

# Datenbanken 1

## Relationale Entwurfstheorie

Nikolaus Augsten

nikolaus.augsten@sbg.ac.at  
FB Computerwissenschaften  
Universität Salzburg



<http://dbresearch.uni-salzburg.at>

Sommersemester 2016

Version 29. Juni 2016

# Inhalt

- 1 Richtlinien für Relationalen Datenbankentwurf
- 2 Funktionale Abhängigkeiten
- 3 Zerlegung von Relationen
- 4 Normalformen

# Inhalt

- 1 Richtlinien für Relationalen Datenbankentwurf
- 2 Funktionale Abhängigkeiten
- 3 Zerlegung von Relationen
- 4 Normalformen

# Ziele des Relationalen Datenbankentwurfs

- Ziel des relationalen Entwurfs sind **gute Schemas** in der Datenbank.
- Die Hauptschwierigkeit ist es, eine **gute Gruppierung der Attribute** in relationale Schemas zu finden.

# Anomalien bei Datenänderung/1

- Beispiel Schema / Instanz:

AngProj(SVN, PNum, Stunden, AName, PName, POrt)

AngProj

SVN	PNum	Stunden	AName	PName	POrt
1234	1	32.5	Schmidt	ProjektX	Salzburg
1234	2	7.5	Schmidt	ProjektY	Wien
6688	3	40.5	Mair	ProjektZ	Linz
4567	1	20.0	Huber	ProjektX	Salzburg
4567	2	20.0	Huber	ProjektY	Wien
3334	2	10.0	Wong	ProjektY	Wien
3334	3	10.0	Wong	ProjektZ	Linz
3334	10	10.0	Wong	Computerization	Innsbruck
3334	20	10.0	Wong	Reorganization	Linz

- AngProj ist kein gutes Schema, da es unter **Anomalien** leidet.

# Anomalien bei Datenänderung/2

- Beispiel Schema:
  - AngProj(SVN, PNum, Stunden, AName, PName, POrt)
- Updateanomalie
  - Wenn der Ort eines Projektes geändert wird, muss er für alle Angestellten im Projekt geändert werden.
- Einfügeanomalie
  - Es kann kein Projekt ohne Angestellte eingefügt werden (SVN darf nicht *null* sein, da Teil des Schlüssels).
- Löschanomalie
  - Wenn ein Projekt gelöscht wird, werden als Nebeneffekt auch alle Angestellten gelöscht, die auf diesem Projekt arbeiten.

# Richtlinien für den relationalen Entwurf

- **Richtlinie 1:** Jedes Tupel einer Relation sollte nur die Instanz *einer* Entität oder Beziehung darstellen.
- **Richtlinie 2:** Update-, Einfüge- und Löschanomalien sollen vermieden werden.
- **Richtlinie 3:** Die Relationen sollen möglichst wenige *null* Werte enthalten; Attribute, die *null* Werte enthalten, kommen in eine eigene Relation (zusammen mit dem Primärschlüssel).
- **Richtlinie 4:** Durch einen natürlichen Join von Relationen sollen keine zusätzlichen (d.h. falschen) Tupel erzeugt werden.

# Inhalt

- 1 Richtlinien für Relationalen Datenbankentwurf
- 2 Funktionale Abhängigkeiten**
- 3 Zerlegung von Relationen
- 4 Normalformen



# Übersicht

- Was sind funktionale Abhängigkeiten?
- Armstrong-Axiome
- Richtigkeit und Vollständigkeit
- Hülle und kanonische Überdeckung

# Funktionale Abhängigkeiten/1

- **Funktionale Abhängigkeiten** (FDs – functional dependencies) werden zwischen Attributmengen  $X \subseteq sch(R)$  und  $Y \subseteq sch(R)$  einer Relation  $R$  definiert.
- **Definition:**  $Y$  ist von  $X$  **funktional abhängig** genau dann, wenn der Wert von  $X$  einen eindeutigen Wert von  $Y$  in  $R$  vorgibt:

$$X \rightarrow Y \Leftrightarrow \forall t_1, t_2 \in R : t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]$$

- $X \rightarrow Y$  bedeutet, dass  $Y$  von  $X$  **funktional abhängt**, bzw., dass die Attribute  $X$  die Attribute  $Y$  **funktional bestimmen**.
- FDs definieren eine **Einschränkung auf das Schema**, d.h., auf alle möglichen relationalen Instanzen von  $R$ .
- **Definition:** Eine Menge  $Y$  ist **trivial funktional abhängig** von  $X$  genau dann wenn  $Y \subseteq X$ .

# Funktionale Abhängigkeiten/2

- Wozu FDs?

- Funktionale Abhängigkeiten werden als **formales Maß** für die Qualität eines relationalen Entwurfs verwendet.
- Funktionale Abhängigkeiten und Schlüssel werden verwendet, um **Normalformen** für Relationen zu definieren.

- Woher kommen FDs?

FDs ergeben sich aus der **zugrundeliegenden Anwendung** und werden abgeleitet von

- der Bedeutung der Attribute,
- der Beziehung der Attribute untereinander.

- **Beispiele** Funktionaler Abhängigkeiten:

- Sozialversicherungsnummer bestimmt Angestelltenname:
  - $\{SVN\} \rightarrow \{AName\}$
- Projektnummer bestimmt Projektname und Projektort:
  - $\{PNum\} \rightarrow \{PName, POrt\}$
- Angestellten SVN und Projektnummer bestimmten die Anzahl der Wochenstunden, die der Angestellte auf dem Projekt arbeitet:
  - $\{SVN, PNum\} \rightarrow \{Stunden\}$

# Funktionale Abhängigkeiten/3

- Schema vs. Instanz:

- FDs sind auf dem Schema definiert und müssen für alle Instanzen gelten
- manche FDs können aufgrund einer gegebenen Instanz ausgeschlossen werden (weil diese die FD verletzen würde)

- Notation:

- Statt  $\{A, B\}$  schreiben wir  $AB$  (oder  $A, B$ ), z.B.  $AB \rightarrow BCD$  statt  $\{A, B\} \rightarrow \{B, C, D\}$ .
- Für eine Menge von Attributen  $X$  (z.B.,  $X = \{A, B, C\}$ ) und ein einzelnes Attribut  $A$  schreiben wir  $X - A$  statt  $X - \{A\}$ .

# Integrierte Übung 6.1

Betrachten Sie die abgebildete Instanz der Relation  $R[A, B, C]$ . Welche der folgenden Aussagen sind korrekt?

