



DATA DRIVEN  
DECISIONS

Sommersemester 2026

# Datenmanagement & -analyse

**Prof. Dr. Christoph M. Flath**  
*Lehrstuhl für Wirtschaftsinformatik und Business Analytics*

*Julius-Maximilians-Universität Würzburg*



- ▶ 1 Rückblick & Motivation
- 2 Funktionale Abhängigkeiten
- 3 Erste Normalform (1NF)
- 4 Zweite Normalform (2NF)
- 5 Dritte Normalform (3NF)
- 6 BCNF & Denormalisierung
- 7 Zusammenfassung



## Letzte Session:

- ER-Modell → Relationales Schema
- Primär- und Fremdschlüssel
- CREATE TABLE mit Constraints

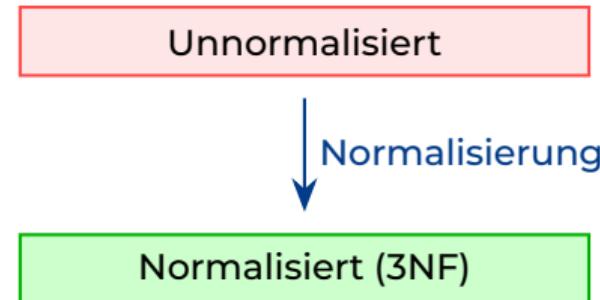
## Diese Session:

- Wann ist ein Schema “gut”?
- Systematische Qualitätsprüfung: **Normalformen**

# Motivation: Wann ist ein Schema "gut"?

## Probleme bei schlechtem Design:

- **Redundanz** – Daten mehrfach gespeichert
- **Änderungsanomalie** – Inkonsistenz bei Updates
- **Einfügeanomalie** – Fehlende Daten blockieren
- **Löschanomalie** – Ungewollter Datenverlust



Normalisierung = systematisches Verfahren zur Beseitigung von Redundanzen

## Ziel:

Schema so gestalten, dass diese Probleme **nicht auftreten können**.

**Bestellung\_Unnorm**

Best_Nr	Kunde	K_Stadt	Produkt	P_Preis	Menge
1001	Müller	München	Laptop	999	1
1001	Müller	München	Maus	29	2
1002	Schmidt	Berlin	Laptop	999	1
1003	Müller	München	Tastatur	79	1

## Probleme:

- “Müller, München” steht **dreimal** (Redundanz)
- “Laptop, 999” steht **zweimal** (Redundanz)
- Wenn Müller umzieht: **3 Zeilen** ändern (Änderungsanomalie)
- Neuer Kunde ohne Bestellung? **Nicht möglich** (Einfügeanomalie)

# Die drei Anomalien im Detail

## Änderungsanomalie:

Müller, München

Müller, München **Inkonsistent!**

Müller, Berlin

Update vergessen →  
Widerspruch in den Daten

## Einfügeanomalie:

Neuer Kunde

Bestellung = ?  
**NULL nicht erlaubt!**

Neuer Kunde ohne  
Bestellung nicht speicherbar

## Löschanomie:

~~Letzte Bestellung~~

von Schmidt  
**Kundendaten weg!**

Bestellung löschen →  
Kundendaten verloren

## Kernproblem

Alle Anomalien entstehen durch **Redundanz** – die gleiche Information an  
mehreren Stellen.

Ausleihe\_Unnorm

Ausleihe_Nr	Leser	L_Adresse	Buch	Autor	Datum
501	Anna	Hauptstr. 1	SQL Guide	Meier	01.03.2026
502	Anna	Hauptstr. 1	Python Basics	Schmidt	05.03.2026
503	Ben	Bahnweg 7	SQL Guide	Meier	10.03.2026
504	Anna	Hauptstr. 1	Java Kompakt	Müller	12.03.2026

## Identifizieren Sie die Redundanzen:

- Anna's Adresse erscheint \_\_\_\_\_ mal
- Der Autor von "SQL Guide" erscheint \_\_\_\_\_ mal
- Was passiert, wenn Anna umzieht?
- Was passiert, wenn wir die letzte Ausleihe von Ben löschen?

# Agenda

- 1 Rückblick & Motivation
- 2 **Funktionale Abhängigkeiten**
- 3 Erste Normalform (1NF)
- 4 Zweite Normalform (2NF)
- 5 Dritte Normalform (3NF)
- 6 BCNF & Denormalisierung
- 7 Zusammenfassung

# Funktionale Abhangigkeit (FD)

## Definition

Ein Attribut B ist **funktional abhangig** von A (geschrieben:  $A \rightarrow B$ ), wenn zu jedem Wert von A **genau ein** Wert von B gehort.

## Beispiele:

- Matrikelnr  $\rightarrow$  Studentename ✓
- PLZ  $\rightarrow$  Ort ✓ (in Deutschland)
- Ort  $\rightarrow$  PLZ ✗ (Munchen hat viele PLZ)
- ISBN  $\rightarrow$  Buchtitel ✓



“Wenn ich A kenne, kenne ich auch B.”

## Verschiedene Notationen (alle äquivalent):

Schreibweise	Bedeutung
$A \rightarrow B$	A bestimmt B
$A \rightarrow B$	A bestimmt B
B ist FD von A	B hängt funktional von A ab
$A \rightarrow B, C$	A bestimmt sowohl B als auch C
$A, B \rightarrow C$	Die Kombination von A und B bestimmt C

### Wichtig:

- $A \rightarrow B$  bedeutet **nicht**  $B \rightarrow A$
- Der Pfeil zeigt die **Richtung** der Abhängigkeit
- Die linke Seite heisst **Determinante**

# FDs in unserem Beispiel

**Bestellung\_Unnorm**(Best\_Nr, Kunde, K\_Stadt, Produkt, P\_Preis, Menge)

## Funktionale Abhängigkeiten:

- Best\_Nr, Produkt → Menge (Schlüssel!)
- Best\_Nr → Kunde, K\_Stadt (Bestellung → Kunde)
- Kunde → K\_Stadt (Kunde → Stadt)
- Produkt → P\_Preis (Produkt → Preis)

## Problem

Es gibt FDs, die **nicht vom gesamten Schlüssel** abhängen!  
Das ist die Ursache für Redundanz.

