umsetzung (zunächst)
  - `ERZEUGER` mit den Attributen `Weingut` und `Adresse`
  - `LIZENZ` mit den beiden Attributen `LizenzNo` und `Hektoliter`
  - `BESITZT` mit den Primärschlüsseln der beiden beteiligten Entity-Typen jeweils als Schlüssel dieses Schemas, also `LizenzNo` und `Weingut`

1:1-Beziehungen: Verschmelzung

- Umsetzung mit Verschmelzung
  - verschmolzene Relation:

ERZEUGER

Weingut	Adresse	LizenzNo	Hektoliter
Rotkäppchen	Freiberg	42-007	10.000
Weingut Müller	Dagstuhl	42-009	250

  - Erzeuger ohne Lizenz erfordern Nullwerte:

ERZEUGER

Weingut	Adresse	LizenzNo	Hektoliter
Rotkäppchen	Freiberg	42-007	10.000
Weingut Müller	Dagstuhl	$\perp$	$\perp$

  - freie Lizenzen führen zu weiteren Nullwerten:

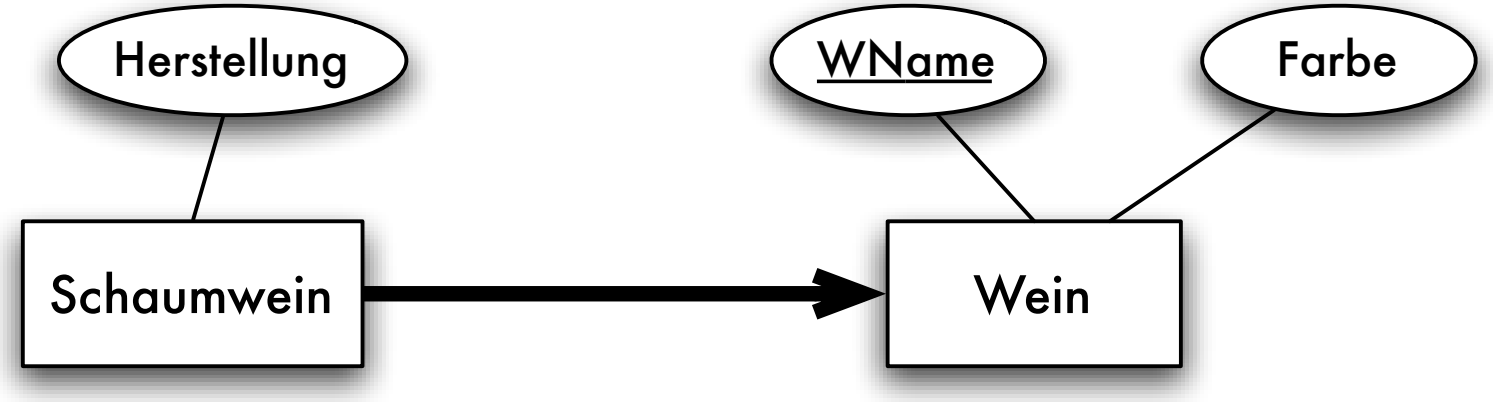
ERZEUGER

Weingut	Adresse	LizenzNo	Hektoliter
Rotkäppchen	Freiberg	42-007	10.000
Weingut Müller	Dagstuhl	$\perp$	$\perp$
$\perp$	$\perp$	42-003	100.000

Abhängige Entity-Typen

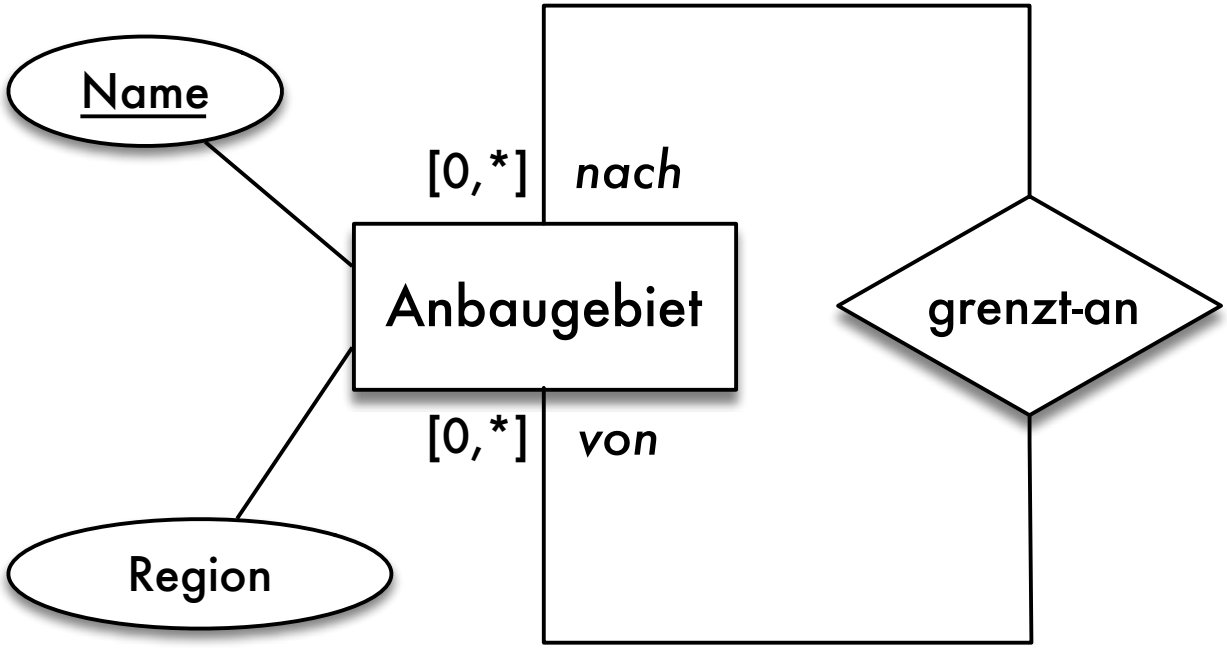
- Umsetzung
  1. $\texttt{WEINJAHRGANG} = \{\texttt{WName}, \texttt{Restsüße}, \texttt{Jahr}\}$ mit $K_{\texttt{WEINJAHRGANG}} = \{\{\texttt{WName}, \texttt{Jahr}\}\}$
  2. $\texttt{WEIN} = \{\texttt{Farbe}, \texttt{WName}\}$ mit $K_{\texttt{WEIN}} = \{\{\texttt{WName}\}\}$
     - Attribut `WName` in `WEINJAHRGANG` ist Fremdschlüssel zur Relation `WEIN`
- Also insgesamt 2 Relationenschemata (begründbar mit zugehörigen Intervallen $[1, 1]$ und $[0, 1]$)

IST-Beziehung



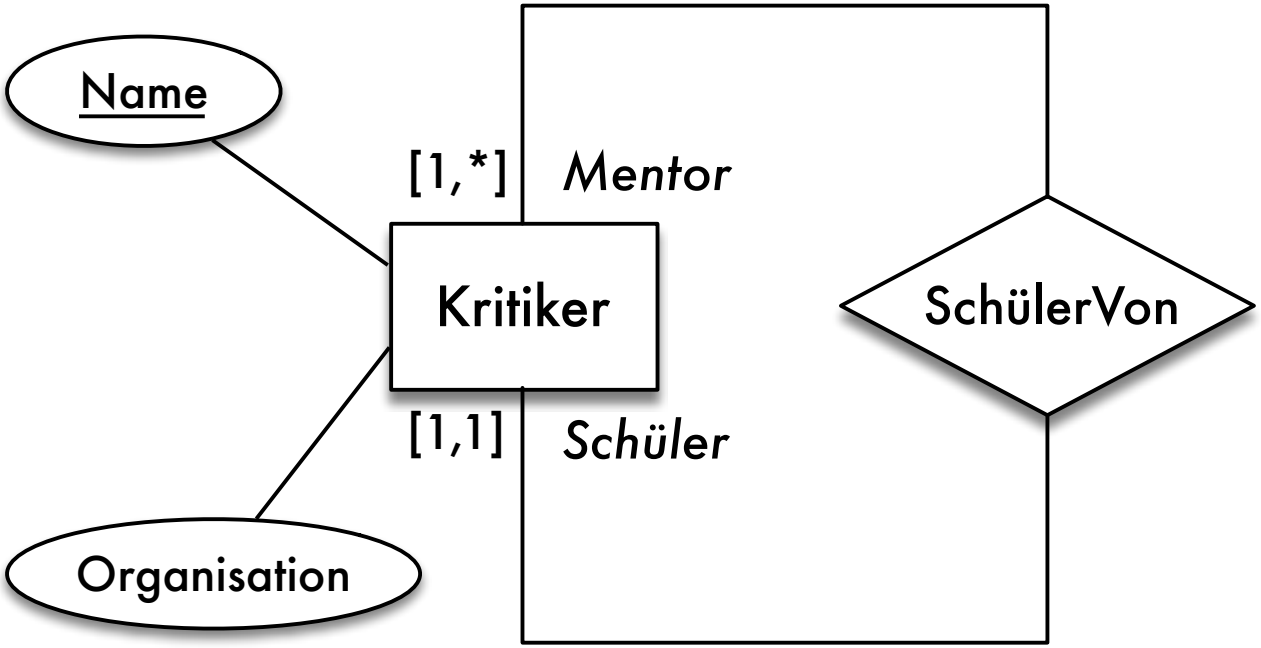
- Umsetzung
  1. $\texttt{WEIN} = \{\texttt{Farbe}, \texttt{WName}\}$ mit $K_{\texttt{WEIN}} = \{\{\texttt{WName}\}\}$
  2. $\texttt{SCHAUMWEIN} = \{\texttt{WName}, \texttt{Herstellung}\}$ mit $K_{\texttt{SCHAUMWEIN}} = \{\{\texttt{WName}\}\}$
     - WName in SCHAUMWEIN ist Fremdschlüssel bezüglich der Relation WEIN
- Also insgesamt 2 Relationenschemata (begründbar mit zugehörigen Intervallen $[1, 1]$ und $[0, 1]$)

