

# Tutorium 10 - Normalformen

18.01.2023 – Finn Kapitza





LUDWIG-  
MAXIMILIANS-  
UNIVERSITÄT  
MÜNCHEN

# 1. Wiederholung - Normalformen



## Warum Normalformen?

- Redundanzen im DB-Schema erzeugen Anomalien:
    - Änderungs Anomalie – Wenn eine Änderung vergessen wird -> Inkonsistenz
    - Einfüge Anomalie – Einfügen eines partiellen Eintrags evtl. nicht möglich
    - Entfernungs Anomalie – Entfernen des letzten Eintrags löscht ungewollt Informationen
  - Ziele:
    - Vermeidung von Redundanzen und Anomalien
    - Schrittweise Beseitigung funktionaler Abhängigkeiten (außer vom gesamten Schlüssel)
- ⇒ Zerlegen des Schemas in ein äquivalentes Schema ohne Redundanzen und Anomalien = **Normalisierung**

Seien  $X, Y$  Attributmengen des Relationenschemas  $R$ , d.h.  $X, Y \subseteq R$

$Y$  ist von  $X$  **funktional abhängig** (oder  $X$  bestimmt  $Y$  funktional),  
d.h.  $X \rightarrow Y \Leftrightarrow$  für alle möglichen Ausprägungen von  $R$  gilt: Zu jedem Wert in  $X$   
existiert genau ein Wert in  $Y$

Formal:

$$X \rightarrow Y \Leftrightarrow \forall r_1, r_2 \in R: r_1.X = r_2.X \Rightarrow r_1.Y = r_2.Y$$

Beispiel: Passnummer  $\rightarrow$  Name

- **Triviale funktionale Abhängigkeit:**  $X \rightarrow Y$ , falls  $Y \subseteq X$ 
  - Bsp.: Passnummer  $\rightarrow$  Passnummer
- **Voll funktionale Abhängigkeit:**  $X \rightarrow Y$ , falls keine echte Teilmenge  $X' \subset X$  existiert mit  $X' \rightarrow Y$ 
  - Bsp.: Passnummer  $\rightarrow$  Name
- **Partiell funktionale Abhängigkeit:** Es existiert Teilmenge  $X' \subset X$  mit  $X' \rightarrow Y$ 
  - Bsp.: Passnummer, Land  $\rightarrow$  Name

- **Transitive funktionale Abhängigkeit:**  $X \rightarrow Z$ , falls gilt:  $X \rightarrow Y$  und  $Y \rightarrow Z$ 
  - Bsp.: Passnummer  $\rightarrow$  Ort, da Passnummer  $\rightarrow$  PLZ und PLZ  $\rightarrow$  Ort

Teilmenge  $S$  der Attribute eines Relationenschemas  $R$  heißt **Schlüssel**, falls gilt:

- 1) **Eindeutigkeit:** Keine mögliche Ausprägung von  $R$  kann zwei verschiedene Tupel enthalten, die sich in allen Attributen von  $S$  Gleichen
- 2) **Minimalität:** Keine echte Teilmenge von  $S$  erfüllt bereits Bedingung (1)

Ein Attribut heißt **prim**, falls es Teil eines Schlüsselkandidaten ist

**Eingabe:** eine Menge  $F$  von funktionalen Abhängigkeiten und eine Menge  $X$  von Attributen

**Ausgabe:** die vollständige Menge von Attributen  $X^+$  für die gilt  $X \rightarrow X^+$   
(also die Menge an Attributen die man von  $X$  mit allen  $F$  herleiten kann)

AttrHülle( $F, X$ )

Erg :=  $X$

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

**foreach** FD  $Y \rightarrow Z \in F$  **do**

**if**  $Y \subseteq \text{Erg}$  **then** Erg := Erg  $\cup$   $Z$

Ausgabe  $X^+ = \text{Erg}$

Solange es Änderungen an  $X^+$  gibt:

Gehe jede FD  $Y \rightarrow Z$  aus  $F$  durch:

Wenn linke Seite echte Teilmenge von aktueller  $X^+$  ist,  
dann ist  $Z$  in neuer  $X^+$



Zerlegung von Relation  $R$  in  $R_1, \dots, R_n$  ist:

- **Verlustlos**, falls gilt:
  - Jede mögliche Ausprägung  $r$  von  $R$  lässt sich durch den natürlichen Join der Ausprägungen  $r_1, \dots, r_n$  konstruieren:  $r = r_1 \bowtie \dots \bowtie r_n$
- **Abhängigkeitserhaltend**, falls gilt:
  - Alle  $FD \in F$  auf  $R$  bleiben in den lokalen funktionalen Abhängigkeiten  $F_i$  bewahrt:  $F = F_1 \cup \dots \cup F_n$

# 1. Normalform

- Alle Attribute enthalten **atomare** Werte (String, Integer, ...) und **keine** Tupel, Listen, usw.
- In relationalen DB sind nicht-atomare Werte eh nicht erlaubt/möglich

⇒ Relationale DB immer in **1. Normalform**

A	B	C	D
1	2	3 4	4 5
2	3	3	4
3	3	4 6	5 7



A	B	C	D
1	2	3	4
1	2	4	5
2	3	3	4
3	3	4	5
3	3	6	7

## 2. Normalform

- Für jedes Attribut  $A$  gilt:
  - $A$  ist **prim** oder
  - $A$  ist **voll funktional abhängig von jedem Schlüsselkandidaten**
- Beseitigung von partiell funktionalen Abhängigkeiten nicht-primer Attribute vom Schlüssel
- 2. NF kann nur verletzt werden wenn Schlüssel aus mehr als einem Attribut besteht und wenn nicht-prime Attribute existieren

## Transformation in 2. Normalform

- 1) Erstelle eine neue Relation für jeden partiellen Schlüssel mit seinen Abhängigen Attributen
- 2) Attribute, die voll funktional vom Schlüssel abhängig sind, bleiben in der ursprünglichen Relation

**Lieferant**(LNr, LName, LStadt, LLand, Ware, Preis)

-> LName, LStadt und LLand hängen nur von LNr ab

**LieferAdr**(LNr, LName, LStadt, Lland)

-> Preis hängt voll funktional vom Schlüssel ab

**Lieferung**(LNr, Ware, Preis)

### 3. Normalform

- Für alle nicht-trivialen funktionalen Abhängigkeiten  $X \rightarrow Y$  gilt:
  - ❖  $X$  enthält **Schlüsselkandidaten** oder
  - ❖  $Y$  ist **prim**
- Nicht-prime Attribute sind nur von (ganzen) Schlüsselkandidaten funktional abhängig
- Beseitigung von funktionalen Abhängigkeiten nicht-primärer Attribute untereinander (= transitive Abhängigkeiten)
- 3. Normalform setzt die 2. Normalform voraus

## Transformation in 3. Normalform

- 1) Erstelle eine neue Relation für alle Nicht-Schlüssel-Attribute und deren funktionalen Abhängigkeiten
- 2) Attribute, die voll funktional vom ursprünglichen Schlüssel abhängig und nicht abhängig von Nicht-Schlüssel-Attributen sind, bleiben in der ursprünglichen Relation

**LieferAdr**(LNr, LName, LStadt, LLand)

-> LLand ist von LStadt funktional abhängig

**LieferAdr**(LNr, LName, LStadt)

**StadtLand**(LStadt, LLand)

Synthesealgorithmus wird verwendet um beliebiges Relationenschema  $R$  mit funktionalen Abhängigkeiten  $F$  in Relationen  $R_1, \dots, R_n$  zu zerlegen für die gilt:

- $R_1, \dots, R_n$  ist eine **verlustlose** Zerlegung von  $R$
- $R_1, \dots, R_n$  ist **abhängigkeitserhaltend**
- $R_1, \dots, R_n$  sind alle in **3. Normalform**