# Übung: FDs identifizieren

Mitarbeiter(MA\_Nr, Name, Abteilung, Abt\_Leiter, Projekt, Stunden)

MA_Nr	Name	Abteilung	Abt_Leiter	Projekt	Stunden
101	Anna	IT	Dr. Weber	Portal	20
101	Anna	IT	Dr. Weber	App	15
102	Ben	HR	Dr. Klein	Portal	30
103	Carla	IT	Dr. Weber	App	25

Welche FDs gibt es? (Ankreuzen)

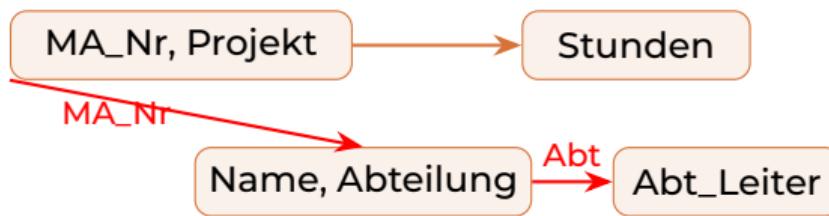
- MA\_Nr → Name
- MA\_Nr → Abteilung
- Abteilung → Abt\_Leiter
- MA\_Nr, Projekt → Stunden
- Projekt → Stunden

# Lösung: FDs identifizieren

Mitarbeiter(MA\_Nr, Projekt, Name, Abteilung, Abt\_Leiter, Stunden)

## Funktionale Abhängigkeiten:

- ✓ MA\_Nr → Name – Ein MA hat genau einen Namen
- ✓ MA\_Nr → Abteilung – Ein MA ist in genau einer Abteilung
- ✓ Abteilung → Abt\_Leiter – Jede Abteilung hat einen Leiter
- ✓ MA\_Nr, Projekt → Stunden – Schlüssel-FD
- ✗ Projekt → Stunden – Falsch! Verschiedene MAs arbeiten unterschiedlich lange



# Volle vs. Partielle Abhangigkeit

## Volle funktionale Abhangigkeit:

B ist **voll** funktional abhangig von A, wenn B von A abhangt, aber **nicht** von einer echten Teilmenge von A.

### Beispiel:

Best\_Nr, Produkt  $\rightarrow$  Menge  
 $\rightarrow$  Menge hangt vom **gesamten** Schlssel ab.

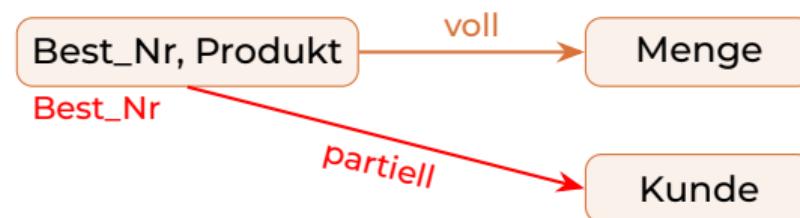
## Partielle Abhangigkeit:

B hangt nur von einem **Teil** des Schlussels ab.

### Beispiel:

Best\_Nr  $\rightarrow$  Kunde  
 $\rightarrow$  Kunde hangt nur von Best\_Nr ab, nicht von Produkt.

**Das verursacht Redundanz!**



# Armstrong's Axiome

## Regeln zur Herleitung von FDs:

### 1. Reflexivität

Wenn  $B \subseteq A$ , dann gilt  $A \rightarrow B$

Beispiel: Vorname, Nachname  $\rightarrow$  Nachname

### 2. Augmentation (Erweiterung)

Wenn  $A \rightarrow B$ , dann gilt auch  $A, C \rightarrow B, C$

Beispiel: Aus  $MA\_Nr \rightarrow Name$  folgt  $MA\_Nr, Projekt \rightarrow Name, Projekt$

### 3. Transitivität

Wenn  $A \rightarrow B$  und  $B \rightarrow C$ , dann gilt  $A \rightarrow C$

Beispiel:  $MA\_Nr \rightarrow Abteilung$  und  $Abteilung \rightarrow Leiter \Rightarrow MA\_Nr \rightarrow Leiter$

**Diese drei Axiome sind vollständig – alle ableitbaren FDs lassen sich damit herleiten!**

## Nützliche Zusatzregeln (ableitbar aus den Axiomen):

### Vereinigung:

Wenn  $A \rightarrow B$  und  $A \rightarrow C$ ,  
dann  $A \rightarrow B, C$

### Beispiel:

$ISBN \rightarrow \text{Titel}$  und  $ISBN \rightarrow \text{Autor}$   
 $\Rightarrow ISBN \rightarrow \text{Titel, Autor}$

### Dekomposition:

Wenn  $A \rightarrow B, C$ ,  
dann  $A \rightarrow B$  und  $A \rightarrow C$

### Beispiel:

$MA\_Nr \rightarrow \text{Name, Abteilung}$   
 $\Rightarrow MA\_Nr \rightarrow \text{Name}$  und  $MA\_Nr \rightarrow \text{Abteilung}$