Rekursive Beziehungen



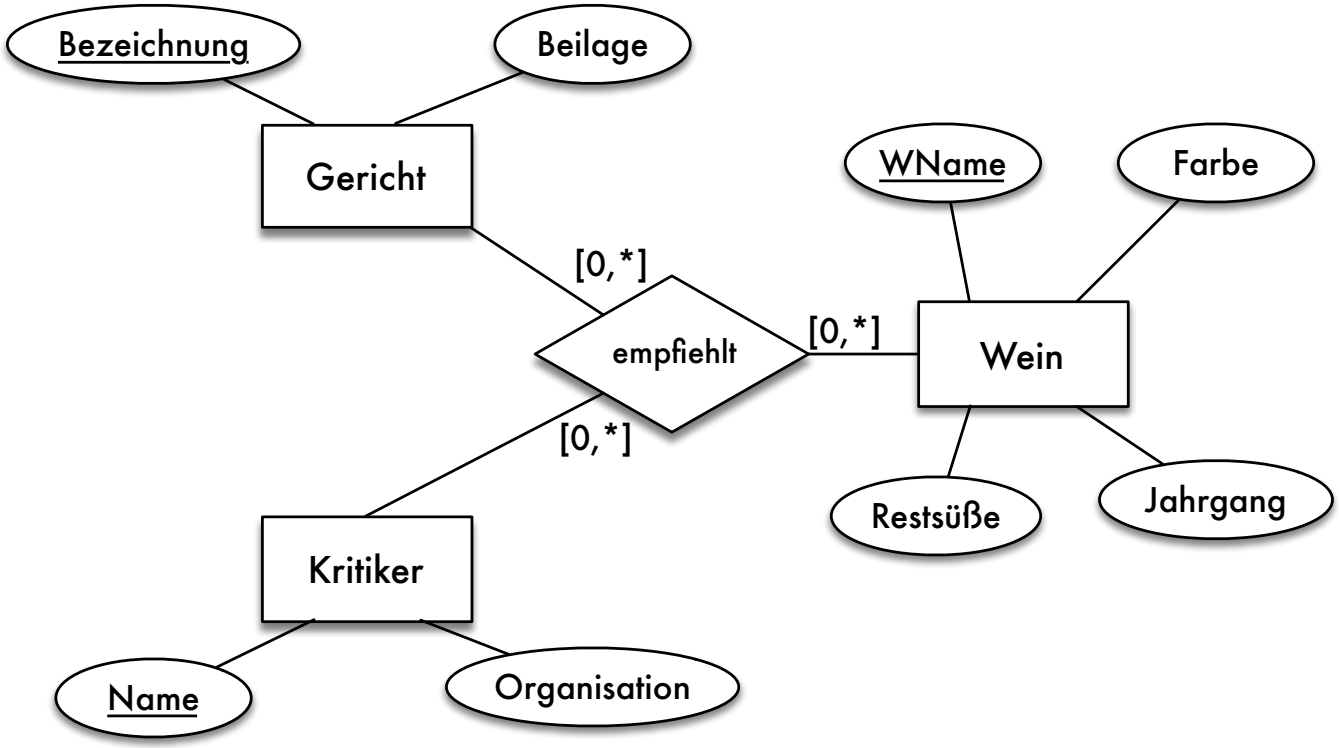
- Umsetzung
  1. $\texttt{ANBAUGEBIET} = \{\texttt{Name}, \texttt{Region}\}$ mit $K_{\texttt{ANBAUGEBIET}} = \{\{\texttt{Name}\}\}$
  2. $\texttt{GRENZT\_AN} = \{\texttt{nach}, \texttt{von}\}$ mit $K_{\texttt{GRENZT\_AN}} = \{\{\texttt{nach}, \texttt{von}\}\}$

Rekursive funktionale Beziehungen



- Umsetzung (falls Schüler Mentor haben muss: $[1, 1]$)
  1. $\texttt{KRITIKER} = \{\texttt{Name}, \texttt{Organisation}, \texttt{Mentorname}\}$ mit $K_{\texttt{KRITIKER}} = \{\{\texttt{Name}\}\}$
     - `Mentorname` ist Fremdschlüssel auf das Attribut `Name` der Relation `KRITIKER`.

Mehrstellige Beziehungen



- jeder beteiligte Entity-Typ wird nach den obigen Regeln behandelt
- für Beziehung `Empfiehlt` werden Primärschlüssel der drei beteiligten Entity-Typen in das resultierende Relationenschema aufgenommen
- Beziehung ist allgemeiner Art (k:m:n-Beziehung): alle Primärschlüssel bilden zusammen den Schlüssel

Mehrstellige Beziehungen: Ergebnis

1. $\texttt{EMPFIEHLT} = \{\texttt{WName}, \texttt{Bezeichnung}, \texttt{Name}\}$ mit $K_{\texttt{EMPFIEHLT}} = \{\{\texttt{WName}, \texttt{Bezeichnung}, \texttt{Name}\}\}$
2. $\texttt{GERICHT} = \{\texttt{Bezeichnung}, \texttt{Beilage}\}$ mit $K_{\texttt{GERICHT}} = \{\{\texttt{Bezeichnung}\}\}$
3. $\texttt{WEIN} = \{\texttt{Farbe}, \texttt{WName}, \texttt{Jahrgang}, \texttt{Restsüße}\}$ mit $K_{\texttt{WEIN}} = \{\{\texttt{WName}\}\}$
4. $\texttt{KRITIKER} = \{\texttt{Name}, \texttt{Organisation}\}$ mit $K_{\texttt{KRITIKER}} = \{\{\texttt{Name}\}\}$

- Die drei Schlüsselattribute von EMPFIEHLT sind Fremdschlüssel für die jeweiligen Ursprungsrelationen.

Übersicht über die Transformationen

ER-Konzept	wird abgebildet auf relationales Konzept
Entity-Typ E_i Attribute von E_i Primärschlüssel P_i	Relationenschema R_i Attribute von R_i Primärschlüssel P_i
Beziehungstyp dessen Attribute 1:n (Intervalle [_,*] und [_,1]) 1:1 (Intervalle [_,1] und [_,1]) n:m (Intervalle [_,*] und [_,*])	Relationenschema Attribute: P_1, P_2 weitere Attribute P_2 wird Primärschlüssel der Beziehung P_1 und P_2 werden jeweils Schlüssel der Beziehung $P_1 \cup P_2$ wird Primärschlüssel der Beziehung
IST-Beziehung	R_1 erhält zusätzlichen Schlüssel P_2
weak entity E_1	erhält Schlüssel P_2 des „starken“ Entity-Typs E_2 als Fremdschlüssel, Schlüssel von E_1 ist dann $P_1 \cup P_2$

Bezeichnungen:

E_1, E_2: an Beziehung beteiligte Entity-Typen

P_1, P_2: deren Primärschlüssel

bei 1:n-Beziehung: E_2 ist die n-Seite, in der Intervallnotation die mit dem Intervall [_,1]

bei IST-Beziehung: E_1 ist speziellerer Entity-Typ

Zusammenfassung

- Phasen des Datenbankentwurfs
- Ziel: Kapazitätserhaltung
- Regeln zur Abbildung von ER-Schemata auf Relationenschemata
- Besondere Rolle der Kardinalitäten und ihre Auswirkung auf die Schlüsselwahl
- weitere Entwurfsschritte
- *Basis: Kapitel 6 von [SSH18] (Biberbuch 1)*

Kontrollfragen

- Welche Schritte umfasst der Datenbankentwurfsprozess?
- Welche Forderungen müssen die Abbildungen (Transformationen) zwischen den einzelnen Entwurfsschritten erfüllen? Warum?
- Wie werden die Konzepte des ER-Modells auf die des Relationenmodell abgebildet?
- Wie werden die verschiedenen Kardinalitäten von Beziehungstypen bei der Abbildung berücksichtigt?

Teil V

Relationale Entwurfstheorie

Relationale Entwurfstheorie

Lernziele für heute ...

- Kenntnisse zur Verfeinerung des relationalen Entwurfs
- Verständnis der Normalformen
- Methodik und Verfahren zur Normalisierung

Relationenmodell

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

Begriffe des Relationenmodells

Begriff	**Informale Bedeutung**
Attribut	Spalte einer Tabelle
Wertebereich	mögliche Werte eines Attributs (auch Domäne)
Attributwert	Element eines Wertebereichs
Relationenschema	Menge von Attributen
Relation	Menge von Zeilen einer Tabelle
Tupel	Zeile einer Tabelle
Datenbankschema	Menge von Relationenschemata
Datenbank	Menge von Relationen (Basisrelationen)

Begriffe des Relationenmodells /2

Begriff	**Informale Bedeutung**
Schlüssel	minimale Menge von Attributen, deren Werte ein Tupel einer Tabelle eindeutig identifizieren
Primärschlüssel	ein beim Datenbankentwurf ausgezeichneter Schlüssel
Fremdschlüssel	Attributmenge, die in einer anderen Relation Schlüssel ist
Fremdschlüsselbedingung	alle Attributwerte des Fremdschlüssels tauchen in der anderen Relation als Werte des Schlüssels auf

Formalisierung Relationenmodell

- **Attribute und Domänen**
  - $\mathcal{U}$ nichtleere, endliche Menge: Universum
  - $A \in \mathcal{U}$: Attribut
  - $\mathcal{D} = \{D_1, \ldots, D_m\}$ Menge endlicher, nichtleerer Mengen: jedes D_i: Wertebereich oder Domäne
  - total definierte Funktion $\mathrm{dom} : \mathcal{U} \longrightarrow \mathcal{D}$
  - $\mathrm{dom}(A)$: Domäne von A
    $w \in \mathrm{dom}(A)$: Attributwert für A

Formalisierung Relationenmodell /2

- **Relationenschemata und Relationen**
  - $R \subseteq \mathcal{U}$: Relationenschema
  - Relation r über $R = \{A_1, \ldots, A_n\}$ (kurz: $r(R)$) ist endliche Menge von Abbildungen $t : R \longrightarrow \bigcup_{i=1}^{m} D_i$, Tupel genannt
  - Es gilt $t(A) \in \mathrm{dom}(A)$ ($t(A)$ Restriktion von t auf $A \in R$)
  - für $X \subseteq R$ analog $t(X)$ X-Wert von t
  - Menge aller Relationen über R: **REL**$(R) := \{r \mid r(R)\}$

Relation: Mathematische Definition vs. DB-Definition

- Mathematik: Definition einer Relation als Teilmenge des kartesischen Produkts der zugrundeliegenden Wertebereiche $r \subseteq \mathrm{dom}(A_1) \times \cdots \times \mathrm{dom}(A_n)$
- DB: problematisch, da die verschiedenen Spalten einer Tabelle dann in ihrer Reihenfolge fixiert
- $$r_1 \subseteq \mathrm{dom}(\texttt{Weingut}) \times \mathrm{dom}(\texttt{Anbaugebiet}) \times \mathrm{dom}(\texttt{Region})$$

  und

  $$r_2 \subseteq \times\, \mathrm{dom}(\texttt{Region}) \times \mathrm{dom}(\texttt{Anbaugebiet}) \times \mathrm{dom}(\texttt{Weingut})$$

  auf folgender Folie sind ungleich

Relation: Beispiel Mathematische Definition

r_1

Weingut	**Anbaugebiet**	**Region**
Creek	Barossa Valley	South Australia
Helena	Napa Valley	Kalifornien
Château La Rose	Saint-Emilion	Bordeaux

r_2

Region	**Anbaugebiet**	**Weingut**
South Australia	Barossa Valley	Creek
Kalifornien	Napa Valley	Helena
Bordeaux	Saint-Emilion	Château La Rose