- a.  $B \rightarrow C$  gilt für die Relation  $R$ .
- b.  $C \rightarrow B$  gilt für die Relation  $R$ .
- c.  $BC \rightarrow A$  gilt in der abgebildeten Instanz von  $R$ .
- d.  $A$  ist der Primärschlüssel von  $R$ .

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

# Schlüssel (Auffrischung)

- **Superschlüssel**: Teilmenge der Attribute einer Relation, welche in jeder gültigen Ausprägung **eindeutige Werte** annimmt.
- Ein **Kandidatschlüssel**  $K$  ist ein Superschlüssel für den gilt, dass durch die Entfernung eines beliebigen Attributes von  $K$  die Superschlüssel-Eigenschaft von  $K$  verloren geht.
- Eine *beliebiger* Kandidatenschlüssel wird als **Primärschlüssel** ausgewählt.
- *Notation*: Die Attribute des Primärschlüssels werden unterstrichen:  
AngProj(SVN, PNum, Stunden, AName, PName, POrt)

# FDs und Schlüssel

- $K \subseteq sch(R)$  ist genau dann ein **Superschlüssel** von  $R$ , wenn

$$K \rightarrow sch(R),$$

d.h.  $K$  bestimmt *alle* Attribute in  $R$ .

- $K \subseteq sch(R)$  ist genau dann ein **Kandidatschlüssel** von  $R$ , wenn folgendes gilt:

1.  $K$  ist ein Superschlüssel von  $R$ , d.h.

$$K \rightarrow sch(R)$$

2.  $K$  kann nicht mehr verkleinert werden, ohne die Superschlüssel-Eigenschaft zu verlieren, d.h.

$$\forall A \in K : (K - A) \not\rightarrow sch(R)$$

# Armstrong-Axiome/1

- Für eine bestimmte Menge  $F$  von FDs können zusätzliche FDs hergeleitet werden, die immer gelten, wenn die FDs in  $F$  gelten.
- **Armstrong-Axiome**<sup>1</sup> (Inferenzregeln):
  - Reflexivität:  $Y \subseteq X \models X \rightarrow Y$
  - Verstärkung:  $X \rightarrow Y \models XZ \rightarrow YZ$
  - Transitivität:  $X \rightarrow Y, Y \rightarrow Z \models X \rightarrow Z$
- **Notation:**
  - $A \models B$  heißt: von  $A$  kann  $B$  hergeleitet werden
  - $XZ$  steht für  $X \cup Z$
- Die Armstrong-Axiome sind **korrekt** und **vollständig**:
  - Diese Regeln sind gültig (korrekt) und alle anderen gültigen Regeln können von diesen Regeln abgeleitet werden (vollständig).

---

<sup>1</sup>William W. Armstrong: Dependency Structures of Data Base Relationships, pages 580-583. IFIP Congress, 1974.



# Integrierte Übung 6.2

Zeige oder widerlege folgende Herleitungen:

1.  $X \rightarrow Y, Z \subseteq Y \models X \rightarrow Z$

2.  $X \rightarrow Y, X \rightarrow W, WY \rightarrow Z \models X \rightarrow Z$

3.  $XY \rightarrow Z, Y \rightarrow W \models XW \rightarrow Z$

# Armstrong-Axiome/2

- Folgende **zusätzliche Inferenzregeln** werden oft verwendet:
  - Dekompositionsregel:  $X \rightarrow YZ \models X \rightarrow Y, X \rightarrow Z$
  - Vereinigungsregel:  $X \rightarrow Y, X \rightarrow Z \models X \rightarrow YZ$
  - Pseudotransitivitätsregeln:  $X \rightarrow Y, WY \rightarrow Z \models WX \rightarrow Z$
- Diese zusätzlichen Inferenzregeln (und alle anderen möglichen Inferenzregeln) lassen sich aufgrund der Vollständigkeit der Armstrong-Axiome aus diesen ableiten.

# Hülle/1

- Die **Hülle**  $F^+$  (closure) der **Menge**  $F$  von FDs ist die Menge aller FDs die von  $F$  hergeleitet werden können.
- Die **Hülle**  $\mathcal{H}(F, X)$  einer **Menge von Attributen**  $X$  bezüglich  $F$  ist die Menge aller Attribute, welche von  $X$  funktional abhängig sind.
- $F^+$  und  $\mathcal{H}(F, X)$  können durch wiederholte Anwendung der Armstrong-Axiome berechnet werden.

## Hülle/2

- Die **Attribut-Hülle**  $\mathcal{H}(F, X)$  der Attributmengens  $X$  bezüglich  $F$  kann auch durch folgenden **Algorithmus** berechnet werden.

$\mathcal{H}(F, X)$

$Erg := X$

**while** (Änderungen an  $Erg$ ) **do**

**foreach** FD  $A \rightarrow B$  **in**  $F$  **do**

**if**  $A \subseteq Erg$  **then**  $Erg := Erg \cup B$

**return**  $Erg$

- Input: Menge  $F$  von FDs, Attributmengens  $X$
- Output: Attributhülle von  $X$  bezüglich  $F$

# Integrierte Übung 6.3

Gegeben die Relation  $R[A, B, C, D]$  mit der Menge  $F = \{AB \rightarrow D, B \rightarrow A, C \rightarrow B\}$  von FDs.

- Berechnen Sie die Attributhülle von  $C$ .
- Bestimmen Sie alle Kandidatenschlüssel von  $R$ .

# Überdeckung und Äquivalenz

- $F$  ist eine **Überdeckung** von  $G$  ( $F$  covers  $G$ ) wenn jede FD in  $G$  von  $F$  hergeleitet werden kann, i.e.,  $G^+ \subseteq F^+$ .
- Zwei Mengen  $F$  und  $G$  von FDs sind **äquivalent** genau dann wenn  $F^+ = G^+$ .
- Gleichbedeutende Definitionen von äquivalent:
  - Jede FD in  $F$  kann von  $G$  hergeleitet werden und jede FD in  $G$  kann von  $F$  hergeleitet werden.
  - $F$  ist eine Überdeckung von  $G$  und  $G$  ist eine Überdeckung von  $F$ .