## Synthesealgorithmus Schritt 1 – kanonische Überdeckung $F_c$ zu $F$

a) Linksreduktion:

Prüfe für jede  $X \rightarrow Y \in F$ :

Prüfe für jedes  $A \in X$ :

$$Y \subseteq \text{AttrHülle}(F, X - A)$$

Wenn obiges gilt, ist  $A$  in  $X$  überflüssig und kann aus  $X$  entfernt werden

$\Rightarrow$  Aus  $X \rightarrow Y$  wird  $(X - A) \rightarrow Y$



## b) Rechtsreduktion

Prüfe für jede (linksreduzierte)  $X \rightarrow Y \in F$ :

Prüfe für jedes  $B \in Y$ :

$$B \subseteq AttrH\ddot{u}lle \left( (F - (X \rightarrow Y)) \cup (X \rightarrow (Y - B)), X \right)$$

$(F - (X \rightarrow Y)) \cup (X \rightarrow (Y - B))$  bedeutet:  $X \rightarrow Y$  wird ersetzt durch  $X \rightarrow (Y - B)$

Wenn obiges gilt, ist  $B$  auf der rechten Seite überflüssig

$\Rightarrow$  Aus  $X \rightarrow Y$  wird  $X \rightarrow (Y - B)$

## Synthesealgorithmus Schritt 1 – kanonische Überdeckung $F_c$ zu $F$

- c) Entferne alle funktionalen Abhängigkeiten (FD) mit leere rechten Seite also:

$$X \rightarrow \{\}$$

- d) Fasse alle FDs mit gleicher linken Seite zusammen

$$\text{Aus } X \rightarrow Y_1, \dots, X \rightarrow Y_n \text{ wird } X \rightarrow (Y_1 \cup \dots \cup Y_n)$$

## Synthesealgorithmus Schritt 2 – Erzeuge Relationenschemas aus $F_c$

Für jede FD  $X \rightarrow Y \in F_c$ :

- Erzeuge Relationenschema  $R_X := X \cup Y$
- Ordne  $R_X$  die FDs  $F_X := \{X' \rightarrow Y' \in F_c \mid X' \cup Y' \in R_X\}$
- Schlüssel sind alle Attribute aus  $X$

## Synthesealgorithmus Schritt 3 – Rekonstruiere einen Schlüsselkandidaten

- Falls eines der in Schritt 2 erzeugten Schemata einen Schlüsselkandidaten von  $R$  bezüglich  $F_c$  enthält, ist nichts zu tun
- Wenn nicht:

Wähle einen Schlüsselkandidaten  $S \subseteq R$  aus und definiere folgendes Schema:

$$R_S := S \text{ mit } F_S := \{\}$$

## Synthesealgorithmus Schritt 4 – Eliminiere überflüssige Relationen

Eliminiere diejenigen Schemata  $R_X$ , die in einem anderen Relationenschema  $R_{X'}$  enthalten sind, d.h.  $R_X \subseteq R_{X'}$

## Boyce–Codd Normalform (BCNF)

- Für alle nicht-trivialen funktionalen Abhängigkeiten  $X \rightarrow Y$  gilt:
  - ❖  $X$  enthält **Schlüsselkandidaten**
- Beseitigt FD unter Attributen, die **prim** sind, aber nicht vollständig eine Schlüssel bilden
- BCNF impliziert 3. Normalform
- Man kann nicht immer eine BCNF-Zerlegen finden, die Abhängigkeiten bewahrt

## 2. Aufgaben



## Aufgabenstellung

<u>mnr</u>	hnr	hersteller	typ	ps	<u>fznr</u>	baujahr	km-stand	n-preis	h-preis	ek-preis
1	1	Opel	Kadett	60	K674	1990	10000	18000	13000	12000
1	1	Opel	Kadett	60	K634	1988	34000	18000	12000	9000
2	1	Opel	Vectra	90	V459	1990	15000	25000	18000	17000
3	1	Opel	Omega	110	O634	1987	45000	30000	22000	15000
4	2	VW	Golf	90	G789	1991	11000	25000	21000	16000
4	2	VW	Golf	90	G713	1991	31000	25000	16000	13000
5	2	VW	Golf	105	G762	1992	28000	28000	19000	17000
6	2	VW	Käfer	60	K634	1986	71000	19000	10000	8000