DB-Definition (Relation als Menge von Abbildungen) dagegen reihenfolgeunabhängig (siehe folgende Folie)

Relation: Beispiel DB-Definition

Attributwerte werden den einzelnen Attributen nun reihenfolgenunabhängig zugewiesen: Sowohl r_1 als auch r_2 bestehen aus Tupeln t_1, t_2, t_3 mit

t_1(`Weingut`)='Creek', t_1(`Anbaugebiet`)='Barossa Valley',
t_1(`Region`)='South Australia'
t_2(`Weingut`)='Helena', t_2(`Anbaugebiet`)='Napa Valley',
t_2(`Region`)='Kalifornien'
t_3(`Weingut`)='Château La Rose', t_3(`Anbaugebiet`)='Saint-Emilion',
t_3(`Region`)='Bordeaux'

also identisch. Tupelschreibweisen, falls Reihenfolge der Attribute festgelegt:
$t_1 = \langle$'Creek', 'Barossa Valley', 'South Australia'$\rangle$
$t_1(\{\texttt{Weingut}, \texttt{Anbaugebiet}\}) = \langle$'Creek', 'Barossa Valley'$\rangle$
für $t_1(\{\texttt{Weingut}, \texttt{Anbaugebiet}\}) = \langle \texttt{Weingut} :$ 'Creek', $\texttt{Anbaugebiet} :$ 'Barossa Valley'$\rangle$

Formalisierung Relationenmodell /3

- **Datenbankschema und Datenbank**
  - Menge von Relationenschemata $S := \{R_1, \ldots, R_p\}$: Datenbankschema
  - Datenbank über S: Menge von Relationen $d := \{r_1, \ldots, r_p\}$, wobei $r_i(R_i)$
  - Datenbank d über S: $d(S)$
  - Relation $r \in d$: Basisrelation

Integritätsbedingungen

- Identifizierende Attributmenge $K := \{B_1, \ldots, B_k\} \subseteq R$:

$$\forall t_1, t_2 \in r \; [t_1 \neq t_2 \quad \Longrightarrow \quad \exists B \in K : t_1(B) \neq t_2(B)]$$

- Schlüssel: ist minimale identifizierende Attributmenge
  - {`Name, Jahrgang, Weingut`} und
  - {`WeinID`} für `WEINE`
- Primattribut: Element eines Schlüssels
- Primärschlüssel: ausgezeichneter Schlüssel
- Oberschlüssel oder **Superkey**: jede Obermenge eines Schlüssels (= identifizierende Attributmenge)
- Fremdschlüssel: $X(R_1) \rightarrow Y(R_2)$

$$\{t(X) | t \in r_1\} \subseteq \{t(Y) | t \in r_2\}$$

Formalisierung Relationenmodell /4

Lokale Integritätsbedingungen für ein Relationenschema R sind Abbildungen $b \in \mathcal{B}$

$$b\colon \{r \mid r(R)\} \to \{\texttt{true}, \texttt{false}\}$$

Relationenschemata mit lokalen Integritätsbedingungen: erweitertes Relationenschema

$$\mathcal{R} := (R, \mathcal{B})$$

Relation r über $\mathcal{R}$ (kurz: $r(\mathcal{R})$) muss lokalen Integritätsbedingungen über $\mathcal{B}$ genügen: r über R mit $b(r) = \texttt{true}$ für alle $b \in \mathcal{B}$ (kurz: $\mathcal{B}(r) = \texttt{true}$)

Menge aller Relationen über einem erweiterten Relationenschema $\mathcal{R}$ bezeichnet mit

$$\textbf{SAT}_R(\mathcal{B}) := \{r \mid r(\mathcal{R})\}$$

Formalisierung Relationenmodell /5

Lokal erweitertes Datenbankschema

$$S := \{\mathcal{R}_1, \ldots, \mathcal{R}_p\}$$

Eine Datenbank über lokal erweitertem Datenbankschema $S := \{\mathcal{R}_1, \ldots, \mathcal{R}_p\}$ ist Menge von Relationen $d := \{r_1, \ldots, r_p\}$, wobei $r_i(\mathcal{R}_i)$ für alle $i \in \{1, \ldots, p\}$ gilt
Datenbank d über S wird mit $d(S)$ bezeichnet, Relation $r \in d$ heißt Basisrelation

Formalisierung Relationenmodell /6

$$\Gamma := \{\gamma \mid \gamma : \{d \mid d(S)\} \longrightarrow \{\mathbf{true}, \mathbf{false}\}\}$$

nennt man Menge globaler Integritätsbedingungen für S

$$\mathcal{S} := (S, \Gamma)$$

global erweitertes Datenbankschema

$d(\mathcal{S})$ ist eine Datenbank $d(S)$ mit $\gamma(d) = \mathbf{true}$ für alle $\gamma \in \Gamma$ (kurz: $\Gamma(d) = \mathbf{true}$).

Die Menge aller gültigen Datenbanken (bezüglich der vorliegenden Integritätsbedingungen) wird mit

$$\mathbf{DAT}(\mathcal{S}) := \{d \mid d(\mathcal{S})\}$$

definiert. Ein Fremdschlüssel ist eine spezielle globale Integritätsbedingung.

Schreibweisen im Relationenmodell

$X = \{A, B, C\} = ABC$
X, Y, Z: Menge von Attributen
A, B, C: einzelne Attribute
K: ein Schlüssel (Menge von Attributen)
$\mathcal{K}$: eine Menge von Schlüsseln (Menge von Attributmengen)
etwa $\mathcal{K} = \{\{A, B\}, \{B, C\}\}$ oder $\{AB, BC\}$
$X \cup Y$ auch kurz XY
$X \cup \{A\}$ auch kurz XA

Relationaler DB-Entwurf: Überblick

- Verfeinern des logischen Entwurfs
- Ziel: Vermeidung von Redundanzen durch Aufspalten von Relationenschemata, ohne gleichzeitig
  - semantische Informationen zu verlieren (Abhängigkeitstreue)
  - die Möglichkeit zur Rekonstruktion der Relationen zu verlieren (Verbundtreue)
- Redundanzvermeidung durch Normalformen (s.u.)

Relation mit Redundanzen

WEINE

WeinID	Name	...	Weingut	Anbaugebiet	Region
1042	La Rose Gr. 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 Res.	...	Müller	Rheingau	Hessen
4961	Chardonnay	...	Bighorn	Napa Valley	Kalifornien

Redundanzen

- Redundanzen in Basisrelationen aus mehreren Gründen unerwünscht:
  - Redundante Informationen belegen unnötigen Speicherplatz
  - Änderungsoperationen auf Basisrelationen mit Redundanzen nur schwer korrekt umsetzbar: wenn eine Information redundant vorkommt, muss eine Änderung diese Information in allen ihren Vorkommen verändern
    - mit normalen relationalen Änderungsoperationen und den in relationalen Systemen vorkommenden lokalen Integritätsbedingungen (Schlüsseln) nur schwer realisierbar

Änderungsanomalien

- Einfügen in die mit Redundanzen behaftete `WEINE`-Relation:

```
insert into WEINE (WeinID, Name, Farbe, Jahrgang,
      Weingut, Anbaugebiet, Region)
   values (4711, 'Chardonnay', 'Weiß', 2004,
      'Helena', 'Rheingau', 'Kalifornien')
```

  - WeinID 4711 bereits anderem Wein zugeordnet: verletzt FD $\texttt{WeinID} \rightarrow \texttt{Name}$
  - Weingut Helena war bisher im Napa Valley angesiedelt: verletzt FD $\texttt{Weingut} \rightarrow \texttt{Anbaugebiet}$
  - Rheingau liegt nicht in Kalifornien: verletzt FD $\texttt{Anbaugebiet} \rightarrow \texttt{Region}$

- auch **`update`**- und **`delete`**-Anomalien

Funktionale Abhängigkeiten

- funktionale Abhängigkeit zwischen Attributmengen X und Y einer Relation

Wenn in jedem Tupel der Relation der Attributwert unter den X-Komponenten den Attributwert unter den Y-Komponenten festlegt.

- Unterscheiden sich zwei Tupel in den X-Attributen nicht, so haben sie auch gleiche Werte für alle Y-Attribute
- Notation für funktionale Abhängigkeit (FD, von functional dependency): $X \rightarrow Y$
- Beispiel:

  `WeinID` $\rightarrow$ `Name, Weingut`
  `Anbaugebiet` $\rightarrow$ `Region`
- aber nicht: `Weingut` $\rightarrow$ `Name`

Schlüssel als Spezialfall

- für Beispiel auf Folie 5-18

  $\texttt{WeinID} \rightarrow \texttt{Name}, \texttt{Farbe}, \texttt{Jahrgang}, \texttt{Weingut}, \texttt{Anbaugebiet}, \texttt{Region}$

- Immer: $\texttt{WeinID} \rightarrow \texttt{WeinID}$,
  dann gesamtes Schema auf rechter Seite
- Wenn linke Seite minimal: Schlüssel
- Formal: Schlüssel X liegt vor, wenn für Relationenschema R FD $X \rightarrow R$ gilt und X minimal

Ziel des Datenbankentwurfs: alle gegebenen funktionalen Abhängigkeiten in „Schlüsselabhängigkeiten“ umformen, ohne dabei semantische Information zu verlieren

Ableitung von FDs

r	A	B	C
	a_1	b_1	c_1
	a_2	b_1	c_1
	a_3	b_2	c_1
	a_4	b_1	c_1

- genügt $A \rightarrow B$ und $B \rightarrow C$
- dann gilt auch $A \rightarrow C$
- nicht ableitbar $C \rightarrow A$ oder $C \rightarrow B$

Ableitung von FDs /2

- Gilt für f über R $\mathbf{SAT}_R(F) \subseteq \mathbf{SAT}_R(f)$, dann impliziert F die FD f (kurz: $F \models f$)
- obiges Beispiel:

$$F = \{A \rightarrow B, B \rightarrow C\} \models A \rightarrow C$$

- Hüllenbildung: Ermittlung aller funktionalen Abhängigkeiten, die aus einer gegebenen FD-Menge abgeleitet werden können
- Hülle $F_R^+ := \{f \mid (f \text{ FD über } R) \wedge F \models f\}$
- Beispiel:

$$\begin{aligned}\{A \rightarrow B, B \rightarrow C\}^+ &= \{A \rightarrow B, B \rightarrow C, A \rightarrow C, AB \rightarrow C, A \rightarrow BC, \ldots, \\ &\quad AB \rightarrow AB, \ldots\}\end{aligned}$$