# Membership und Äquivalenz

- **Membership-Problem:** Ist  $X \rightarrow Y$  in  $F^+$ ?
- Das Membership-Problem für  $X \rightarrow Y$  und einer Menge  $F$  von FDs kann folgendermaßen ausgedrückt werden:

$$X \rightarrow Y \in F^+ \Leftrightarrow Y \subseteq \mathcal{H}(F, X)$$

- Der Algorithmus zur Berechnung der Attributhülle kann also für das Membership-Problem angewandt werden.
- **Membership-Algorithmus zur Äquivalenz** zwischen  $F$  und  $G$ :
  1. Teste für alle FDs in  $F$  ob sie in  $G^+$  sind.
  2. Teste für alle FDs in  $G$  ob sie in  $F^+$  sind.
  3.  $F$  und  $G$  sind genau dann äquivalent, wenn alle Membership-Tests erfolgreich waren.

# Beispiel: Äquivalenz/1

Betrachte  $F = \{A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow H\}$  und  $G = \{A \rightarrow CD, E \rightarrow AH\}$ . Sind  $F$  und  $G$  äquivalent?

Zeigen oder widerlegen Sie die Äquivalenz indem Sie die folgenden Aussagen überprüfen:

1.  $F$  ist eine Überdeckung von  $G$  (mithilfe der Armstrong-Axiome)
2.  $G$  ist eine Überdeckung von  $F$  (mithilfe von Membership-Tests)



# Beispiel: Äquivalenz/2

1.  $F$  ist Überdeckung von  $G$ :

Für jedes  $X \rightarrow Y \in G$  überprüfen wir:  $X \rightarrow Y \in F^+$ ?

$A \rightarrow CD \in F^+$ :

Verstärkung  
 $A \rightarrow C \models A \rightarrow AC$

Transitivität  
 $A \rightarrow AC, AC \rightarrow D \models A \rightarrow D$

Vereinigung  
 $A \rightarrow C, A \rightarrow D \models A \rightarrow CD$

$E \rightarrow AH \in F^+$ :

Vereinigung  
 $E \rightarrow H, E \rightarrow AD \models E \rightarrow HAD$

Dekomposition  
 $E \rightarrow HAD \models E \rightarrow AH, E \rightarrow D$

2.  $G$  ist Überdeckung von  $F$ :

Für jedes  $X \rightarrow Y \in F$  überprüfen wir:  $X \rightarrow Y \in G^+$ ?

$A \rightarrow C$ :

$C \in \mathcal{H}(G, A) = \{A, C, D\}$

$\Rightarrow A \rightarrow C \in G^+$

$AC \rightarrow D$ :

$D \in \mathcal{H}(G, AC) = \{A, C, D\}$

$\Rightarrow AC \rightarrow D \in G^+$

$E \rightarrow AD$ :

$AD \in \mathcal{H}(G, E) = \{E, A, H, C, D\}$

$\Rightarrow E \rightarrow AD \in G^+$

$E \rightarrow H$ :

$H \in \mathcal{H}(G, E) = \{E, A, H, C, D\}$

$\Rightarrow E \rightarrow H \in G^+$

# Kanonische Überdeckung

- Zu einer gegebenen Menge  $F$  von FDs nennt man  $F_c$  eine **kanonische Überdeckung** wenn folgende drei Eigenschaften erfüllt sind:
  1.  $F_c^+ = F^+$  (d.h.,  $F_c$  und  $F$  sind äquivalent)
  2. In  $F_c$  existieren keine FDs  $X \rightarrow Y$ , bei denen  $X$  oder  $Y$  überflüssige Attribute enthalten, d.h., es muss gelten:
    - Keine FD  $X \rightarrow Y$  in  $F_c$  kann durch  $X' \rightarrow Y$  mit  $X' \subseteq X$  ersetzt werden ohne die Äquivalenz zu  $F$  zu verletzen.
    - Keine FD  $X \rightarrow Y$  in  $F_c$  kann durch  $X \rightarrow Y'$  mit  $Y' \subseteq Y$  ersetzt werden ohne die Äquivalenz zu  $F$  zu verletzen.
  3. Jede linke Seite einer funktionalen Abhängigkeit in  $F_c$  ist einzigartig.
- Die kanonische Überdeckung ist sozusagen eine **minimale Menge von FDs** welche noch äquivalent ist zu  $F$ .
- Jedes Menge  $F$  von FDs hat eine kanonische Überdeckung.
- Es kann mehrere kanonische Überdeckungen geben.

# Algorithmus für Kanonische Überdeckung/1

Eine kanonische Überdeckung der Menge  $F$  von FDs kann folgendermaßen berechnet werden.

1. **Linksreduktion:** Führe für alle  $X \rightarrow Y \in F$  eine Linksreduktion durch.

- Linksreduktion für  $X \rightarrow Y$ : Überprüfe für alle einzelnen Attribute  $A \in X$ :

$$Y \subseteq \mathcal{H}(F, X - A)$$

Falls dies gilt, ist  $A$  überflüssig und  $X \rightarrow Y$  wird in  $F$  durch  $(X - A) \rightarrow Y$  ersetzt:

$$F := F - \{X \rightarrow Y\} \cup \{(X - A) \rightarrow Y\}$$

2. **Rechtsreduktion:** Führe für alle (verbleibenden)  $X \rightarrow Y \in F$  eine Rechtsreduktion durch.

- Rechtsreduktion für  $X \rightarrow Y$ : Überprüfe für alle  $B \in Y$  ob

$$B \in \mathcal{H}(F - \{X \rightarrow Y\} \cup \{X \rightarrow (Y - B)\}, X)$$

Falls dies gilt, ist  $B$  auf der rechten Seite überflüssig und kann eliminiert werden, d.h.,  $X \rightarrow Y$  wird in  $F$  durch  $X \rightarrow (Y - B)$  ersetzt.

# Algorithmus für Kanonische Überdeckung/2

3. **Entfernen von leeren Mengen:** Entferne alle FDs der Form  $X \rightarrow \emptyset$ , die möglicherweise durch die Rechtsreduktion entstanden sind.
4. **Vereinigung:** Fasse mittels der Vereinigungsregel FDs der Form  $X \rightarrow Y_1, X \rightarrow Y_2, \dots, X \rightarrow Y_n$  zusammen zu  $X \rightarrow (Y_1 \cup Y_2 \cup \dots \cup Y_n)$ .

Eigenschaften des Algorithmus:

- Dieser Algorithmus erzeugt eine der möglichen kanonischen Überdeckungen.
- Je nach Ordnung der FDs können andere kanonische Überdeckungen herauskommen.

# Beispiel: Kanonische Überdeckung berechnen

Gegeben die Menge  $F = \{A \rightarrow B, A \rightarrow C, AB \rightarrow E, B \rightarrow ED\}$  von FDs.  
Bestimme die kanonische Überdeckung  $F_c$  von  $F$ .