## Pseudotransitivität

Wenn  $A \rightarrow B$  und  $B, C \rightarrow D$ , dann  $A, C \rightarrow D$

# Quiz: Welche FDs sind ableitbar?

**Gegeben:**

- $A \rightarrow B$
- $B \rightarrow C$
- $C, D \rightarrow E$

**Welche FDs können abgeleitet werden?**

FD	Ableitbar?	Begründung
$A \rightarrow C$		

# Quiz: Welche FDs sind ableitbar?

**Gegeben:**

- $A \rightarrow B$
- $B \rightarrow C$
- $C, D \rightarrow E$

**Welche FDs können abgeleitet werden?**

FD	Ableitbar?	Begründung
$A \rightarrow C$	✓	Transitivität: $A \rightarrow B \rightarrow C$
$A, D \rightarrow E$		

# Quiz: Welche FDs sind ableitbar?

**Gegeben:**

- $A \rightarrow B$
- $B \rightarrow C$
- $C, D \rightarrow E$

**Welche FDs können abgeleitet werden?**

FD	Ableitbar?	Begründung
$A \rightarrow C$	✓	Transitivität: $A \rightarrow B \rightarrow C$
$A, D \rightarrow E$	✓	Pseudotrans.: $A \rightarrow C, CD \rightarrow E$
$B \rightarrow E$		

# Quiz: Welche FDs sind ableitbar?

**Gegeben:**

- $A \rightarrow B$
- $B \rightarrow C$
- $C, D \rightarrow E$

**Welche FDs können abgeleitet werden?**

FD	Ableitbar?	Begründung
$A \rightarrow C$	✓	Transitivität: $A \rightarrow B \rightarrow C$
$A, D \rightarrow E$	✓	Pseudotrans.: $A \rightarrow C, CD \rightarrow E$
$B \rightarrow E$	✗	D fehlt für $C, D \rightarrow E$
$A \rightarrow B, C$		

# Quiz: Welche FDs sind ableitbar?

**Gegeben:**

- $A \rightarrow B$
- $B \rightarrow C$
- $C, D \rightarrow E$

**Welche FDs können abgeleitet werden?**

FD	Ableitbar?	Begründung
$A \rightarrow C$	✓	Transitivität: $A \rightarrow B \rightarrow C$
$A, D \rightarrow E$	✓	Pseudotrans.: $A \rightarrow C, CD \rightarrow E$
$B \rightarrow E$	✗	D fehlt für $C, D \rightarrow E$
$A \rightarrow B, C$	✓	Vereinigung: $A \rightarrow B, A \rightarrow C$

# Hands-on

## Funktionale Abhangigkeiten erkennen

marimo: 08-normalisierung.py

Aufgabe 8.1: FDs aus Beispieldaten ableiten

# Agenda

- 1 Rückblick & Motivation
- 2 Funktionale Abhängigkeiten
- **3 Erste Normalform (1NF)**
- 4 Zweite Normalform (2NF)
- 5 Dritte Normalform (3NF)
- 6 BCNF & Denormalisierung
- 7 Zusammenfassung

# Erste Normalform (1NF)

## Definition: 1NF

Eine Relation ist in **1NF**, wenn alle Attributwerte **atomar** sind (keine Listen, keine geschachtelten Strukturen).

### Nicht in 1NF:

Student	Kurse
Anna	DMA, BWL, Statistik
Ben	DMA

✗ "Kurse" enthält eine **Liste**

### In 1NF:

Student	Kurs
Anna	DMA
Anna	BWL
Anna	Statistik
Ben	DMA

✓ Jede Zelle enthält **einen** Wert

## Merke

1NF ist die **Mindestanforderung** für relationale Datenbanken!

## Typische Anzeichen:

- Komma-separierte Werte: "München, Berlin, Hamburg"
- Nummerierte Spalten: Telefon1, Telefon2, Telefon3
- JSON/XML in einer Spalte
- "Wiederholungsgruppen"

## Lösung:

- ① Wiederholende Werte in **separate Zeilen** aufteilen
- ② Oder: **Neue Tabelle** erstellen (bei 1:N oder M:N)

### Beispiel: Telefonnummern

Person(ID, Name, Telefon1, Telefon2, Telefon3)

↓

Person(ID, Name) + Telefon(Person\_ID, Nummer)

Ausgangstabelle (nicht 1NF):

Kurs_ID	Kursname	Dozenten
DMA01	Datenmanagement	Flath, Müller
BWL01	Einführung BWL	Schmidt

**Schritt 1:** Wiederholende Werte identifizieren → “Dozenten”

**Schritt 2:** Aufteilen in atomare Werte:

Kurs_ID	Kursname	Dozent
DMA01	Datenmanagement	Flath
DMA01	Datenmanagement	Müller
BWL01	Einführung BWL	Schmidt

**Hinweis:** Der Schlüssel hat sich geändert! Jetzt: (Kurs\_ID, Dozent)

# Quiz: Welche Tabelle ist in 1NF?

**Tabelle A:**

ID	Name	Skills
1	Anna	Java, Python
2	Ben	SQL

**Tabelle C:**

ID	Name	Skill1	Skill2
1	Anna	Java	Python
2	Ben	SQL	NULL

**Tabelle B:**

ID	Name	Skill
1	Anna	Java
1	Anna	Python
2	Ben	SQL

**Antwort:**

# Quiz: Welche Tabelle ist in 1NF?

**Tabelle A:**

ID	Name	Skills
1	Anna	Java, Python
2	Ben	SQL

**Tabelle B:**

ID	Name	Skill
1	Anna	Java
1	Anna	Python
2	Ben	SQL

**Tabelle C:**

ID	Name	Skill1	Skill2
1	Anna	Java	Python
2	Ben	SQL	NULL

**Antwort:**

- A: **X** Liste in Zelle
- B: **✓** Atomar!
- C: **X** Wiederholungsgruppe

# Agenda

- 1 Rückblick & Motivation
- 2 Funktionale Abhängigkeiten
- 3 Erste Normalform (1NF)
- ▶ **4 Zweite Normalform (2NF)**
- 5 Dritte Normalform (3NF)
- 6 BCNF & Denormalisierung
- 7 Zusammenfassung

# Zweite Normalform (2NF)

## Definition: 2NF

Eine Relation ist in **2NF**, wenn sie in 1NF ist und jedes Nicht-Schlüsselattribut **voll funktional abhängig** vom **gesamten** Primärschlüssel ist.