Ableitungsregeln

F1	Reflexivität	$X \supseteq Y \implies X \rightarrow Y$
F2	Augmentation	$\{X \rightarrow Y\} \implies XZ \rightarrow YZ$ sowie $XZ \rightarrow Y$
F3	Transitivität	$\{X \rightarrow Y, Y \rightarrow Z\} \implies X \rightarrow Z$
F4	Dekomposition	$\{X \rightarrow YZ\} \implies X \rightarrow Y$
F5	Vereinigung	$\{X \rightarrow Y, X \rightarrow Z\} \implies X \rightarrow YZ$
F6	Pseudotransitivität	$\{X \rightarrow Y, WY \rightarrow Z\} \implies WX \rightarrow Z$

F1-F3 bekannt als Armstrong-Axiome (sound, complete)

- *gültig* (sound): Regeln leiten keine FDs ab, die logisch nicht impliziert
- *vollständig* (complete): alle implizierten FDs werden abgeleitet
- *unabhängig* (independent) oder auch bzgl. $\subseteq$ minimal: keine Regel kann weggelassen werden

Beweis: F1

- Annahme: $X \supseteq Y$, $X, Y \subset R$, $t_1, t_2 \in r(R)$ mit $t_1(X) = t_2(X)$
- dann folgt: $t_1(Y) = t_2(Y)$ wegen $X \supseteq Y$
- daraus folgt: $X \rightarrow Y$

Beweis: F2

- Annahme: $X \rightarrow Y$ gilt in $r(R)$, jedoch nicht: $XZ \rightarrow YZ$
- dann müssen zwei Tupel $t_1, t_2 \in r(R)$ existieren, so dass gilt
  (1) $t_1(X) = t_2(X)$
  (2) $t_1(Y) = t_2(Y)$
  (3) $t_1(XZ) = t_2(XZ)$
  (4) $t_1(YZ) \neq t_2(YZ)$
- Widerspruch wegen $t_1(Z) = t_2(Z)$ aus (1) und (3), woraus folgt: $t_1(YZ) = t_2(YZ)$ (in Verbindung mit (4))

Beweis: F3

- Annahme: in $r(R)$ gelten:
  (1) $X \rightarrow Y$
  (2) $Y \rightarrow Z$
- demzufolge für zwei beliebige Tupel $t_1, t_2 \in r(R)$ mit $t_1(X) = t_2(X)$ muss gelten:
  (3) $t_1(Y) = t_2(Y)$ (wegen (1))
  (4) $t_1(Z) = t_2(Z)$ (wegen (3) und (2))
- daher gilt: $X \rightarrow Z$

Alternative Regelmenge

- B-Axiome oder RAP-Regeln

R Reflexivität	$\{\} \implies X \rightarrow X$	
A Akkumulation	$\{X \rightarrow YZ, Z \rightarrow AW\} \implies X \rightarrow YZA$	
P Projektivität	$\{X \rightarrow YZ\} \implies X \rightarrow Y$	

- RAP-Regelmenge ist vollständig, da Armstrong-Axiome daraus abgeleitet werden können
- RAP-Regeln sind auch gültig und unabhängig

Membership-Problem

Kann eine bestimmte FD $X \rightarrow Y$ aus der vorgegebenen Menge F abgeleitet werden, d.h. wird sie von F impliziert?

Membership-Problem: „$X \rightarrow Y \in F^+$?“

- Hülle einer Attributmenge X bzgl. F ist $X_F^+ := \{A \mid X \rightarrow A \in F^+\}$
- Membership-Problem kann durch das modifizierte Problem

  Membership-Problem (2): „$Y \subseteq X_F^+$?“

  in linearer Zeit gelöst werden

Algorithmus CLOSURE

- Ermittlung von X_F^+: die Hülle von X bzgl. F

CLOSURE(F, X):
  $X^+ := X$
  repeat
    $\overline{X}^+ := X^+$ /* **R**-*Regel* */
    forall FDs $Y \rightarrow Z \in F$
      if $Y \subseteq X^+$ **then** $X^+ := X^+ \cup Z$ /* **A**-*Regel* */
  until $X^+ = \overline{X}^+$
  return X^+

MEMBER$(F, X \rightarrow Y)$: /* *Test auf* $X \rightarrow Y \in F^+$ */
  return $Y \subseteq$ **CLOSURE**(F, X) /* **P**-*Regel* */

- Beispiel: $A \rightarrow C \in \{\underbrace{A \rightarrow B}_{f_1}, \underbrace{B \rightarrow C}_{f_2}\}^+$?

Überdeckungen

- F heißt äquivalent zu G
- oder: F Überdeckung von G; kurz: $F \equiv G$ falls $F^+ = G^+$
- d.h.:

$$\forall g \in G : g \in F^+ \ \wedge \ \forall f \in F : f \in G^+$$

- wichtige Entwurfsaufgabe: Finden einer Überdeckung, die
  - einerseits so wenig Attribute wie möglich in ihren funktionalen Abhängigkeiten und
  - andererseits möglichst wenig funktionale Abhängigkeiten insgesamt enthält
- verschiedene Formen von Überdeckung: nichtredundant, reduziert, minimal, ringförmig

Schemaeigenschaften

- Relationenschemata, Schlüssel und Fremdschlüssel so wählen, dass
  1. alle Anwendungsdaten aus den Basisrelationen hergeleitet werden können,
  2. nur semantisch sinnvolle und konsistente Anwendungsdaten dargestellt werden können und
  3. die Anwendungsdaten möglichst nicht-redundant dargestellt werden.
- Hier: Forderung 3
  - Redundanzen innerhalb einer Relation: Normalformen
  - globale Redundanzen: Minimalität

Normalformen

- legen Eigenschaften von Relationenschemata fest
- verbieten bestimmte Kombinationen von funktionalen Abhängigkeiten in Relationen
- sollen Redundanzen und Anomalien vermeiden

Erste Normalform

- erlaubt nur atomare Attribute in den Relationenschemata, d.h. als Attributwerte sind Elemente von Standard-Datentypen wie **`integer`** oder **`string`** erlaubt, aber keine Konstruktoren wie **`array`** oder **`set`**
- Nicht in 1NF:

Weingut	**Anbaugebiet**	**Region**	**WName**
Ch. La Rose	Saint-Emilion	Bordeaux	La Rose Grand Cru
Creek	Barossa Valley	South Australia	Creek Shiraz, Pinot Noir
Helena	Napa Valley	Kalifornien	Zinfandel, Pinot Noir
Müller	Rheingau	Hessen	Riesling Reserve
Bighorn	Napa Valley	Kalifornien	Chardonnay

Erste Normalform /2

- in erster Normalform:

Weingut	Anbaugebiet	Region	WName
Ch. La Rose	Saint-Emilion	Bordeaux	La Rose Grand Cru
Creek	Barossa Valley	South Australia	Creek Shiraz
Creek	Barossa Valley	South Australia	Pinot Noir
Helena	Napa Valley	Kalifornien	Zinfandel
Helena	Napa Valley	Kalifornien	Pinot Noir
Müller	Rheingau	Hessen	Riesling Reserve
Bighorn	Napa Valley	Kalifornien	Chardonnay

Zweite Normalform

- partielle Abhängigkeit liegt vor, wenn ein Attribut funktional schon von einem Teil des Schlüssels abhängt

Name	Weingut	Farbe	Anbaugebiet	Region	Preis
La Rose Grand Cru	Ch. La Rose	Rot	Saint-Emilion	Bordeaux	39.00
Creek Shiraz	Creek	Rot	Barossa Valley	South Australia	7.99
Pinot Noir	Creek	Rot	Barossa Valley	South Australia	10.99
Zinfandel	Helena	Rot	Napa Valley	Kalifornien	5.99
Pinot Noir	Helena	Rot	Napa Valley	Kalifornien	19.99
Riesling Reserve	Müller	Weiß	Rheingau	Hessen	14.99
Chardonnay	Bighorn	Weiß	Napa Valley	Kalifornien	9.90

f_1: `Name, Weingut` $\rightarrow$ `Preis`

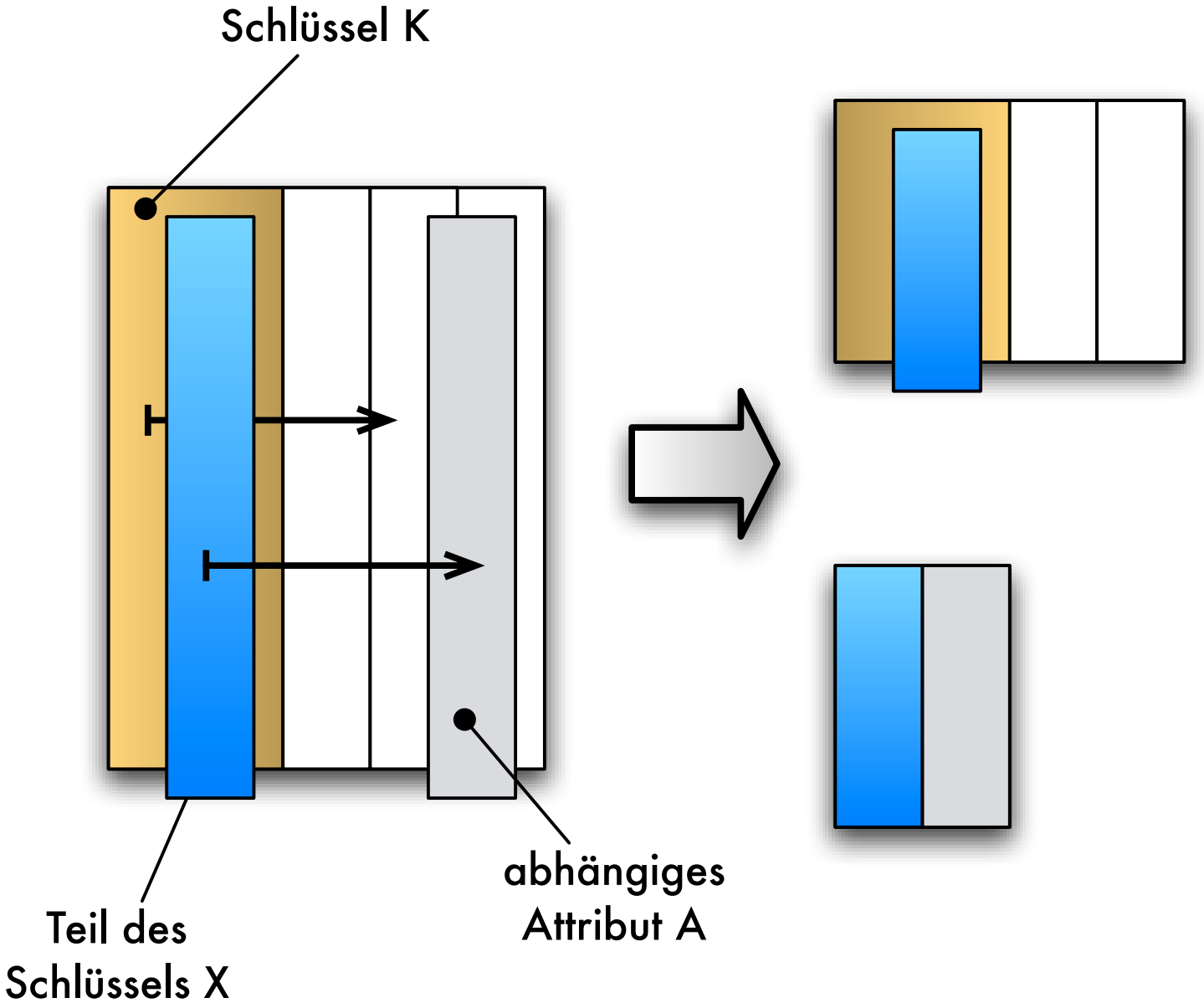
f_2: `Name` $\rightarrow$ `Farbe`

f_3: `Weingut` $\rightarrow$ `Anbaugebiet, Region`

f_4: `Anbaugebiet` $\rightarrow$ `Region`

- Zweite Normalform eliminiert derartige partielle Abhängigkeiten bei Nichtschlüsselattributen