1. Linksreduktion:  $F = \{A \rightarrow B, A \rightarrow C, B \rightarrow E, B \rightarrow ED\}$ 
  - wenn links nur 1 Attribut steht kann nie reduziert werden
  - $AB \rightarrow E$  könnte optional auch um  $B$  reduziert werden
2. Rechtsreduktion:  $F = \{A \rightarrow B, A \rightarrow C, B \rightarrow \emptyset, B \rightarrow ED\}$ 
  - jedes der fünf Attribute auf der rechten Seite muss überprüft werden
  - statt  $B \rightarrow E$  könnte optional  $B \rightarrow ED$  rechts um  $E$  reduziert werden
3. Entfernen von leeren Mengen:  $F = \{A \rightarrow B, A \rightarrow C, B \rightarrow ED\}$
4. Vereinigung:  $F = \{A \rightarrow BC, B \rightarrow ED\}$

Die Kanonische Überdeckung ist  $F_c = \{A \rightarrow BC, B \rightarrow ED\}$

# Inhalt

- 1 Richtlinien für Relationalen Datenbankentwurf
- 2 Funktionale Abhängigkeiten
- 3 Zerlegung von Relationen**
- 4 Normalformen

# Zerlegungen und deren Eigenschaften

- Zerlegung (decomposition) von Relationen
- Verlustlosigkeit (lossless join decomposition)
- Abhängigkeitsbewahrung (dependency preservation)

# Relationale Entwurf durch Zerlegung

- Sinn der Zerlegung einer Relation ist das **Vermeiden von Redundanzen**
- **Zerlegung**: Schema einer Relation  $R$  in Schemas  $Z = \{R_1, R_2, \dots, R_n\}$  zerlegen, sodass

$$sch(R) = sch(R_1) \cup sch(R_2) \cup \dots \cup sch(R_n)$$

- Für eine Instanz von  $R$  berechnen sich die **neuen Relationen  $R_i$** :

$$R_1 = \pi_{sch(R_1)}(R)$$

$$R_2 = \pi_{sch(R_2)}(R)$$

...

$$R_n = \pi_{sch(R_n)}(R)$$

- **Korrektheitskriterien** für Zerlegungen von  $R$  in  $Z = \{R_1, R_2, \dots, R_n\}$ :
  - **Verlustlosigkeit**: für jede Instanz muss  $R$  aus  $R_1, R_2, \dots, R_n$  rekonstruierbar sein.
  - **Abhängigkeitsbewahrung**: die FDs von  $R$  müssen auf  $R_1, R_2, \dots, R_n$  übertragbar sein.



# Integrierte Übung 6.4

Gegeben eine Relation  $R[A, B, C]$  mit der Instanz  
 $R = \{(\alpha, x, 0), (\beta, y, 2), (\gamma, z, 1), (\alpha, y, 2)\}$ .

Zerlege  $R$  in  $R_1[A, B]$ ,  $R_2[B, C]$ .

# Verlustlosigkeit

- **Verlustlosigkeit** (lossless join decomposition):  
Eine Zerlegung von  $R$  mit FDs  $F$  in  $R_1, R_2, \dots, R_n$  ist genau dann verlustlos, wenn für jede Instanz von  $R$  die  $F$  erfüllt gilt:

$$\pi_{sch(R_1)}(R) \bowtie \pi_{sch(R_2)}(R) \bowtie \dots \bowtie \pi_{sch(R_n)}(R) = R$$

- **Beachte:** Das Wort “verlustlos” bezieht sich auf Information, nicht die Anzahl der Tupel. Im Gegenteil: Joins auf nicht verlustlose Zerlegungen erzeugen zusätzliche (falsche) Tupel.
- **Satz:**  $R_1$  und  $R_2$  sind eine **verlustlose Zerlegung** von  $R$  bezüglich der FDs  $F$  genau dann wenn
  - $(sch(R_1) \cap sch(R_2)) \rightarrow sch(R_1)$  ist in  $F^+$  oder
  - $(sch(R_1) \cap sch(R_2)) \rightarrow sch(R_2)$  ist in  $F^+$
- **Intuition:** Die Join-Attribute zwischen  $R_1$  und  $R_2$  sollen entweder für  $R_1$  oder  $R_2$  einen Schlüssel darstellen, d.h., alle natürlichen Joins sind Fremdschlüssel-Joins.

# Integrierte Übung 6.5

Gegeben eine Relation  $R[A, B, C]$  mit der Instanz  $R = \{(\alpha, x, 0), (\beta, y, 2), (\gamma, z, 1), (\alpha, y, 2)\}$  und den funktionalen Abhängigkeiten  $FD = \{AC \rightarrow B, B \rightarrow C\}$ .

$R$  wird in  $Z = \{R_1[A, B], R_2[B, C]\}$  zerlegt.

- Überprüfe, ob für diese Instanz von  $R$  Verlustlosigkeit gilt.
- Ist die Zerlegung  $Z$  für  $R$  mit  $F$  für alle Instanzen verlustlos?
- Was passiert mit der Zerlegung, wenn das Tupel  $(\alpha, y, 2)$  durch  $(\alpha, z, 2)$  ersetzt wird, sodass  $B \rightarrow C$  nicht mehr erfüllt ist?



# Integrierte Übung 6.5

- c. Was passiert mit der Zerlegung, wenn das Tupel  $(\alpha, y, 2)$  durch  $(\alpha, z, 2)$  ersetzt wird, sodass  $B \rightarrow C$  nicht mehr erfüllt ist?

# Einschränkung

- Geben eine Zerlegung  $Z = \{R_1, R_2, \dots, R_n\}$  von  $R$  mit FDs  $F_R$ .
- Eine **Einschränkung**  $F_{R_i}$  von  $F_R$  auf  $R_i$  ist eine Menge von FDs, sodass  $F_{R_i}$  äquivalent ist zur Menge aller FDs  $X \rightarrow Y \in F_R^+$  mit  $(X \cup Y) \subseteq R_i$ .

- **Beispiel:**  $R[A, B, C, D]$ ,  $F_R = \{A \rightarrow C, C \rightarrow B\}$   
Zerlegung:  $R_1[A, B]$ ,  $R_2[C, D]$

Einschränkungen:  $F_{R_1} = \{A \rightarrow B\}$ ,  $F_{R_2} = \emptyset$

→ Es reicht also *nicht* die FDs von  $F_R$  zu übernehmen, in denen nur Attribute von  $R_i$  vorkommen

- $A \rightarrow B$  gehört zu  $F_{R_1}$  obwohl es in  $F_R$  nicht vorkommt
- $A \rightarrow B$  kommt jedoch in  $F_R^+$  vor

→  $F_{R_2}$  enthält nur triviale FDs.