**Anders gesagt:** Keine partiellen Abhängigkeiten!



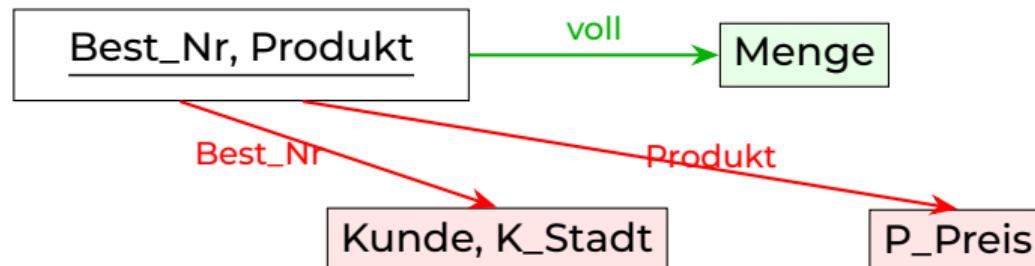
## Wichtig

2NF ist nur relevant bei **zusammengesetzten** Primärschlüsseln!  
Bei einfachem PK ist eine 1NF-Relation automatisch in 2NF.

**Bestellung\_Pos**(Best\_Nr, Produkt, Kunde, K\_Stadt, P\_Preis, Menge)

## Funktionale Abhängigkeiten:

- Best\_Nr, Produkt → Menge ✓ voll abhängig
- Best\_Nr → Kunde, K\_Stadt ✗ partiell!
- Produkt → P\_Preis ✗ partiell!



# 2NF herstellen: Zerlegung

**Lösung:** Attribute, die nur von einem Teil des Schlüssels abhängen, in **eigene Tabellen** auslagern.

**Vorher (nicht 2NF):**

Bestellung\_Pos(Best\_Nr, Produkt, Kunde, K\_Stadt, P\_Preis, Menge)

↓ Zerlegung

**Nachher (2NF):**

Bestellung(Best\_Nr, Kunde, K\_Stadt)

Produkt(Produkt, P\_Preis)

Best\_Position(Best\_Nr, Produkt, Menge)

## Ergebnis

Jede Tabelle hat nur noch Attribute, die **voll** vom Schlüssel abhängen.

# 2NF: Schritt-für-Schritt Walkthrough

**Ausgangstabelle:** Vorlesung(VL\_Nr, Dozent\_ID, VL\_Name, Raum, Dozent\_Name)

## Schritt 1: FDs identifizieren

- VL\_Nr, Dozent\_ID → Raum – voll (Dozent kann unterschiedliche Räume haben)
- VL\_Nr → VL\_Name – **partiell!**
- Dozent\_ID → Dozent\_Name – **partiell!**

## Schritt 2: Partielle FDs auslagern

- Neue Tabelle für VL\_Nr → VL\_Name
- Neue Tabelle für Dozent\_ID → Dozent\_Name

## Schritt 3: Ergebnis (2NF)

- Vorlesung(VL\_Nr, VL\_Name)
- Dozent(Dozent\_ID, Dozent\_Name)
- VL\_Dozent(VL\_Nr, Dozent\_ID, Raum)

## Achtung!

Bei einem **einfachen** (nicht zusammengesetzten) Primärschlüssel gibt es **keine partiellen Abhängigkeiten**.

### Beispiel:

Kunde(Kunden\_ID, Name, Stadt, PLZ)

- Kunden\_ID → Name – voll (Schlüssel hat nur 1 Attribut)
- Kunden\_ID → Stadt – voll
- Kunden\_ID → PLZ – voll

⇒ Diese Relation ist **automatisch in 2NF!**

**Aber:** Es könnte trotzdem eine 3NF-Verletzung geben (transitive Abhängigkeit: PLZ → Stadt)

# Quiz: Welche NF?

**Gegeben:** Buchung(Hotel\_ID, Gast\_ID, Datum, Zimmer, Hotel\_Name, Gast\_Name)

**FDs:**

- Hotel\_ID, Gast\_ID, Datum → Zimmer
- Hotel\_ID → Hotel\_Name
- Gast\_ID → Gast\_Name

**In welcher Normalform ist die Relation?**

- A) Nicht in 1NF
- B) In 1NF, aber nicht 2NF
- C) In 2NF, aber nicht 3NF
- D) In 3NF

# Quiz: Welche NF?

**Gegeben:** Buchung(Hotel\_ID, Gast\_ID, Datum, Zimmer, Hotel\_Name, Gast\_Name)

**FDs:**

- Hotel\_ID, Gast\_ID, Datum → Zimmer
- Hotel\_ID → Hotel\_Name
- Gast\_ID → Gast\_Name

**In welcher Normalform ist die Relation?**

- A) Nicht in 1NF
- B) In 1NF, aber nicht 2NF
- C) In 2NF, aber nicht 3NF
- D) In 3NF