Eliminierung partieller Abhängigkeiten



Zweite Normalform /2

- Beispielrelation in 2NF

  R1(Name, Weingut, Preis)
  R2(Name, Farbe)
  R3(Weingut, Anbaugebiet, Region)

Zweite Normalform /3

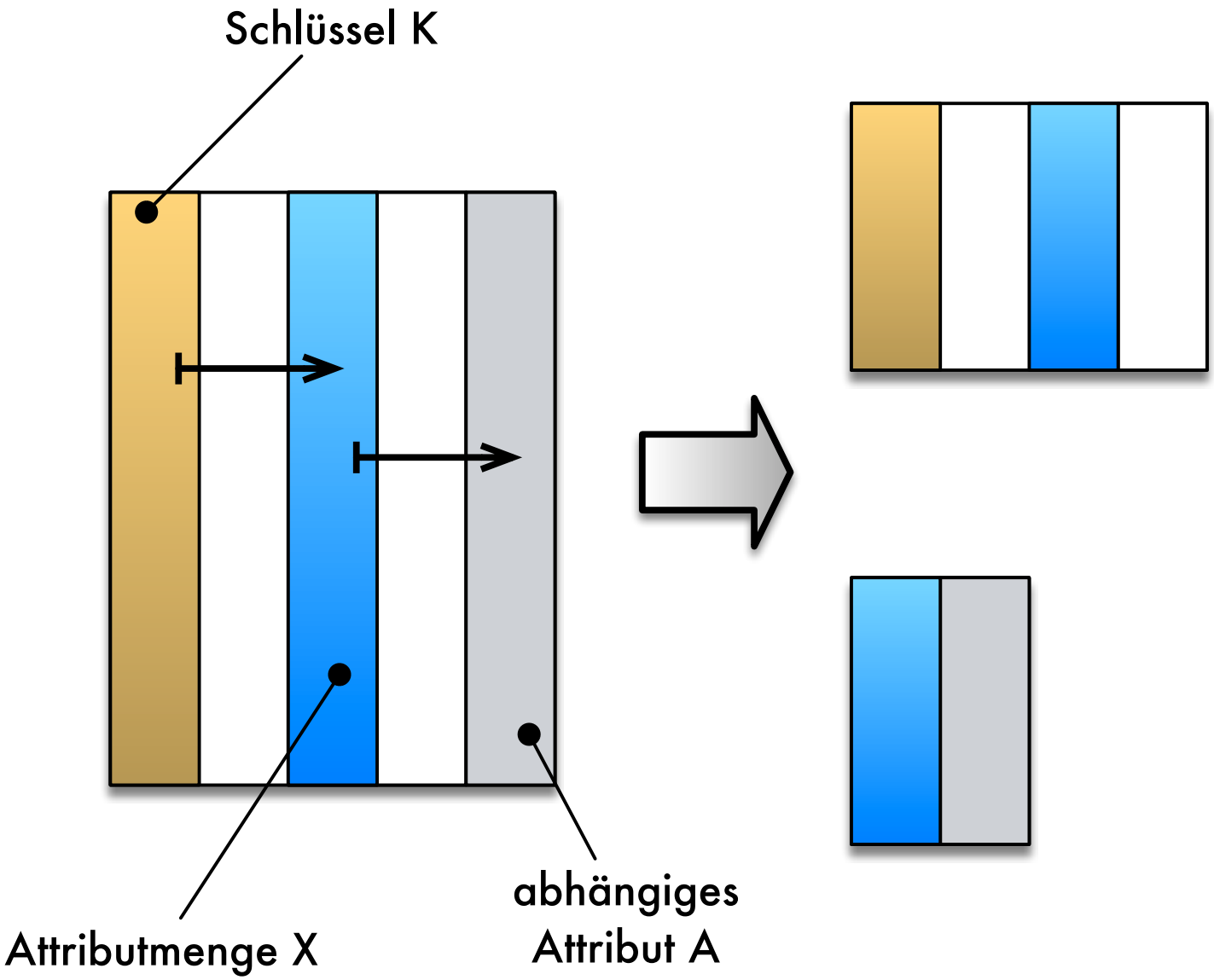
- Hinweis: partiell abhängiges Attribut stört nur, wenn es kein Primattribut ist
- 2NF formal: erweitertes Relationenschema $\mathcal{R} = (R, \mathcal{K})$, FD-Menge F über R

- Y **hängt partiell** von X bzgl. F ab, wenn die FD $X \rightarrow Y$ nicht linksreduziert ist
- Y **hängt voll** von X ab, wenn die FD $X \rightarrow Y$ linksreduziert ist
- $\mathcal{R}$ ist in 2NF, wenn $\mathcal{R}$ in 1NF ist und jedes Nicht-Primattribut von R voll von jedem Schlüssel von $\mathcal{R}$ abhängt

Dritte Normalform

- eliminiert (zusätzlich) transitive Abhängigkeiten
- etwa `Weingut` $\rightarrow$ `Anbaugebiet` und `Anbaugebiet` $\rightarrow$ `Region` in Relation auf Folie 5-42
- man beachte: 3NF betrachtet nur Nicht-Schlüsselattribute als Endpunkt transitiver Abhängigkeiten

Eliminierung transitiver Abhängigkeiten



Dritte Normalform /2

- transitive Abhängigkeit in `R3`, d.h. `R3` verletzt 3NF
- Beispielrelation in 3NF

```
R3_1(Weingut, Anbaugebiet)
R3_2(Anbaugebiet, Region)
```

Dritte Normalform: formal

- Relationenschema R, $X \subseteq R$ und F ist eine FD-Menge über R

- $A \in R$ heißt transitiv abhängig von X bezüglich F genau dann, wenn es ein $Y \subseteq R$ gibt mit $X \rightarrow Y, Y \not\rightarrow X, Y \rightarrow A, A \notin XY$
- erweitertes Relationenschema $\mathcal{R} = (R, \mathcal{K})$ ist in 3NF bezüglich F genau dann, wenn

$$\nexists A \in R: \quad A \text{ ist Nicht-Primattribut in } R \wedge A \text{ transitiv abhängig von einem } K \in \mathcal{K} \text{ bezüglich } F.$$

Boyce-Codd-Normalform

- Verschärfung der 3NF: Eliminierung transitiver Abhängigkeiten auch zwischen Primattributen

Name	Weingut	Händler	Preis
La Rose Grand Cru	Château La Rose	Weinkontor	39.90
Creek Shiraz	Creek	Wein.de	7.99
Pinot Noir	Creek	Wein.de	10.99
Zinfandel	Helena	GreatWines.com	5.99
Pinot Noir	Helena	GreatWines.com	19.99
Riesling Reserve	Müller	Weinkeller	19.99
Chardonnay	Bighorn	Wein-Dealer	9.90

- FDs:

$$\begin{aligned} \texttt{Name}, \texttt{Weingut} &\rightarrow \texttt{Preis} \\ \texttt{Weingut} &\rightarrow \texttt{Händler} \\ \texttt{Händler} &\rightarrow \texttt{Weingut} \end{aligned}$$

- Schlüsselkandidaten: { Name, Weingut } und { Name, Händler }
- in 3NF, nicht jedoch in BCNF

Boyce-Codd-Normalform /2

- erweitertes Relationenschema $\mathcal{R} = (R, \mathcal{K})$, FD-Menge F
- BCNF formal:

$\nexists A \in R : A$ transitiv abhängig von einem $K \in \mathcal{K}$ bezüglich F.

- Schema in BCNF:

  WEINE(<u>Name, Weingut</u>, Preis)
  WEINHANDEL(<u>Weingut</u>, Händler)

- BCNF kann jedoch Abhängigkeitstreue verletzen, daher oft nur bis 3NF

Minimalität

- Global Redundanzen vermeiden
- andere Kriterien (wie Normalformen) mit möglichst wenig Schemata erreichen
- Beispiel: Attributmenge ABC, FD-Menge $\{A \rightarrow B, B \rightarrow C\}$
- Datenbankschemata in dritter Normalform:

$$S = \{(AB, \{A\}), (BC, \{B\})\}$$

$$S' = \{(AB, \{A\}), (BC, \{B\}), (AC, \{A\})\}$$

Redundanzen in S'

Schemaeigenschaften

Kennung	Schemaeigenschaft	Kurzcharakteristik
	1NF	nur atomare Attribute
	2NF	keine partielle Abhängigkeit eines Nicht-Primattributes von einem Schlüssel
S 1	3NF	keine transitive Abhängigkeit eines Nicht-Primattributes von einem Schlüssel
	BCNF	keine transitive Abhängigkeit eines Attributes von einem Schlüssel
S 2	Minimalität	minimale Anzahl von Relationenschemata, die die anderen Eigenschaften erfüllt

Transformationseigenschaften

- Bei einer Zerlegung einer Relation in mehrere Relationen ist darauf zu achten, dass
  1. nur semantisch sinnvolle und konsistente Anwendungsdaten dargestellt (Abhängigkeitstreue) und
  2. alle Anwendungsdaten aus den Basisrelationen hergeleitet werden können (Verbundtreue)

Abhängigkeitstreue

- Abhängigkeitstreue: eine Menge von Abhängigkeiten kann äquivalent in eine zweite Menge von Abhängigkeiten transformiert werden
- spezieller: in die Menge der Schlüsselabhängigkeiten, da diese vom Datenbanksystem effizient überprüft werden kann
  - die Menge der Abhängigkeiten soll äquivalent zu der Menge der Schlüsselbedingungen im resultierenden Datenbankschema sein
  - Äquivalenz sichert zu, dass mit den Schlüsselabhängigkeiten semantisch genau die gleichen Integritätsbedingungen ausgedrückt werden wie mit den funktionalen oder anderen Abhängigkeiten vorher