# Berechnen der Einschränkung

- gegeben: Relation  $R$  mit FDs  $F_R$ , Zerlegung  $Z = \{R_1, R_2, \dots, R_n\}$
- gesucht: Einschränkung  $F_{R_i}$

- Algorithmus:

$$F_{R_i} = \emptyset$$

für jede echte Teilmenge  $X \subset \text{sch}(R_i)$ :

$$Y = \mathcal{H}(F_R, X) \setminus \{X\} \quad // \text{triviale FDs entfernen}$$

$$Y = Y \cap \text{sch}(R_i) \quad // \text{nur Attribute aus } R_i \text{ betrachten}$$

falls  $Y \neq \emptyset$ :

$$F_{R_i} = F_{R_i} \cup \{X \rightarrow Y\}$$

return  $F_{R_i}$

# Abhängigkeitsbewahrung/2

- **Abhängigkeitsbewahrung:** Eine Zerlegung  $Z = \{R_1, R_2, \dots, R_n\}$  von  $R$  mit FDs  $F_R$  ist abhängigkeitsbewahrend genau dann wenn für die entsprechenden Einschränkungen  $F_{R_i}$  gilt:

$$F_R^+ = (F_{R_1} \cup F_{R_2} \cup \dots \cup F_{R_n})^+$$

- **Intuition:** Bei abhängigkeitsbewahrender Zerlegung kann jede FD **lokal** auf einer Relation  $R_i$  geprüft werden.
- **Praktische Bedeutung** der Abhängigkeitsbewahrung:
  - FDs müssen bei jeder Änderung der Datenbank geprüft werden.
  - Wenn FDs nicht auf einzelnen Relation  $R_i$  geprüft werden können, muss ein Join  $R_i \bowtie R_j$  zur Prüfung durchgeführt werden.
  - Das ist in der Praxis viel zu teuer.



# Beispiel: Einschränkung u. Abhängigkeitsbewahrung

Gegeben  $R[A, C, D]$ ,  $F_R = \{A \rightarrow D, D \rightarrow C, C \rightarrow D\}$ , sowie die Zerlegung von  $R$  in  $R_1[A, C]$ ,  $R_2[C, D]$ .

- Berechnen Sie die Einschränkungen  $F_{R_1}$  und  $F_{R_2}$  von  $F_R$ .
- Ist diese Zerlegung abhängigkeitsbewahrend?

- Einschränkungen:

$$\mathcal{H}(F, A) \setminus \{A\} = \{D, C\}$$

$$\mathcal{H}(F, C) \setminus \{C\} = \{D\}$$

$$\mathcal{H}(F, D) \setminus \{D\} = \{C\}$$

$$F_{R_1} = \{A \rightarrow C\}$$

$$F_{R_2} = \{C \rightarrow D, D \rightarrow C\}$$

- Wir prüfen:  $F_R^+ = (F_{R_1} \cup F_{R_2})^+$   
 $\{A \rightarrow D, D \rightarrow C, C \rightarrow D\}^+ =$   
 $\{A \rightarrow C, C \rightarrow D, D \rightarrow C\}^+$

$F_R^+ \supseteq (F_{R_1} \cup F_{R_2})^+$ , da wir  $F_{R_1}$  und  $F_{R_2}$  aus  $F_R$  hergeleitet haben.

$$F^+ \subseteq (F_{R_1} \cup F_{R_2})^+?$$

$$A \rightarrow C, C \rightarrow D \stackrel{T}{\models} A \rightarrow D \quad \checkmark$$

# Inhalt

- 1 Richtlinien für Relationalen Datenbankentwurf
- 2 Funktionale Abhängigkeiten
- 3 Zerlegung von Relationen
- 4 Normalformen**

# Normalisierung/1

## Übersicht über die Normalformen:

- **1NF**: Attributwerte müssen atomar sein.
- **2NF, 3NF, BCNF**: basieren auf Schlüsseln und FDs einer Relation
- **4NF**: basieren auf Schlüsseln und mehrwertigen Abhängigkeiten (multi-valued dependencies, MVDs)
- **5NF**: basieren auf Schlüsseln und Join Dependencies (JDs)

Weiters müssen beim relationalen Entwurf berücksichtigt werden:

- **Verlustlosigkeit** der entsprechenden Joins (sehr wichtig, darf niemals geopfert werden)
- **Abhängigkeitsbewahrung** der funktionalen Abhängigkeiten (kann unter Umständen aufgegeben werden)

# Normalisierung/2

- **Normalisierung:** Die Attribute eines (schlechten) Schemas einer Relation werden auf kleinere (gute) Schemas aufteilen, welche den Normalformen genügen.
- Die Normalisierung wurde von **Codd im Jahr 1972** eingeführt.
- Während des **Normalisierungsprozesses** werden eine Reihe von Tests auf einem Schema durchgeführt um zu überprüfen, ob sich das Schema in einer bestimmten Normalform befindet.
- Eine **normalisierte Datenbank** besteht aus *guten* Schemas.
- **Praxistipp:** Datenbank-Designer brauchen nicht bis zur höchsten Normalform normalisieren:
  - es gibt einen Trade-off zwischen NF und Abfrage-Effizienz
  - normalerweise wird 3NF, BCNF oder 4NF ausgewählt

# Normalisierung/3

- **Kontrollierte Redundanz:**
  - Redundanz, welche dem System bekannt ist
  - kontrollierte Redundanz ist gut
  - Beispiele: Fremdschlüssel, Indices
- **Denormalisierung:**
  - Der Join mehrerer Relationen in einer höheren Normalform wird als Relation gespeichert.
  - Die Ergebnis-Relation befindet sich in einer niedrigeren Normalform, da Joins Normalformen zerstören.