**Antwort: B)** – Partielle Abhängigkeiten vorhanden (Hotel\_Name, Gast\_Name)

# Hands-on

## 1NF und 2NF anwenden

marimo: 08-normalisierung.py

Aufgaben 8.2 – 8.3

**Pause  
15 Minuten  
Kaffee holen!**

# Agenda

- 1 Rückblick & Motivation
- 2 Funktionale Abhängigkeiten
- 3 Erste Normalform (1NF)
- 4 Zweite Normalform (2NF)
- ▶ **5 Dritte Normalform (3NF)**
- 6 BCNF & Denormalisierung
- 7 Zusammenfassung

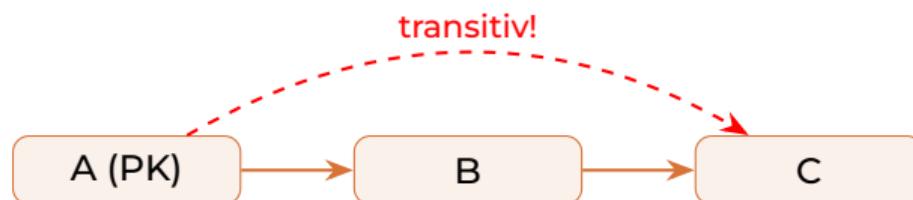
# Dritte Normalform (3NF)

## Definition: 3NF

Eine Relation ist in **3NF**, wenn sie in 2NF ist und kein Nicht-Schlüsselattribut **transitiv** vom Primärschlüssel abhängt.

### Transitive Abhängigkeit:

$A \rightarrow B$  und  $B \rightarrow C \Rightarrow C$  hängt **transitiv** von A ab.



**Problem:** C ist redundant gespeichert – wenn B sich wiederholt, wird auch C wiederholt!

Bestellung(Best\_Nr, Kunde, K\_Stadt)

## Funktionale Abhangigkeiten:

- $\text{Best\_Nr} \rightarrow \text{Kunde}$  ✓
- $\text{Kunde} \rightarrow \text{K\_Stadt}$  (Kunde bestimmt Stadt)
- $\Rightarrow \text{Best\_Nr} \rightarrow \text{K\_Stadt}$  ✗ transitiv!



**Redundanz:** Wenn Mller 5 Bestellungen hat, steht ‘‘Mnchen’’ 5x in der Tabelle.

# 3NF herstellen: Zerlegung

**Lösung:** Die transitive Abhängigkeit in eine **eigene Tabelle** auslagern.

**Vorher (nicht 3NF):**

Bestellung(Best\_Nr, Kunde, K\_Stadt)

↓ Zerlegung

**Nachher (3NF):**

Bestellung(Best\_Nr, #Kunde)

Kunde(Kunde, K\_Stadt)

## Ergebnis

Jedes Nicht-Schlüsselattribut hängt **direkt** (nicht transitiv) vom Schlüssel ab.

**Ausgangstabelle (2NF):** Mitarbeiter(MA\_Nr, Name, Abt\_ID, Abt\_Name, Abt\_Ort)

## Schritt 1: Transitive FDs finden

- $\text{MA\_Nr} \rightarrow \text{Abt\_ID}$  – direkt
- $\text{Abt\_ID} \rightarrow \text{Abt\_Name}$  – nicht vom PK!
- $\text{Abt\_ID} \rightarrow \text{Abt\_Ort}$  – nicht vom PK!
- $\Rightarrow \text{MA\_Nr} \rightarrow \text{Abt\_Name}$  – **transitiv!**
- $\Rightarrow \text{MA\_Nr} \rightarrow \text{Abt\_Ort}$  – **transitiv!**

## Schritt 2: Transitive Attribute auslagern

## Schritt 3: Ergebnis (3NF)

- Mitarbeiter(MA\_Nr, Name, #Abt\_ID)
- Abteilung(Abt\_ID, Abt\_Name, Abt\_Ort)

# Beispiel: Universitätssystem

## Ausgangstabelle:

Einschreibung(Matrikel, Student\_Name, Kurs\_ID, Kurs\_Name, Dozent\_ID,  
Dozent\_Name, Note)

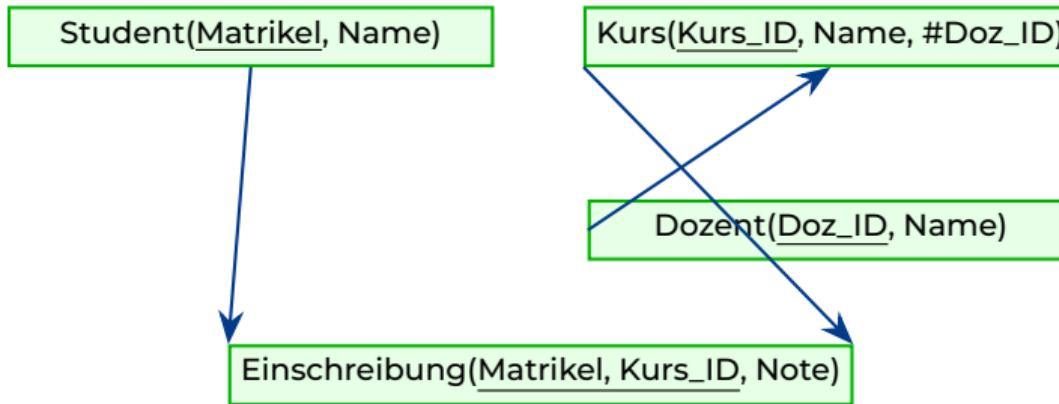
## FDs:

- Matrikel → Student\_Name
- Kurs\_ID → Kurs\_Name, Dozent\_ID
- Dozent\_ID → Dozent\_Name
- Matrikel, Kurs\_ID → Note

## Probleme:

- Partielle Abhängigkeit: Student\_Name von Matrikel (nicht 2NF)
- Partielle Abhängigkeit: Kurs\_Name, Dozent\_ID von Kurs\_ID (nicht 2NF)
- Transitive Abhängigkeit: Dozent\_Name über Dozent\_ID (nicht 3NF)

## Normalisierte Tabellen (3NF):

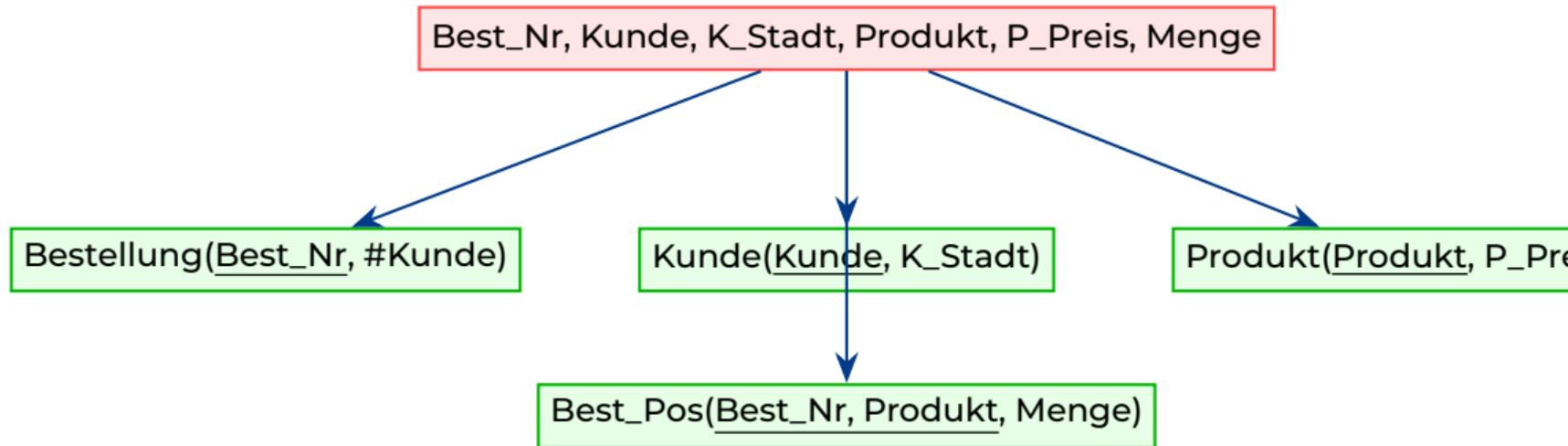


## Vorteile:

- Studentendaten nur einmal gespeichert
- Kursdaten nur einmal gespeichert
- Dozentendaten nur einmal gespeichert
- Keine Anomalien mehr möglich!

## Ausgangstabelle (nicht normalisiert):

Bestellung\_Unnorm(Best\_Nr, Kunde, K\_Stadt, Produkt, P\_Preis, Menge)



**Ergebnis: 4 Tabellen in 3NF – keine Redundanz, keine Anomalien!**

# Quiz: Welche Normalform?

**Bewerten Sie jede Relation:**

Relation	NF?
Film( <u>ID</u> , Titel, Regisseur, Genre, Genre_Beschreibung)	

# Quiz: Welche Normalform?

Bewerten Sie jede Relation:

Relation	NF?
Film( <u>ID</u> , Titel, Regisseur, Genre, Genre_Beschreibung)	nicht 3NF
Buch( <u>ISBN</u> , Titel, Autor1, Autor2, Autor3)	

# Quiz: Welche Normalform?

Bewerten Sie jede Relation:

Relation	NF?
Film( <u>ID</u> , Titel, Regisseur, Genre, Genre_Beschreibung)	nicht 3NF
Buch( <u>ISBN</u> , Titel, Autor1, Autor2, Autor3)	nicht 1NF
Bestellung( <u>Best_Nr</u> , Artikel, Preis, Menge, Kunde_Name)	

# Quiz: Welche Normalform?

Bewerten Sie jede Relation:

Relation	NF?
Film( <u>ID</u> , Titel, Regisseur, Genre, Genre_Beschreibung)	nicht 3NF
Buch( <u>ISBN</u> , Titel, Autor1, Autor2, Autor3)	nicht 1NF
Bestellung( <u>Best_Nr</u> , Artikel, Preis, Menge, Kunde_Name)	nicht 2NF
Mitarbeiter( <u>MA_ID</u> , Name, Gehalt)	

# Quiz: Welche Normalform?

Bewerten Sie jede Relation:

Relation	NF?
Film( <u>ID</u> , Titel, Regisseur, Genre, Genre_Beschreibung)	nicht 3NF
Buch( <u>ISBN</u> , Titel, Autor1, Autor2, Autor3)	nicht 1NF
Bestellung( <u>Best_Nr</u> , Artikel, Preis, Menge, Kunde_Name)	nicht 2NF
Mitarbeiter( <u>MA_ID</u> , Name, Gehalt)	3NF

Erklärungen:

- Film: Genre → Genre\_Beschreibung ist transitiv
- Buch: Wiederholungsgruppe (mehrere Autor-Spalten)
- Bestellung: Preis hängt nur von Artikel ab (partiell)
- Mitarbeiter: Alle Attribute hängen direkt vom Schlüssel ab

# Häufige Fehler bei der Normalisierung

## Fehler 1: FDs übersehen

- Alle Geschäftsregeln beachten!
- "Jeder Kunde hat genau eine Adresse" → FD

## Fehler 2: Schlüssel falsch bestimmt

- Schlüssel muss minimal sein
- Alle Attribute bestimmen

## Fehler 3: Zu früh aufhören

- 2NF reicht nicht!
- Transitive FDs prüfen

## Fehler 4: Zu viel zerlegen

- Nur bei echten Verletzungen zerlegen
- JOINs kosten Performance

### Tipp

Bei Unsicherheit: Beispieldaten durchspielen! Tritt Redundanz auf?