Abhängigkeitstreue: Beispiel

- Zerlegung des Relationenschemas WEINE (Folie 5-43) in 3NF:

  R1(<u>Name, Weingut</u>, Preis)
  R2(<u>Name</u>, Farbe)
  R3_1(<u>Weingut</u>, Anbaugebiet)
  R3_2(<u>Anbaugebiet</u>, Region)

  mit Schlüsselabhängigkeiten

  $$\begin{aligned} \texttt{Name}, \texttt{Weingut} &\rightarrow \texttt{Preis} \\ \texttt{Name} &\rightarrow \texttt{Farbe} \\ \texttt{Weingut} &\rightarrow \texttt{Anbaugebiet} \\ \texttt{Anbaugebiet} &\rightarrow \texttt{Region} \end{aligned}$$

- äquivalent zu FDs $f_1 \ldots f_4$ (Folie 5-43) $\rightsquigarrow$ abhängigkeitstreu

Abhängigkeitstreue: Beispiel /2

- Postleitzahl-Struktur der Deutschen Post

  `PLZ (P), Ort (O), Strasse(S), Hausnummer(H)`

  und funktionalen Abhängigkeiten F

  $$\mathrm{OSH} \rightarrow \mathrm{P},\ \mathrm{P} \rightarrow \mathrm{O}$$

- für ein Datenbankschema S bestehend aus dem einzigen Relationenschema

  $$(\mathrm{OSHP},\ \{\mathrm{OSH}, \mathrm{PSH}\}),$$

  ist Menge der Schlüsselabhängigkeiten

  $$\{\ \mathrm{OSH} \rightarrow \mathrm{OSHP},\ \mathrm{PSH} \rightarrow \mathrm{OSHP}\ \}$$

  nicht äquivalent zu F und somit S nicht abhängigkeitstreu

Adressdatenbank gutes Beispiel für entweder BCNF oder abhängigkeitstreu

Abhängigkeitstreue formal

- lokal erweitertes Datenbankschema $S = \{(R_1, \mathcal{K}_1), \ldots, (R_p, \mathcal{K}_p)\}$; ein Menge F lokaler Abhängigkeiten

S charakterisiert vollständig F (oder: ist abhängigkeitstreu bezüglich F) genau dann, wenn

$$F \equiv \{K \rightarrow R \mid (R, \mathcal{K}) \in S, K \in \mathcal{K}\}$$

Verbundtreue

- zur Erfüllung des Kriteriums der Normalformen müssen Relationenschemata teilweise in kleinere Relationenschemata zerlegt werden
- für Beschränkung auf „sinnvolle“ Zerlegungen gilt Forderung, dass die Originalrelation wieder aus den zerlegten Relationen mit dem natürlichen Verbund zurückgewonnen werden kann (engl.: lossless join)
  $\rightsquigarrow$ Verbundtreue

nicht verwechseln: wenn kein lossless join vorliegt, verliert man keine Tupel, sondern es entstehen weitere unsinnige
nicht verwechseln: lossless join $\neq$ outer join mit dangling Tupeln

Verbundtreue: Beispiele

- Zerlegung des Relationenschemas $R = ABC$ in

$$R_1 = AB \text{ und } R_2 = BC$$

- Dekomposition bei Vorliegen der Abhängigkeiten

$$F = \{A \rightarrow B, C \rightarrow B\}$$

  ist nicht verbundtreu
- dagegen bei Vorliegen von

$$F' = \{A \rightarrow B, B \rightarrow C\}$$

  verbundtreu

Verbundtreue Dekomposition

- Originalrelation:

A	B	C
1	2	3
4	2	3

- Dekomposition:

A	B
1	2
4	2

B	C
2	3

- Verbund (verbundtreu):

A	B	C
1	2	3
4	2	3

Nicht verbundtreue Dekomposition

- Originalrelation:

A	B	C
1	2	3
4	2	5

- Dekomposition:

A	B
1	2
4	2

B	C
2	3
2	5

- Verbund (nicht verbundtreu):

A	B	C
1	2	3
4	2	5
1	2	5
4	2	3

Verbundtreue formal

Die Dekomposition einer Attributmenge X in $X_1, \ldots, X_p$ mit $X = \bigcup_{i=1}^{p} X_i$ heißt verbundtreu ($\pi \bowtie$-treu, lossless) bezüglich einer Menge von Abhängigkeiten F über X genau dann, wenn

$$\forall r \in \mathbf{SAT}_X(F) : \pi_{X_1}(r) \bowtie \cdots \bowtie \pi_{X_p}(r) = r$$

gilt.

- einfaches Kriterium für Verbundtreue bei Dekomposition in zwei Relationenschemata: Dekomposition von X in X_1 und X_2 ist verbundtreu bzgl. F, wenn $X_1 \cap X_2 \rightarrow X_1 \in F^+$ oder $X_1 \cap X_2 \rightarrow X_2 \in F^+$

Verbundtreue: allgemeines Kriterium

G Menge von Schlüsselabhängigkeiten
Dann gilt für eine abhängigkeitstreue Zerlegung

$$\exists i \in \{1, \ldots, p\} : X_i \rightarrow X \in G^+ \quad \Longrightarrow \quad \text{Dekomposition von } X \text{ in } X_1, \ldots, X_p \text{ ist verbundtreu bezüglich } G \tag{1}$$

minimale Teilmenge von X_i Universalschlüssel

- Universalschlüssel muss also in einer Menge der Partition X_i (also einem Relationenschema) vollständig enthalten sein
- Kriterium nur sinnvoll anzuwenden, wenn abhängigkeitstreue Zerlegung vorliegt: dann Schlüsselabhängigkeiten G im obigen Kriterium äquivalent zur Ausgangs-FD-Menge F

Transformationseigenschaften

Kennung	Transformationseigenschaft	Kurzcharakteristik
T1	Abhängigkeitstreue	alle gegebenen Abhängigkeiten sind durch Schlüssel repräsentiert
T2	Verbundtreue	Originalrelationen können durch den Verbund der Basisrelationen wiedergewonnen werden

Entwurfsverfahren: Ziele

- Universum $\mathcal{U}$ und FD-Menge F gegeben
- lokal erweitertes Datenbankschema $S = \{(R_1, \mathcal{K}_1), \ldots, (R_p, \mathcal{K}_p)\}$ berechnen mit
  - $\boxed{\text{T}}\boxed{1}$: S charakterisiert vollständig F
  - $\boxed{\text{S}}\boxed{1}$: S ist in 3NF bezüglich F
  - $\boxed{\text{T}}\boxed{2}$: Dekomposition von $\mathcal{U}$ in $R_1, \ldots, R_p$ ist verbundtreu bezüglich F
  - $\boxed{\text{S}}\boxed{2}$: Minimalität, d.h.
    $\nexists S' : S'$ erfüllt $\boxed{\text{T}}\boxed{1}$, $\boxed{\text{S}}\boxed{1}$, $\boxed{\text{T}}\boxed{2}$ und $|S'| < |S|$

Entwurfsverfahren: Beispiel

- Datenbankschemata schlecht entworfen, wenn nur eins dieser vier Kriterien nicht erfüllt
- Beispiel: $S = \{(AB, \{A\}), (BC, \{B\}), (AC, \{A\})\}$ erfüllt $\boxed{\text{T}}\boxed{1}$, $\boxed{\text{S}}\boxed{1}$ und $\boxed{\text{T}}\boxed{2}$ bezüglich $F = \{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$
  in dritter Relation AC-Tupel redundant oder inkonsistent
- korrekt: $S' = \{(AB, \{A\}), (BC, \{B\})\}$

Dekomposition

- Geg.: initiales Universalrelationenschema $\mathcal{R} = (\mathcal{U}, \mathcal{K}(F))$ mit allen Attributen und einer von erfassten FDs F über R implizierten Schlüsselmenge
  - Attributmenge $\mathcal{U}$ und eine FD-Menge F
  - suche alle $K \rightarrow \mathcal{U}$ mit K minimal, für die $K \rightarrow \mathcal{U} \in F^+$ gilt ($\mathcal{K}(F)$)
- Ges.: Zerlegung in $D = \{\mathcal{R}_1, \mathcal{R}_2, \ldots\}$ von 3NF-Relationenschemata

Dekomposition: Algorithmus

DECOMPOSE($\mathcal{R}$)

Setze $D := \{\mathcal{R}\}$

while $\mathcal{R}' \in D$, das 3NF nicht erfüllt

/ Finde Attribut A, das transitiv von K abhängig ist */*

if Schlüssel K mit $K \rightarrow Y, Y \not\rightarrow K, Y \rightarrow A, A \notin KY$

then

/ Zerlege Relationenschema R bzgl. A */*

$R_1 := R - A$, $R_2 := YA$

$\mathcal{R}_1 := (R_1, \mathcal{K})$, $\mathcal{R}_2 := (R_2, \mathcal{K}_2 = \{Y\})$

$D := (D - \mathcal{R}') \cup \{\mathcal{R}_1\} \cup \{\mathcal{R}_2\}$

end if

end while

return D

Dekomposition: Beispiel

- initiales Relationenschema $R = ABC$
- funktionale Abhängigkeiten $F = \{A \rightarrow B, B \rightarrow C\}$
- Schlüssel $K = A$

(weiter an der Tafel)

Dekomposition: Beispiel /2

- initiales Relationenschema R mit `Name`, `Weingut`, `Preis`, `Farbe`, `Anbaugebiet`, `Region`
- funktionale Abhängigkeiten

f_1: `Name`, `Weingut` $\rightarrow$ `Preis`
f_2: `Name`, `Weingut` $\rightarrow$ `Weingut`
f_3: `Name`, `Weingut` $\rightarrow$ `Name`
f_4: `Name` $\rightarrow$ `Farbe`
f_5: `Weingut` $\rightarrow$ `Anbaugebiet`, `Region`
f_6: `Anbaugebiet` $\rightarrow$ `Region`

(weiter an der Tafel)