# Erste Normalform (1NF)/1

- **Verbietet:**
  - zusammengesetzte Attribute
  - mehrwertige Attribute
  - verschachtelte Relationen: Attribute, deren Wert für jedes Tupel eine Relation ist
- 1NF wird oft als **Teil der Definition** einer Relation gesehen.
- **Beispiel:** Folgende Instanz der Relation *Fachbereiche* ist nicht in 1NF (mehrwertiges Attribut):

Fachbereiche

FName	FNum	LeiterSVN	Standorte
Research	5	334455	{ Salzburg, Wien, Linz }
Administration	4	987654	{ Innsbruck }
Headquarters	1	888666	{ Linz }

# Erste Normalform (1NF)/2

- **Abhilfe** um 1NF zu erhalten:
  - zusammengesetzte Attribute: jeder Teil wird ein eigenes Attribut
  - mehrwertige Attribute: neues Tupel für jeden Wert des mehrwertigen Attributs erzeugen
  - geschachtelte Relationen: neues Tupel für jedes Tupel der geschachtelten Relation erzeugen
  
- **Beispiel:** *Fachbereiche* in 1NF gebracht.

Fachbereiche\_1NF

FName	FNum	LeiterSVN	Standort
Research	5	334455	Salzburg
Research	5	334455	Wien
Research	5	334455	Linz
Administration	4	987654	Innsbruck
Headquarters	1	888666	Linz

# Zweite Normalform (2NF)/1

- **Zweite Normalform (2NF):** Eine Relation  $R$  befindet sich in der zweiten Normalform (2NF) genau dann wenn sie sich in 1NF befindet und jedes Nicht-Schlüssel Attribut voll funktional abhängig von allen Kandidatenschlüsseln ist.
- **Nicht-Schlüssel Attribut:** Attribut, das nicht Teil eines Kandidatenschlüssels (inklusive Primärschlüssel) ist.
- Ein Attribut  $A$  ist **voll funktional abhängig** vom Kandidatenschlüssel  $K$  ( $K \xrightarrow{\bullet} A$ ), wenn es keine echte Teilmenge  $X \subset K$  gibt, sodass  $X \rightarrow A$ :

$$K \xrightarrow{\bullet} A \Leftrightarrow K \rightarrow A \wedge \forall X \subset K : X \not\rightarrow A$$

- **Intuition:** 2NF ist verletzt, wenn mehrere Entitäten in einer einzigen Relation modelliert werden.



# Zweite Normalform (2NF)/2

- Beispiel: Folgende Relation ist nicht in 2NF:

AngProj

SVN	PNum	Stunden	AName	PName	POrt
1234	1	32.5	Schmidt	ProductX	Salzburg
1234	2	7.5	Schmidt	ProductY	Wien
6688	3	40.5	Mair	ProductZ	Linz
4567	1	20.0	Huber	ProductX	Salzburg
4567	2	20.0	Huber	ProductY	Wien
3334	2	10.0	Wong	ProductY	Wien
3334	3	10.0	Wong	ProductZ	Linz
3334	10	10.0	Wong	Computerization	Innsbruck
3334	20	10.0	Wong	Reorganization	Linz

- Warum ist *AngProj* nicht in 2NF?
  - Kandidatenschlüssel ist  $\{SVN, PNum\}$ , von dem aber nur *Stunden* voll funktional abhängig ist.
  - *SVN* ist ein Teilschlüssel der *AName* bestimmt.
  - *PNum* ist ein Teilschlüssel der *PName* und *POrt* bestimmt.

# Zweite Normalform (2NF)/3

- **Abhilfe** um 2NF zu erhalten:

- neue Relation für jeden Teilschlüssel mit seinen abhängigen Attributen
- eine Relation mit ursprünglichem Schlüssel und allen voll funktional abhängigen Attributen

- **Beispiel:** *AngProj*[*SVN*, *PNum*, *Stunden*, *AName*, *PName*, *POrt*]

- FDs:

- $\{SVN, PNum\} \rightarrow Stunden, SVN \rightarrow AName, PNum \rightarrow \{PName, POrt\}$

- $\{SVN, PNum\}$  ist einziger Kandidatenschlüssel.

- *SVN* und *PNum* sind Teilschlüssel mit abhängigen Attributen.

## 2NF Normalisierung von *AngProj*:

- *AngProj*<sub>1</sub>(*SVN*, *AName*)
- *AngProj*<sub>2</sub>(*PNum*, *PName*, *POrt*)
- *AngProj*<sub>3</sub>(*SVN*, *PNum*, *Stunden*)

# Integrierte Übung 6.6

Befinden sich die folgenden Relationen in 2NF?

- a.  $R[A, B, C]$  mit  $F = \{B \rightarrow C\}$
- b.  $R[A, B, C]$  mit  $F = \{A \rightarrow BC, B \rightarrow C\}$

## Zweite Normalform reicht nicht aus

- Für Relation *AngFB* gelten folgende funktionale Abhängigkeiten:
  - $fd1 : SVN \rightarrow sch(AngFB)$  (d.h. *SVN* ist Kandidatenschlüssel)
  - $fd2 : FNum \rightarrow \{FName, LeiterSVN\}$

AngFB

AName	SVN	Jahrgang	Adresse	FNum	FName	LeiterSVN
Schmidt	1234	1965	Linz	5	Research	2345
Schmidt	2345	1965	Linz	5	Research	2345
Wong	6688	1968	Linz	4	Admin	4567
Zelaya	4567	1941	Linz	4	Admin	4567
Borg	3334	1937	Dallas	4	Admin	4567

- *AngFB* ist in 2NF, dennoch gibt es Redundanz:
  - *FName* und *LeiterSVN* für einen Fachbereich werden für jeden Angestellten redundant abgelegt.
- Problem: *FName* und *LeiterSVN* sind transitiv abhängig vom Schlüssel *SVN*:
  - $SVN \rightarrow FNum, FNum \rightarrow FName$
  - $SVN \rightarrow FNum, FNum \rightarrow LeiterSVN$

# Dritte Normalform (3NF)

- **Dritte Normalform (3NF):** Eine Relation  $R$  befindet sich in 3NF genau dann wenn sie sich in 1NF befindet und für alle FDs  $X \rightarrow Y \in F^+$  mindestens eine der folgenden Bedingungen gilt:
  - $X \rightarrow Y$  ist trivial (d.h.  $Y \subseteq X$ )
  - $X$  ist ein Superschlüssel von  $R$
  - jedes Attribut  $A \in Y$  ist in einem Kandidatenschlüssel von  $R$  enthalten
- **Intuition:** 3NF verbietet transitive Abhängigkeiten.
- **3NF  $\subset$  2NF:** eine Relation in 3NF ist auch in 2NF

# Beispiel: Dritte Normalform (3NF)

- Wir betrachten Relation *AngFB* mit den funktionale Abhängigkeiten:
  - $fd1 : SVN \rightarrow sch(AngFB)$  (d.h. *SVN* ist Kandidatenschlüssel)
  - $fd2 : FNum \rightarrow \{FName, LeiterSVN\}$

AngFB

AName	SVN	Jahrgang	Adresse	FNum	FName	LeiterSVN
Schmidt	1234	1965	Linz	5	Research	2345
Schmidt	2345	1965	Linz	5	Research	2345
Wong	6688	1968	Linz	4	Admin	4567
Zelaya	4567	1941	Linz	4	Admin	4567
Borg	3334	1937	Dallas	4	Admin	4567

- $fd2$  erfüllt keine der drei Bedingungen für 3NF:
    - $fd2$  ist nicht trivial
    - *FNum* ist keine Superschlüssel von *AngFB*
    - *FName* ist in keinem Kandidatenschlüssel enthalten
- ⇒ *AngFB* ist nicht in 3NF.