- Modelle (*mnr*) werden fortlaufend nummeriert
- Modell ist charakterisiert durch *hersteller*, *typ* und *psj*
- Für jedes Modell ist fahrzeugnummer (*fznr*) eindeutig -> {*mnr*, *fznr*} = Schlüssel



## Aufgabe 10.1 – Probleme bei nicht normalisierten Datenbanken

<u>mnr</u>	<u>hnr</u>	hersteller	typ	ps	<u>fznr</u>	baujahr	km-stand	n-preis	h-preis	ek-preis
1	1	Opel	Kadett	60	K674	1990	10000	18000	13000	12000
1	1	Opel	Kadett	60	K634	1988	34000	18000	12000	9000
2	1	Opel	Vectra	90	V459	1990	15000	25000	18000	17000
3	1	Opel	Omega	110	O634	1987	45000	30000	22000	15000
4	2	VW	Golf	90	G789	1991	11000	25000	21000	16000
4	2	VW	Golf	90	G713	1991	31000	25000	16000	13000
5	2	VW	Golf	105	G762	1992	28000	28000	19000	17000
6	2	VW	Käfer	60	K634	1986	71000	19000	10000	8000

⇒ Redundanzen

## Aufgabe 10.1 – Probleme bei nicht normalisierten Datenbanken

- **Einfüge-Anomalien:**

- Kein Modell ohne Fahrzeug
- Kein Hersteller ohne Modell und Fahrzeug

- **Änderungs-Anomalien:**

- Änderung der PS eines Modells muss in allen Tupeln eingetragen werden
- Ändert ein Hersteller seinen Namen -> Änderung in allen Tupeln

- **Entfernungs-Anomalien:**

- Mit letztem Fahrzeug eines Modells wird Modell-Information gelöscht
- Mit letztem Fahrzeug eines Hersteller-Information gelöscht

## Aufgabe 10.2 – 2.Normalform

Die Menge der vollen und nicht-trivialen funktionalen Abhängigkeiten sei im folgenden gegeben durch

$$F = \{ \\ mnr \rightarrow hnr, hersteller, typ, ps$$

$$hnr \rightarrow hersteller$$

$$mnr, fznr \rightarrow \underbrace{baujahr, km - stand, n - preis, h - preis, ek - preis}_{\}$$

## Aufgabe 10.2.a – Erläutern Sie, warum das gegebene Schema nicht in der 2. Normalform genügt

- Schlüsselkandidat ist  $SK = \{mnr, fznr\}$
- In 2NF wenn: Jedes Attribut ist prim oder voll fkt abhängig von jedem SK
- *Hnr, hersteller, typ, ps* sind nicht **voll funktional abhängig** vom SK und sind nicht **prim**

## Aufgabe 10.2.b – Überführen Sie die Relation in die 2.NF und geben Sie die so entstehenden Relationen an

- Transformation in 2. Normalform: Zerlegung der Relation „Auto“
- Erstelle eine neue Relation für jeden **partiellen Schlüssel** mit seinen abhängigen Attributen

Modell

<u>mnr</u>	hnr	hersteller	typ	ps
1	1	Opel	Kadett	60
2	1	Opel	Vectra	90
3	1	Opel	Omega	110
4	2	VW	Golf	90
5	2	VW	Golf	105
6	2	VW	Käfer	60

## Aufgabe 10.2.b – Überführen Sie die Relation in die 2.NF und geben Sie die so entstehenden Relationen an

2. Attribute, die **voll funktional** vom (ursprünglichen) Schlüssel abhängig sind, bleiben in der ursprünglichen Relation