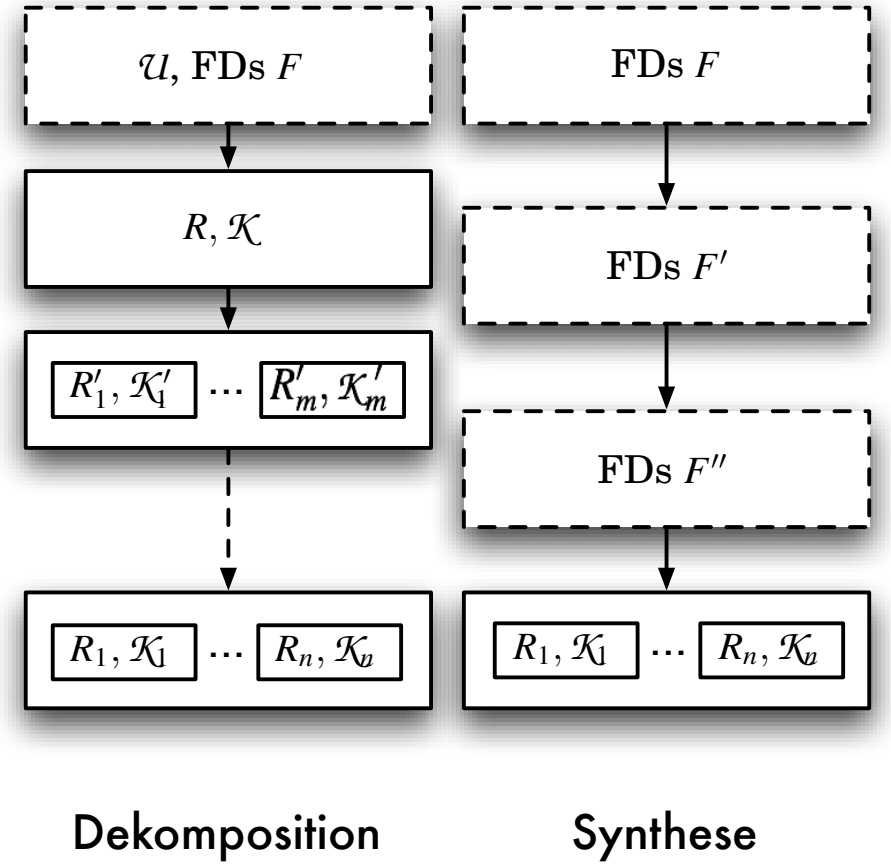
Dekomposition: Bewertung

- Vorteile: 3NF, Verbundtreue
- Nachteile: restliche Kriterien nicht, reihenfolgeabhängig, NP-vollständig (Schlüsselsuche)

Syntheseverfahren

- Prinzip: Synthese formt Original-FD-Menge F in resultierende Menge von Schlüsselabhängigkeiten G so um, dass $F \equiv G$ gilt
- Berechnung einer nichtredundanten, reduzierten, schließlich minimalen, ringförmigen Überdeckung
- „Abhängigkeitstreue“ im Verfahren verankert
- 3NF und Minimalität wird auch erreicht, reihenfolgeunabhängig
- Zeitkomplexität: quadratisch

Vergleich Dekomposition — Synthese



Wdh.: FD-Überdeckungen

- F heißt äquivalent zu G
- oder: F Überdeckung von G; kurz: $F \equiv G$ falls $F^+ = G^+$
- d.h.:

$$\forall g \in G : g \in F^+ \ \wedge \ \forall f \in F : f \in G^+$$

- wichtige Entwurfsaufgabe: Finden einer Überdeckung, die
  - einerseits so wenig Attribute wie möglich in ihren funktionalen Abhängigkeiten und
  - andererseits möglichst wenig funktionale Abhängigkeiten insgesamt enthält
- verschiedene Formen von Überdeckung: nichtredundant, reduziert, minimal, ringförmig

Nichtredundante Überdeckung

F nichtredundant, wenn

$$\nexists F' \subset F : F' \equiv F \iff \nexists f \in F : F \setminus \{f\} \models F$$

Es können also in F keine funktionalen Abhängigkeiten weggelassen werden, ohne Hülle von F zu verändern

```
NONREDUNDANTCOVER(F):
    var G: FD-Menge
    G := F
    forall FD f ∈ F
        if MEMBER(G − f, f)  /* f redundant in G ? */
            then G := G − f
    end for
    return G
```

Nichtredundante Überdeckung: Beispiel

Gegeben $F = \{A \rightarrow B, B \rightarrow A, B \rightarrow C, A \rightarrow C\}$

$$\textsc{NonredundantCover}(F) = \{A \rightarrow B, B \rightarrow A, A \rightarrow C\}$$

oder

$$\textsc{NonredundantCover}(F) = \{A \rightarrow B, B \rightarrow A, B \rightarrow C\}$$

- Nichtredundante Überdeckung einer FD-Menge ist nicht eindeutig, da Reihenfolge der FDs Einfluss auf Ergebnis hat
- weitere nichtredundante Überdeckung $\{A \rightarrow B, B \rightarrow A, AB \rightarrow C\}$ kann mithilfe des Algorithmus gar nicht erzeugt werden

Reduzierte Überdeckung

- Ziel: Entfernen überflüssiger Attribute auf linker bzw. rechter Seite von FDs
- Linksreduktion: entfernt unwesentliche Attribute auf der linken Seite einer FD entfernt
- Rechtsreduktion: entsprechend auf der rechten Seite
- erw. Relationenschema $\mathcal{R} = (R, \mathcal{K})$, FD-Menge F über R, A ist ein Attribut aus R und $X \rightarrow Y$ eine FD aus F

A heißt unwesentlich in $X \rightarrow Y$ bzgl. F, wenn

$$[X = AZ, Z \neq X \quad \Longrightarrow \quad (F - \{X \rightarrow Y\}) \cup \{Z \rightarrow Y\} \equiv F] \quad \vee$$

$$[Y = AW, W \neq Y \quad \Longrightarrow \quad (F - \{X \rightarrow Y\}) \cup \{X \rightarrow W\} \equiv F]$$

Reduzierte Überdeckung /2

- A kann also aus der FD $X \rightarrow Y$ entfernt werden, ohne dass sich die Hülle von F ändert
- FD $X \rightarrow Y$ heißt linksreduziert, wenn kein Attribut in X unwesentlich ist.
- FD $X \rightarrow Y$ heißt rechtsreduziert, wenn kein Attribut in Y unwesentlich ist.

Berechnung Reduzierte Überdeckung

REDUCEDCOVER(F):

forall FD $X \rightarrow Y \in F$ /* *Linksreduktion* */

forall $A \in X$ /* *A unwesentlich ?* */

if $Y \subseteq$ **CLOSURE**$(F, X - \{A\})$

then ersetze $X \rightarrow Y$ durch $(X - A) \rightarrow Y$ in F

end for

end for

forall verbleibende FD $X \rightarrow Y \in F$ /* *Rechtsreduktion* */

forall $B \in Y$ /* *B unwesentlich ?* */

if $B \subseteq$ **CLOSURE**$(F - \{X \rightarrow Y\} \cup \{X \rightarrow (Y - B)\}, X)$

then ersetze $X \rightarrow Y$ durch $X \rightarrow (Y - B)$

end for

end for

Eliminiere FDs der Form $X \rightarrow \emptyset$

Vereinige FDs der Form $X \rightarrow Y_1, X \rightarrow Y_2, \ldots$ zu $X \rightarrow Y_1 Y_2 \ldots$

return resultierende FDs

Reduzierte Überdeckung: Beispiel

- Geg.: FD-Menge

$$F = \{f_1 : A \rightarrow B, f_2 : AB \rightarrow C, f_3 : A \rightarrow C, f_4 : B \rightarrow A, f_5 : C \rightarrow E\}$$

1. Linksreduktion: bei FD f_2 Attribut A streichen, da $C \subseteq \textsc{Closure}(\{A \rightarrow B, B \rightarrow C, A \rightarrow C, B \rightarrow A, C \rightarrow E\}, \{AB\})$
2. Rechtsreduktion: FD f_3 durch $A \rightarrow \{\}$ ersetzt, da $C \subseteq \textsc{Closure}(\{A \rightarrow B, B \rightarrow C, A \rightarrow \{\}, B \rightarrow A, C \rightarrow E\}, \{A\})$
3. Streichen von $A \rightarrow \{\}$
4. im letzten Schritt FDs mit derselben linken Seite B zu einer FD $B \rightarrow AC$ verschmelzen

- Ergebnis:

$$\textsc{ReducedCover}(F) = \{A \rightarrow B, B \rightarrow AC, C \rightarrow E\}$$

Bildung von Äquivalenzklassen

- FDs aus F zu Äquivalenzklassen zusammenfassen
- alle FDs in eine Klasse, die äquivalente linke Seiten haben
- linke Seiten zweier FDs sind äquivalent, wenn sie sich gegenseitig funktional bestimmen
- FDs $X_1 \rightarrow Y_1$ und $X_2 \rightarrow Y_2$ in einer Äquivalenzklasse, wenn $X_1 \rightarrow X_2$ und $X_2 \rightarrow X_1$ gelten

Beispiel: Äquivalenzklassen

$$\textsc{ReducedCover}(F) = \{A \rightarrow B, B \rightarrow AC, C \rightarrow E\}$$

- Es gilt $A \rightarrow B$ und $B \rightarrow A$
- ersten beiden FDs in eine Äquivalenzklasse C_1 aufnehmen
- Da zwar $A \rightarrow C$, aber nicht $C \rightarrow A$ gilt, bildet die dritte FD eine eigene Äquivalenzklasse C_2
- Äquivalenzklasseneinteilung von F:

$$\textsc{EquivClass}(F) := \{C_1 = \{A \rightarrow B, B \rightarrow AC\}, C_2 = \{C \rightarrow E\}\}$$

Minimale Überdeckungen

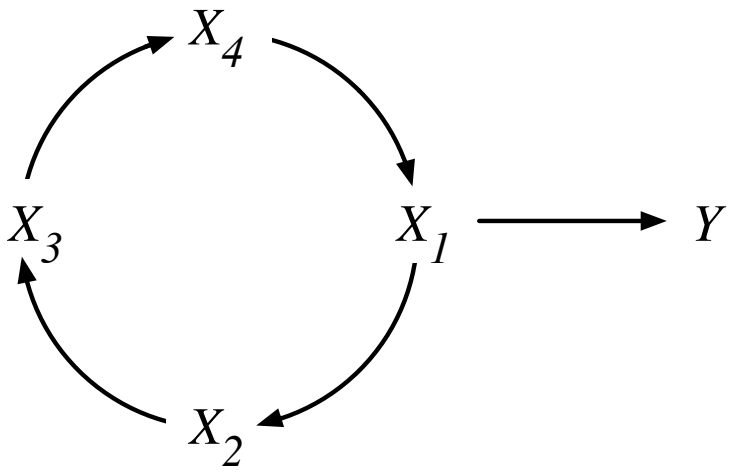
- Eine minimale Überdeckung ist eine Überdeckung, die eine minimale Anzahl von FDs enthält
- $=$ Auswahl der kleinsten aller nicht-redundanten Überdeckungen
- FD-Menge F heißt minimal gdw.