# Synthesealgorithmen zur Zerlegung in 3NF/1

- **Synthesealgorithmen:** Zerlegt das Schema einer Relation  $R$  mit funktionalen Abhängigkeiten  $F$  in die Schemas  $R_1, R_2, \dots, R_n$  mit folgenden Eigenschaften:
  - alle  $R_i$  ( $1 \leq i \leq n$ ) sind in 3NF
  - die Zerlegung ist **verlustfrei**
  - die Zerlegung ist **abhängigkeitsbewahrend**

# Synthesealgorithms zur Zerlegung in 3NF/2

Relation  $R$  mit funktionalen Abhängigkeiten  $F$  in 3NF zerlegen:

1. Bestimme kanonische Überdeckung  $F_c$  zu  $F$ .
2. Für jede funktionale Abhängigkeit  $X \rightarrow Y \in F_c$ :
  - a. Kreiere eine Relation  $R_X$  mit dem Schema  $X \cup Y$ .
  - b. Ordne  $R_X$  die FDs  $F_X = \{X' \rightarrow Y' \in F_c \mid X' \cup Y' \subseteq sch(R_X)\}$  zu.
3. Falls keine der neuen Relation  $R_X$  einen Kandidatenschlüssel von  $R$  bezüglich  $F_c$  enthält, wähle einen Kandidatenschlüssel  $K \subseteq sch(R)$ :
  - a. Erzeuge eine Relation  $R_K$  mit Schema  $K$ .
  - b. Die FDs von  $R_K$  sind  $F_K = \emptyset$ .
4. Eliminiere  $R$  und alle neu erzeugten Relationen  $R_i$  die in einer anderen Relation  $R_j$  enthalten sind, d.h.,  $sch(R_i) \subseteq sch(R_j)$ .



# Beispiel: Synthesealgorithms

Zerlegen Sie  $R[A, B, C, D, E, G]$  mit  
 $F = \{A \rightarrow BD, AB \rightarrow E, B \rightarrow EG, C \rightarrow AB\}$  in 3NF.

1. kanonische Überdeckung  $F_c = \{A \rightarrow BD, B \rightarrow EG, C \rightarrow A\}$
2.  $R_A[A, B, D], F_A = \{A \rightarrow BD\}$   
 $R_B[B, E, G], F_B = \{B \rightarrow EG\}$   
 $R_C[C, A], F_C = \{C \rightarrow A\}$
3. Nichts zu tun, da Kandidatenschlüssel  $C$  in  $R_3$  enthalten.
4. Keine redundanten Teilschemas.

**3NF-Zerlegung:**  $R_A$  mit  $F_A$ ,  $R_B$  mit  $F_B$ ,  $R_C$  mit  $F_C$

# Boyce-Codd NormalForm/1

- **Boyce-Codd Normal Form (BCNF):** Eine Relation  $R$  ist in BCNF genau dann wenn sie in 1NF ist und für alle  $X \rightarrow Y \in F^+$  mindestens eine der folgenden Bedingungen gilt:
  - $X \rightarrow Y$  ist trivial (d.h.  $Y \subseteq X$ )
  - $X$  ist ein Superschlüssel von  $R$
- **BCNF  $\subset$  3NF:** Eine Relation in BCNF ist auch in 3NF.
- **Beispiel:** Folgende Relation ist in 3NF aber nicht in BCNF:

Lernen

Stud	Kurs	Buch
Schmidt	Data Structures	Bertram
Schmidt	Data Management	Martin
Hall	Compilers	Hoffman
Brown	Data Structures	Horowitz
Gale	Data Structures	Horowitz

# Boyce-Codd Normal Form/2

Lernen

Stud	Kurs	Buch
Schmidt	Data Structures	Bertram
Schmidt	Data Management	Martin
Hall	Compilers	Hoffman
Brown	Data Structures	Horowitz
Gale	Data Structures	Horowitz

- FDs in der Relation *Lernen*:
  - fd1:  $\{Stud, Kurs\} \rightarrow Buch$
  - fd2:  $Buch \rightarrow Kurs$
- *Lernen* ist in 3NF:
  - fd1:  $\{Stud, Kurs\}$  ist Kandidatenschlüssel (d.h. auch Superschlüssel)
  - fd2: *Kurs* ist im Kandidatenschlüssel enthalten
- *Lernen* ist nicht in BCNF:
  - fd2 ist nicht trivial und *Buch* ist kein Superschlüssel

# Boyce-Codd Normal Form/3

- Zerlegung von *Lernen* in  $KB[Kurs, \underline{Buch}]$  und  $SB[\underline{Stud}, \underline{Buch}]$ :

KB

Kurs	Buch
Data Structures	Bertram
Data Management	Martin
Compilers	Hoffman
Data Structures	Horowitz

SB

Stud	Buch
Schmidt	Bertram
Schmidt	Martin
Hall	Hoffman
Brown	Horowitz
Gale	Horowitz

- FDs in den Relationen *KB* und *SB*:
  - $KB : Buch \rightarrow Kurs$
  - $SB : \text{nur triviale Anhängigkeiten}$
  - $\{Kurs, Stud\} \rightarrow Buch$  von *Lernen* ist verloren gegangen

# Dekompositionsalgorithmus zur Zerlegung in BCNF/1

- **Dekompositionsalgorithmus:** Zerlegt das Schema einer Relation  $R$  mit funktionalen Abhängigkeiten  $F$  in die Schemas  $R_1, R_2, \dots, R_n$  mit folgende Eigenschaften:
  - alle  $R_i$  ( $1 \leq i \leq n$ ) sind in **BCNF**
  - die Zerlegung ist **verlustfrei**
- Die Zerlegung in BCNF ist in manchen Fällen **nicht abhängigkeitsbewahrend**.