# Agenda

- 1 Rückblick & Motivation
- 2 Funktionale Abhängigkeiten
- 3 Erste Normalform (1NF)
- 4 Zweite Normalform (2NF)
- 5 Dritte Normalform (3NF)
- ▶ **6 BCNF & Denormalisierung**
- 7 Zusammenfassung

# Boyce-Codd-Normalform (BCNF)

## Definition: BCNF

Eine Relation ist in **BCNF**, wenn für jede nicht-triviale FD  $X \rightarrow Y$  gilt: X ist ein **Superschlüssel**.

## BCNF ist strenger als 3NF:

- 3NF erlaubt: Schlüsselattribut hängt von Nicht-Schlüssel ab
- BCNF verbietet das

## In der Praxis:

- Die meisten 3NF-Relationen sind auch BCNF
- BCNF kann manchmal nicht verlustfrei erreicht werden
- Für Klausuren: 3NF reicht meist!

## Merksatz

“Jedes Attribut hängt vom Schlüssel ab, vom ganzen Schlüssel, und von nichts ausser dem Schlüssel.” (Codd)

**Beispiel:** Kurs(Student, Fach, Dozent)

**FDs:**

- Student, Fach → Dozent – Schlüssel-FD
- Dozent → Fach – Jeder Dozent unterrichtet nur ein Fach

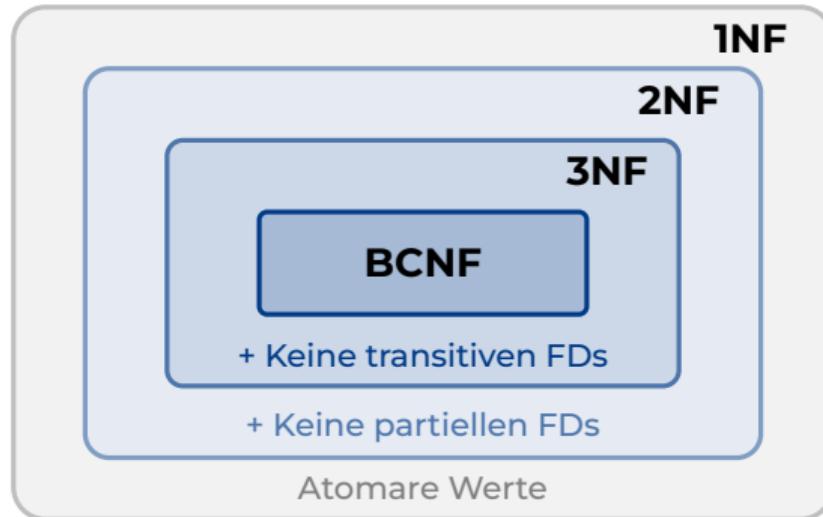
**3NF-Prüfung:**

- Keine partiellen Abhängigkeiten (Dozent hängt vom ganzen Schlüssel ab)
- Keine transitiven Abhängigkeiten (Dozent ist kein Nicht-Schlüsselattribut in der zweiten FD)
- ⇒ ✓ Ist in 3NF!

**BCNF-Prüfung:**

- Dozent → Fach – Dozent ist kein Superschlüssel!
- ⇒ ✗ Ist NICHT in BCNF!

# Normalformen: Übersicht



Normalform	Anforderung
1NF	Atomare Werte
2NF	+ Keine partiellen Abhängigkeiten
3NF	+ Keine transitiven Abhängigkeiten
BCNF	+ Jede Determinante ist Superschlüssel

## Systematisches Vorgehen:

### ① 1NF prüfen:

- Sind alle Werte atomar? (Keine Listen, keine Wiederholungsgruppen)

### ② Schlüssel bestimmen:

- Welche Attributkombination bestimmt alle anderen?

### ③ 2NF prüfen: (nur bei zusammengesetztem Schlüssel)

- Hängen alle Nicht-Schlüsselattribute vom *ganzen* Schlüssel ab?

### ④ 3NF prüfen:

- Gibt es FDs zwischen Nicht-Schlüsselattributen?
- Falls ja: transitive Abhängigkeit!

## Normalisierung hat auch Nachteile:

- Viele Tabellen → viele JOINS nötig
- JOINS können **Performance kosten**
- Komplexere Abfragen

## Denormalisierung = bewusste Einführung von Redundanz

### Wann sinnvoll?

- Lesende Zugriffe dominieren (wenig Updates)
- Performance kritisch (z.B. Reporting, Data Warehouse)
- Daten ändern sich selten

Wichtig

Denormalisierung ist eine **bewusste Entscheidung** nach Abwägung!  
Erst normalisieren, dann gezielt denormalisieren.