$$\forall F' \left[F' \equiv F \Rightarrow |F| \leq |F'| \right]$$

- Berechnung: In einigen Fällen können zwei FDs einer Äquivalenzklasse zu einer FD $X \rightarrow Y_1 Y_2$ zusammengefasst werden
  - zwei FDs $X_1 \rightarrow Y_1$ und $X_2 \rightarrow Y_2$ in einer Äquivalenzklasse, dann gilt $X_1 \rightarrow X_2$ und $X_2 \rightarrow X_1$
  - Können wir nachweisen, dass $X_1 \rightarrow X_2$ mit MEMBER-Algorithmus abgeleitet werden kann, ohne FDs aus der Äquivalenzklasse von $X_1 \rightarrow Y_1$ zu benutzen, so können wir die FDs zu einer FD $X_1 \rightarrow Y_1 Y_2$ zusammenfassen
- Dieser Schritt für alle Paare von FDs in allen Äquivalenzklassen, so ergibt sich minimale Überdeckung (MINIMALCOVER(F))

Ringförmige Überdeckungen

Da die FDs einer Äquivalenzklasse in die Form $X_1 \rightarrow X_2, X_2 \rightarrow X_3, \ldots, X_n \rightarrow X_1, X_1 \rightarrow Y$ überführt werden können, nennt man eine Überdeckung dieser Form eine ringförmige Überdeckung



Ein Ring (= eine Äquivalenzklasse) ergibt ein Relationenschema, die X_i sind alle Schlüssel dieses Relationenschemas

Syntheseverfahren: Ablauf

- Geg.: Relationenschema R mit FDs F
- Ges.: $R_1, \ldots R_n$ abhängigkeitstreu, minimal, alle R_i in 3NF
- Algorithmus:

SYNTHESIZE(F):

$F' :=$ **NONREDUNDANTCOVER**(F) /* *nicht-redundante Überdeckung* */

$F'' :=$ **REDUCEDCOVER**(F') /* *reduzierte Überdeckung von* F' */

$\bar{F} :=$ **EQUIVCLASS**(F'') /* *Äquivalenzklassen zu* F'' */

/* *Bilde Äquivalenzklassen* C_i *von FDs aus* F'' *mit gleichen oder* */

/* *äquivalenten linken Seiten, d.h.* $C_i = \{X_{i1} \rightarrow Y_{i1}, X_{i2} \rightarrow Y_{i2}, \ldots\}$ */

$\hat{F} :=$ **MINIMALCOVER**$(\bar{F})$ /* *minimale Überdeckung zu* $\bar{F}$ */

Bilde zu jeder Äquivalenzklasse C_i eine Ringform

$R_{C_i} = \{X_{i1} \cup X_{i2} \cup \ldots \cup Y_i\}$

mit Schlüsselmenge $K_{C_i} = \{X_{i1}, X_{i2}, \ldots\}$

return $\{(R_{C_1}, K_{C_1}), (R_{C_2}, K_{C_2}), \ldots\}$

Synthese: Beispiel

- FD-Menge

$$F = \{A \rightarrow B, AB \rightarrow C, A \rightarrow C, B \rightarrow A, C \rightarrow E\}$$

- minimale Überdeckung

$$\textsc{MinimalCover}(F) = \{A \rightarrow B, B \rightarrow AC, C \rightarrow E\}$$

- Zusammenfassung zu Äquivalenzklassen
  $\textsc{EquivClass}(F) = \{C_1, C_2\}$

$$\begin{aligned} C_1 &= \{A \rightarrow B, B \rightarrow AC\} \\ C_2 &= \{C \rightarrow E\} \end{aligned}$$

- Syntheseergebnis

$$(ABC, \{\{A\}, \{B\}\}), (CE, \{C\})$$

Erreichung der Verbundtreue

- Erreichen der Verbundtreue durch einfachen „Trick“:
  - Erweitern der Original-FD-Menge F um $\mathcal{U} \rightarrow \delta$ um Dummy-Attribut δ
  - δ wird nach Synthese entfernt
  - Dieser Trick erzeugt ein Relationenschema mit Universalschlüssel
- Beispiel: $\{A \rightarrow B, C \rightarrow E\}$
  - Syntheseergebnis $(AB, \{A\}), (CE, \{C\})$ ist nicht verbundtreu, da Universalschlüssel in keinem Schema enthalten ist
  - Dummy-FD $ABCE \rightarrow \delta$; reduziert auf $AC \rightarrow \delta$
  - liefert drittes Relationenschema

$$(AC, \{AC\})$$

Synthese: Beispiel

- Relationenschema und FD-Menge von Folie 5-66
- Ablauf
  1. minimale Überdeckung: Entfernen von f_2, f_3 sowie `Region` in f_5
  2. Äquivalenzklassen:

$$
\begin{aligned}
C_1 &= \{\texttt{Name}, \texttt{Weingut} \rightarrow \texttt{Preis}\} \\
C_2 &= \{\texttt{Name} \rightarrow \texttt{Farbe}\} \\
C_3 &= \{\texttt{Weingut} \rightarrow \texttt{Anbaugebiet}\} \\
C_4 &= \{\texttt{Anbaugebiet} \rightarrow \texttt{Region}\}
\end{aligned}
$$

  3. Ableitung der Relationenschemata
- Synthese genauer, inklusive vollständigem Algorithmus und Beweisen: Master-Vorlesung Theorie relationaler Datenbanken

Weitere Abhängigkeiten

- Mehrwertige Abhängigkeit (kurz: MVD)
  - innerhalb einer Relation r wird einem Attributwert von X eine Menge von Y-Werten zugeordnet, unabhängig von den Werten der restlichen Attribute $\rightsquigarrow$ Vierte Normalform
- Verbundabhängigkeit (kurz: JD)
  - kann R ohne Informationsverlust in $R_1, \ldots, R_p$ aufgetrennt werden: $\bowtie [R_1, \ldots, R_p]$
- Inklusionsabhängigkeit (kurz: IND)
  - auf der rechten Seite einer Fremdschlüsselabhängigkeit nicht unbedingt den Primärschlüssel einer Relation

Mehrwertige Abhängigkeiten

- Folge der 1NF
- Mehrwertige Abhängigkeiten erzeugen Redundanz:

WEIN_EMPFEHLUNG

WName	Jahrgang	Gericht
Chardonnay	2002	Geflügel
Chardonnay	2002	Fisch
Chardonnay	2003	Fisch
Chardonnay	2003	Geflügel
Shiraz	2003	Wild
Shiraz	2003	Lamm
Shiraz	2004	Wild
Shiraz	2004	Lamm

  - eine (oder mehrere) Gruppe von Attributwerten ist von einem Schlüssel bestimmt, unabhängig von anderen Attributen
  - hier: Menge von Jahrgängen plus Menge von Gerichten

$$\texttt{WName} \twoheadrightarrow \texttt{Jahrgang}, \texttt{WName} \twoheadrightarrow \texttt{Gericht}$$

  - Resultat: Redundanz durch Bildung aller Kombinationen

Mehrwertige Abhängigkeiten formal

- Relation $r(R)$ mit $X, Y \subseteq R$, $Z := R - (X \cup Y)$ genügt der MVD $X \twoheadrightarrow Y$ gdw.

$$\begin{aligned} \forall t_1, t_2 \in r : \quad & [(t_1 \neq t_2 \wedge t_1(X) = t_2(X)) \\ \Longrightarrow \quad & \exists t_3 \in r : t_3(X) = t_1(X) \wedge t_3(Y) = t_1(Y) \wedge \\ & t_3(Z) = t_2(Z)] \end{aligned}$$

- Relation $r(R)$ mit $R = XYZ$ und $X \twoheadrightarrow Y$:
  - wenn $(x_1, y_1, z_1) \in r$ und $(x_1, y_2, z_2) \in r$
  - dann auch: $(x_1, y_1, z_2) \in r$ und $(x_1, y_2, z_1) \in r$
- Bsp.: wegen ('Chardonnay', 2002, 'Geflügel') und ('Chardonnay', 2003, 'Fisch') müssen auch ('Chardonnay', 2002, 'Fisch') und ('Chardonnay', 2003, 'Geflügel') enthalten sein

Mehrwertige Abhängigkeiten und 4NF

- wünschenswerte Schemaeigenschaft beu Vorliegen von MVDs: vierte Normalform
- fordert die Beseitigung derartiger Redundanzen: keine zwei MVDs zwischen Attributen einer Relation
- Beispiel von Folie 5-86 verletzt diese Forderung
- Prinzip
  - Elimination der rechten Seite einer der beiden mehrwertigen Abhängigkeiten,
  - linke Seite mit dieser rechten Seite in neue Relation kopiert

Vierte Normalform

WEIN_JAHR

WName	Jahrgang
Chardonnay	2002
Chardonnay	2003
Shiraz	2003
Shiraz	2004

WEIN_GERICHT

WName	Gericht
Chardonnay	Geflügel
Chardonnay	Fisch
Shiraz	Wild
Shiraz	Lamm

Vierte Normalform formal

- Relationenschema R mit $X, Y \subseteq R$, MVD-Menge M über R
- MVD $X \twoheadrightarrow Y$ heißt trivial genau dann, wenn $Y \subseteq X$ oder $X \cup Y = R$

erweitertes Relationenschema $\mathcal{R} = (R, \mathcal{K})$ ist in vierter Normalform (4NF) bezüglich M genau dann, wenn für alle $X \twoheadrightarrow Y \in M^+$ gilt:

$X \twoheadrightarrow Y$ ist trivial oder $X \supseteq K$ für ein $K \in \mathcal{K}$.

Nichttriviale MVDs

- Erweiterung der Relation `WEIN_JAHR` von Folie 5-89 um Attribute `Farbe` und `Restsüße`
- MVD $\texttt{WName} \rightarrow\!\!\rightarrow \texttt{Jahrgang}$ ist nicht mehr trivial
- Zerlegung:

  $\texttt{WEIN\_JAHR1}(\underline{\texttt{WName, Jahrgang}})$
  $\texttt{WEIN\_JAHR2}(\underline{\texttt{WName}}\texttt{, Farbe, Restsüße})$

Zusammenfassung

- funktionale Abhängigkeiten (FDs)
- Normalformen (1NF - 3NF, BCNF)
- Abhängigkeitstreue und Verbundtreue
- Entwurfsverfahren
- mehrwertige Abhängigkeiten (MVDs)
- *Basis: Kapitel 5, 7 und 9 von [SSH18] (Biberbuch 1)*
  - Kapitel 5: Relationales Datenbankmodell
  - Kapitel 7: Relationaler Entwurf, Schema- und Transformationseigenschaften, Dekomposition
  - Kapitel 9: FD-Überdeckungen, Synthese, MVDs

Kontrollfragen

- Welches Ziel hat die Normalisierung relationaler Schemata?
- Welche Eigenschaften relationaler Schemata werden bei den Normalformen berücksichtigt?
- Was unterscheidet 3NF und BCNF?
- Was fordern Abhängigkeitstreue und Verbundtreue?
- Wie testet man 3NF, Abhängigkeitstreue und Verbundtreue? In welcher Reihenfolge?