# Dekompositionsalgorithmus zur Zerlegung in BCNF/2

Relation  $R$  mit funktionalen Abhängigkeiten  $F_R$  in BCNF zerlegen:

1.  $Z = \{R\}$
2. Solange es ein  $R_i \in Z$  gibt, sodass  $R_i$  nicht in BCNF ist:
  - a. Finde eine FD  $X \rightarrow Y \in F_{R_i}^+$ , sodass:
    - $X \cap Y = \emptyset$ , d.h. keine (teilweise) triviale Anhängigkeit und
    - $X \not\rightarrow sch(R_i)$ , d.h.  $X$  ist kein Superschlüssel von  $R_i$
  - b. Erweitere  $Y$  zu  $Y'$ , das möglichst viele abhängige Attribute enthält:

$$X \rightarrow Y' \quad \text{wobei } Y' = \mathcal{H}(F_{R_i}, X) - X$$

- c. Zerlege  $R_i$  in zwei Relationen  $R_{i1}$  und  $R_{i2}$  mit den Schemas
  - $sch(R_{i1}) = X \cup Y'$   
 $F_{R_{i1}}$  ist die Einschränkung von  $R_i$  auf  $R_{i1}$
  - $sch(R_{i2}) = R_i - Y'$   
 $F_{R_{i2}}$  ist die Einschränkung von  $R_i$  auf  $R_{i2}$
- d. Ersetze  $R_i$  durch  $R_{i1}$  und  $R_{i2}$  in  $Z$ .

# Beispiel: Dekompositionsalgorithmus/1

Gegeben  $R[A, B, C, D]$  mit  $F_R = \{A \rightarrow C, C \rightarrow D, B \rightarrow D\}$ . Zerlege das Schema verlustlos in BCNF. Ist die Zerlegung abhängigkeitsbewahrend?

## Lösung:

Nebenrechnung: Attributhülle der linken Seiten der FDs in  $F_R$

$$\mathcal{H}(F_R, A) = \{A, C, D\} \quad (A \rightarrow CD)$$

$$\mathcal{H}(F_R, B) = \{B, D\} \quad (B \rightarrow D)$$

$$\mathcal{H}(F_R, C) = \{C, D\} \quad (C \rightarrow D)$$

Einzigster Kandidatenschlüssel:  $AB$

# Beispiel: Dekompositionsalgorithmus/2

1.  $Z = \{R\}$
2. Loop 1:  $Z = \{R\}$ ,  $R$  ist nicht in BCNF
  - a. teste  $A \rightarrow C$ 

$$X = \{A\}, Y = \{C\}$$

$$\{A\} \cap \{C\} = \emptyset \quad \checkmark$$

$$A \not\rightarrow sch(R): \mathcal{H}(F_R, A) = \{A, C, D\} \neq sch(R) \quad \checkmark$$
  - b.  $X = \{A\}, Y = \{C\}, Y' = \mathcal{H}(F_R, A) - \{A\} = \{C, D\}$
  - c.  $R_1[A, C, D], F_{R_1} = \{A \rightarrow CD, C \rightarrow D\}$   
 $R_2[A, B], F_{R_2} = \emptyset$
  - d.  $Z = \{R_1, R_2\}$



## Beispiel: Dekompositionsalgorithmus/3

2. Loop 2:  $Z = \{R_1, R_2\}$

$R_2$  ist in BCNF, da  $F_{R_2} = \emptyset$

$R_1$  ist wegen  $C \rightarrow D$  nicht in BCNF:

- $C \rightarrow D$  ist nicht trivial
- $C$  ist kein Superschlüssel

a. teste  $C \rightarrow D$ :

$$X = \{C\}, Y = \{D\}$$

$$\{C\} \cap \{D\} = \emptyset \quad \checkmark$$

$$C \not\rightarrow sch(R_1): \mathcal{H}(F_R, C) = \{C, D\} \neq sch(R_1) \quad \checkmark$$

b.  $X = \{C\}, Y = \{D\}, Y' = \mathcal{H}(F_{R_1}, C) - \{C\} = \{D\}$

c.  $R_{11}[C, D], F_{R_{11}} = \{C \rightarrow D\}$

$$R_{12}[A, C], F_{R_{12}} = \{A \rightarrow C\}$$

d.  $Z = \{R_{11}, R_{12}, R_2\}$

$$F_{R_{11}} = \{C \rightarrow D\}, F_{R_{12}} = \{A \rightarrow C\}, F_{R_2} = \emptyset$$

Loop 3: alle Relationen in  $Z$  sind in BCNF  $\Rightarrow$  Ende

## Beispiel: Dekompositionsalgorithmus/4

Ist die Zerlegung  $Z = \{R_{11}, R_{12}, R_2\}$  mit  $F_{R_{11}} = \{C \rightarrow D\}$ ,  
 $F_{R_{12}} = \{A \rightarrow C\}$ ,  $F_{R_2} = \emptyset$  abhängigkeitsbewahrend?

$$F_Z = F_{R_{11}} \cup F_{R_{12}} \cup F_{R_2} = \{C \rightarrow D, A \rightarrow C\}$$

$F_R^+ \supseteq F_Z^+$  ist erfüllt, da  $F_{R_{11}}, F_{R_{12}}, F_{R_2}$  aus  $F_R$  abgeleitet wurden

$F_R^+ \subseteq F_Z^+$  ist nicht erfüllt, da  $B \rightarrow D$  verloren geht:

$$B \rightarrow D \in F_R^+ \text{ aber } B \rightarrow D \notin F_Z^+ : \mathcal{H}(F_Z, B) = \{B\}$$

Die Zerlegung ist nicht abhängigkeitsbewahrend.

# Zusammenfassung Normalformen

- Die **gebräuchlichsten Normalformen** sind:
  - 1NF: atomare Attribute
  - 2NF: jede Relation entspricht einer eigenen Entität
  - 3NF: keine transitiven Abhängigkeiten
  - BCNF: nur Schlüsselabhängigkeiten erlaubt
- Wenn eine Relation in **BCNF** ist, gibt es **keine Redundanz** aufgrund funktionaler Abhängigkeiten mehr.
- Für die Normalformen gilt:
$$\text{BCNF} \subset \text{3NF} \subset \text{2NF} \subset \text{1NF}$$
- Jede Relation lässt sich **verlustlos** bis zu **BCNF** zerlegen.
- Jede Relation lässt sich **abhängigkeitsbewahrend** bis zu **3NF** zerlegen.
- Bei der Zerlegung in BCNF können FDs verloren gehen.