Fahrzeug

<u>mnr</u>	<u>fznr</u>	baujahr	km-stand	n-preis	h-preis	ek-preis
1	K674	1990	10000	18000	13000	12000
1	K634	1988	34000	18000	12000	9000
2	V459	1990	15000	25000	18000	17000
3	O634	1987	45000	30000	22000	15000
4	G789	1991	11000	25000	21000	16000
4	G713	1991	31000	25000	16000	13000
5	G762	1992	28000	28000	19000	17000
6	K634	1986	71000	19000	10000	8000

## Aufgabe 10.3 – 3. Normalform

Überführen Sie das Schema aus 10.2 in die 3. Normalform

- Relation *Fahrzeug* (SK = {*mnr*, *fznr*}):
  - $mnr, fznr \rightarrow baujahr, km - stand, n - preis, h - preis, ek - preis$  ✓
  - > ist in 3. Normalform
- Relation *Modell* (SK = {*mnr*}):
  - $mnr \rightarrow hnr, hersteller, typ, ps$  ✓
  - $hnr \rightarrow hersteller$  ✗
  - > *hnr* enthält **keinen SK** und *hersteller* ist nicht **prim**

## Aufgabe 10.3 – Transformation in 3. Normalform

Weiter Zerlegung der Relation *Modell*:

1. Erstelle eine neue Relation für alle Nicht-Schlüssel-Attribute und deren funktionalen Abhängigkeiten

Hersteller

<u>hnr</u>	hersteller
1	Opel
2	VW



## Aufgabe 10.3 – Transformation in 3. Normalform

2. Attribute, die voll funktional vom ursprünglichen Schlüssel abhängig und nicht abhängig von Nicht-Schlüssel-Attributen sind, bleiben in der ursprünglichen Relation:

Modell

<u>mnr</u>	hnr	typ	ps
1	1	Kadett	60
2	1	Vectra	90
3	1	Omega	110
4	2	Golf	90
5	2	Golf	105
6	2	Käfer	60

## Aufgabe 10.5 – Synthesealgorithmus

Gegeben sei das folgende Relationenschema  $R(A, B, C, D, E, F)$ , sowie die Menge der zugehörigen nicht-trivialen funktionalen Abhängigkeiten:

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

## Aufgabe 10.5.a – Begründen Sie, warum $\{B, C\}$ der einzige SK ist

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

### 1. Eindeutigkeit:

- $AttrH\ddot{u}lle(F, \{B, C\}) = \{B, C, F, D, A, E\}$

### 2. Minimalität:

- $AttrH\ddot{u}lle(F, \{B\}) = \{B, A, E, F\} \neq \{A, B, C, D, E, F\}$
- $AttrH\ddot{u}lle(F, \{C\}) = \{C, F, D\} \neq \{A, B, C, D, E, F\}$

- Warum ist  $\{B, C\}$  der **einzige** SK?

⇒ Weder B noch C lassen sich herleiten (stehen nur auf linker Seite)

## Aufgabe 10.5.b – Bringen Sie das Relationenschema R mithilfe des Synthesealgorithmus in die 3. Normalform. Führen Sie jeden Schritt mit Begründung durch und kennzeichnen Sie ihn falls nichts zu tun ist

### 1. Bestimmung der kanonischen Überdeckung $F_c$ zu $F$

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

#### a) Linksreduktion:

- Alle FD mit linker Seite die aus einem Attribut besteht können nicht links reduziert werden
- $C, A \rightarrow D$  wird zu  $C \rightarrow D$ , da  $A$  “überflüssig”, aber  $C$  nicht:
  - $D \notin \text{AttrHülle}(F, \{C, A\} - \{C\}) = \text{AttrHülle}(F, \{A\}) = \{A\}$
  - $D \in \text{AttrHülle}(F, \{C, A\} - \{A\}) = \text{AttrHülle}(F, \{C\}) = \{C, F, D\}$

## Aufgabe 10.5.b - Synthesealgorithmus

### b) Rechtsreduktion:

$$F = \{C \rightarrow \cancel{D} \mid C \rightarrow F, D \mid B \rightarrow A, E \mid E \rightarrow F, A\}$$