# Denormalisierung: Beispiele

## Typische Denormalisierungsstrategien:

### 1. Berechnete Spalten speichern:

- Statt:  $\text{SUM}(\text{Positionen})$  bei jeder Abfrage
- Speichern:  $\text{Bestellung.Gesamtsumme}$
- Vorteil: Schneller Zugriff
- Nachteil: Bei Änderung nachführen

### 2. Häufige JOINS vorwegnehmen:

- Statt: Bestellung JOIN Kunde
- Speichern: Kundenname in Bestellung
- Vorteil: Kein JOIN nötig
- Nachteil: Redundanz

## Data Warehouse

In analytischen Systemen (OLAP) ist Denormalisierung Standard!  
→ “Star Schema”, “Snowflake Schema”

# Wann welche Normalform?

Situation	Empfehlung	Grund
OLTP-System	3NF (mindestens)	Datenintegrität
Data Warehouse	1NF/2NF	Leseperformance
Reporting-DB	Denormalisiert	Schnelle Abfragen
Stammdaten	3NF/BCNF	Wenig Änderungen
Bewegungsdaten	3NF	Viele Änderungen

## Faustregel:

- **Viele Schreibzugriffe** → Hohe Normalform (3NF)
- **Viele Lesezugriffe** → Ggf. denormalisieren
- **Im Zweifel:** Erst normalisieren, dann messen, dann optimieren

# Hands-on

## 3NF-Normalisierung durchführen

marimo: 08-normalisierung.py

Aufgaben 8.4 – 8.5

# Agenda

- 1 Rückblick & Motivation
- 2 Funktionale Abhängigkeiten
- 3 Erste Normalform (1NF)
- 4 Zweite Normalform (2NF)
- 5 Dritte Normalform (3NF)
- 6 BCNF & Denormalisierung
- ▶ **7 Zusammenfassung**

## Funktionale Abhangigkeit:

- $A \rightarrow B = A$  bestimmt  $B$
- Voll vs. partiell
- Transitiv:  $A \rightarrow B \rightarrow C$

## Normalformen:

- 1NF: Atomare Werte
- 2NF: Keine partiellen Abhangigkeiten
- 3NF: Keine transitiven Abhangigkeiten

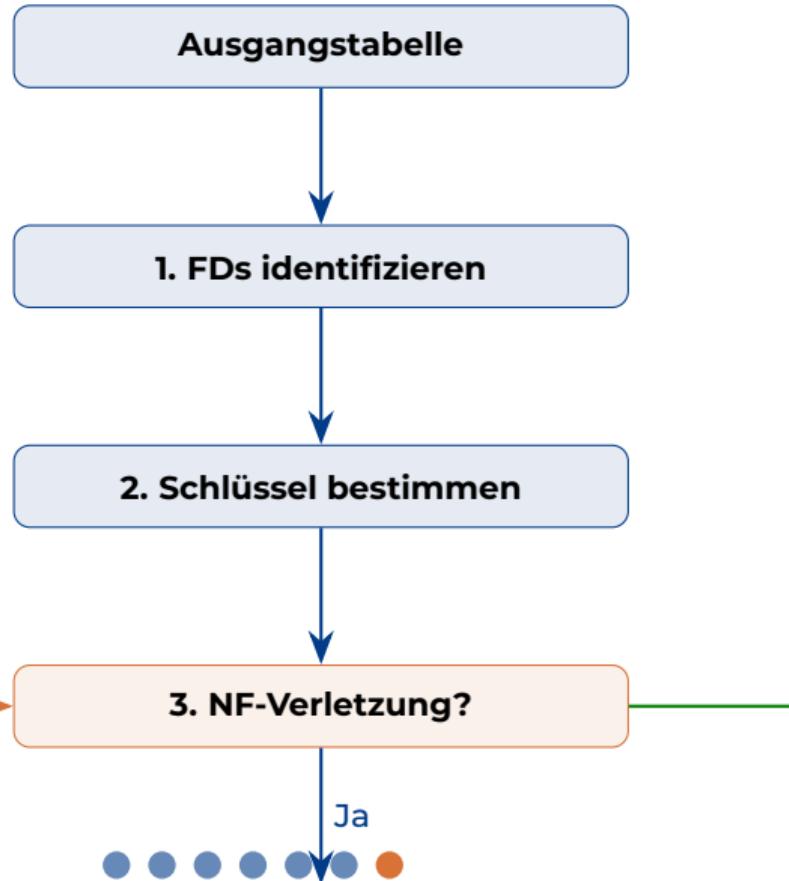
## Normalisierungsalgorismus:

- ① FDs identifizieren
- ② Schlssel bestimmen
- ③ Verletzungen finden
- ④ Tabelle zerlegen
- ⑤ Wiederholen bis 3NF

## Denormalisierung:

- Bewusst Redundanz einfuhren
- Fur Performance
- Nach Abwagung!

# Normalisierung: Der Algorithmus



Wiederholen

NF	Anforderung	Prüffrage	Lösung
1NF	Atomare Werte	Listen in Zellen? Wiederholungsgruppen?	Aufteilen in Zeilen/Tabellen
2NF	Voll abhängig vom PK	Hängt Attribut von Teil des Schlüssels ab?	Teil-FD auslagern
3NF	Keine transitive FD	FD zwischen Nicht-Schlüssel-Attributen?	Transitive FD auslagern
BCNF	Determinante = Superschlüssel	Determinante kein Schlüssel?	Auslagern

## Merksatz (Codd):

*"Jedes Attribut hängt ab vom Schlüssel, vom ganzen Schlüssel, und von nichts außer dem Schlüssel – so wahr mir Codd helfe!"*

## Vorlesung 9: Joins

- Daten aus mehreren Tabellen verknüpfen
- INNER JOIN, LEFT JOIN, RIGHT JOIN
- Self-Joins
- Join-Performance



**Jetzt haben wir gute Tabellen – in Vorlesung 9 lernen wir, sie wieder zusammenzuführen!**

# Fragen?

christoph.flath@uni-wuerzburg.de