- $C \rightarrow D$  wird zu  $C \rightarrow \emptyset$ , da:
  - $D \in \text{AttrHülle}((F - (C \rightarrow D)) \cup (C \rightarrow \emptyset), \{C\}) = \{C, F, D\}$
- $B \rightarrow A, E$  wird zu  $B \rightarrow E$ , da:
  - $A \in \text{AttrHülle}(F - (B \rightarrow A, E) \cup (B \rightarrow E), \{B\}) = \{B, E, F, A\}$

## Aufgabe 10.5.b – Synthesealgorithmus

### c) Entfernung von rechtsleeren Abhängigkeiten

$$F = \{C \rightarrow \emptyset \mid C \rightarrow F, D \mid B \rightarrow E \mid E \rightarrow F, A\}$$

wird zu

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

## Aufgabe 10.5.b – Syntheseargorithmus

### d) Zusammenfassen von Abhängigkeiten mit gleicher linker Seite

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

Nix zu tun

$$\Rightarrow F_c = \{C \rightarrow F, D \mid B \rightarrow E \mid E \rightarrow F, A\}$$

## Aufgabe 10.5.b – Synthesealgorithmus

2. Erzeugen eines neues Relationenschemas aus  $F_c$ :

- $R_1(\underline{C}, F, D) \quad \rightarrow \quad F_1 = \{C \rightarrow F, D\}$
- $R_2(\underline{B}, E) \quad \rightarrow \quad F_2 = \{B \rightarrow E\}$
- $R_3(\underline{E}, F, A) \quad \rightarrow \quad F_3 = \{E \rightarrow F, A\}$



## Aufgabe 10.5.b – Synthesalgorithmus

### 3. Rekonstruktion eines Schlüsselkandidaten:

Neue Relation für Schlüsselkandidaten  $\{B, C\}$

$$\Rightarrow R_4(\underline{B}, \underline{C}) \quad \rightarrow \quad F_4 = \emptyset$$

### 4. Elimination überflüssiger Relationen

In diesem Schritt ist nichts zu tun

⇒ Es ergeben sich folgende Relationen:

- $R_1(\underline{C}, F, D)$
- $R_2(\underline{B}, E)$
- $R_3(\underline{E}, F, A)$
- $R_4(\underline{B}, \underline{C})$

## Aufgabe 10.4 – Boyce-Codd Normalform

Geben Sie ein Beispiel an, bei dem die 3.Normalform noch nicht zu einem „guten“ Datenbankdesign führt, sondern erst die Zerlegung in eine BCNF alle Redundanzen beseitigt

Beispiel: **FachLehrerSchüler**(Fach, Lehrer, Schüler)

Es gilt:

- Jeder Schüler hat einen Lehrer pro Fach:  $Schüler, Fach \rightarrow Lehrer$
- Jeder Lehrer Vertritt nur ein Fach (aber zu jedem Fach kann es mehrere Lehrer geben:  $Lehrer \rightarrow Fach$ )
- $SKs = \{\{Schüler, Fach\}, \{Schüler, Lehrer\}\}$

## Aufgabe 10.4 – Boyce-Codd Normalform

### Normalformen:

- 3NF: ((Schüler,Fach) enthält **SK** und Fach ist **prim**) -> auch 2NF und 1NF
- BCNF: (Lehrer enthält **keinen SK**)

### Anomalien:

- Einfügen: kein Lehrer mit zugehörigem Fach ohne Schüler
- Entfernen: mit letztem Schüler wird Info über Lehrer und Fach gelöscht

### BCNF:

- **LehrerFach**(Lehrer, Fach)
  - **SchülerLehrer**(Schüler, Lehrer)
- > nicht abhängigkeiterhaltend



LUDWIG-  
MAXIMILIANS-  
UNIVERSITÄT  
MÜNCHEN



Finn Kapitza – [Finn.Kapitza@campus.lmu.de](mailto:Finn.Kapitza@campus.lmu.de)