

Sommersemester 2026

Datenmanagement & -analyse

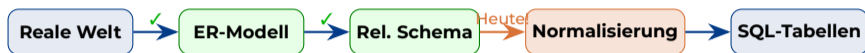
Prof. Dr. Christoph M. Flath

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**

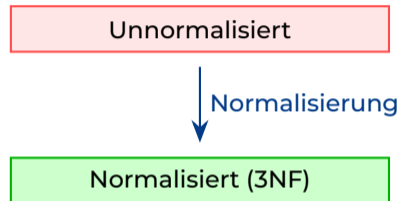
Motivation: Wann ist ein Schema “gut”?

Probleme bei schlechtem Design:

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

Ziel:

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



Normalisierung = systematisches Verfahren zur Beseitigung von Redundanzen

Bestellung_Unnorm

<u>Best_Nr</u>	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:

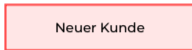
- “Müller, München” steht **dreimal** (Redundanz)
- “Laptop, 999” steht **zweimal** (Redundanz)
- Wenn Müller umzieht: **3 Zeilen** ändern (Änderungsanomalie)
- Neuer Kunde ohne Bestellung? **Nicht möglich** (Einfügeanomalie)

Änderungsanomalie:



Update vergessen →
Widerspruch in den Daten

Einfügeanomalie:



Bestellung = ?

NULL nicht erlaubt!

Neuer Kunde ohne
Bestellung nicht speicherbar

Löschanomalie:



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?

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

Definition

Ein Attribut B ist **funktional abhängig** von A (geschrieben: $A \rightarrow B$), wenn zu jedem Wert von A **genau ein** Wert von B gehört.

Beispiele:

- Matrikelnr \rightarrow Studentenname ✓
- PLZ \rightarrow Ort ✓ (in Deutschland)
- Ort \rightarrow PLZ ✗ (München 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**

Bestellung_Unnorm(Best_Nr, Kunde, K_Stadt, Produkt, P_Preis, Menge)

Funktionale Abhängigkeiten:

- Best_Nr, Produkt \rightarrow Menge (Schlüssel!)
- Best_Nr \rightarrow Kunde, K_Stadt (Bestellung \rightarrow Kunde)
- Kunde \rightarrow K_Stadt (Kunde \rightarrow Stadt)
- Produkt \rightarrow P_Preis (Produkt \rightarrow Preis)

Problem

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

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

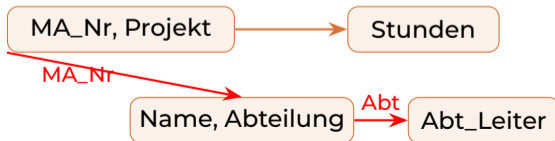
Welche FDs gibt es? (Ankreuzen)

- ☐ MA_Nr → Name
- ☐ MA_Nr → Abteilung
- ☐ Abteilung → Abt_Leiter
- ☐ MA_Nr, Projekt → Stunden
- ☐ Projekt → Stunden

Mitarbeiter(MA_Nr, Projekt, Name, Abteilung, Abt_Leiter, Stunden)

Funktionale Abhängigkeiten:

- ✓ $MA_Nr \rightarrow Name$ – Ein MA hat genau einen Namen
- ✓ $MA_Nr \rightarrow Abteilung$ – Ein MA ist in genau einer Abteilung
- ✓ $Abteilung \rightarrow Abt_Leiter$ – Jede Abteilung hat einen Leiter
- ✓ $MA_Nr, Projekt \rightarrow Stunden$ – Schlüssel-FD
- ✗ $Projekt \rightarrow Stunden$ – Falsch! Verschiedene MAs arbeiten unterschiedlich lange



Volle funktionale Abhängigkeit:

B ist **voll** funktional abhängig von A, wenn B von A abhängt, aber **nicht** von einer echten Teilmenge von A.

Beispiel:

Best_Nr, Produkt \rightarrow Menge
 \rightarrow Menge hängt vom **gesamten** Schlüssel ab.

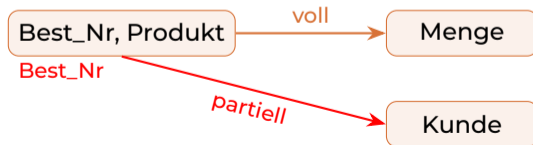
Partielle Abhängigkeit:

B hängt nur von einem **Teil** des Schlüssels ab.

Beispiel:

Best_Nr \rightarrow Kunde
 \rightarrow Kunde hängt nur von Best_Nr ab, nicht von Produkt.

Das verursacht Redundanz!



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:

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

Dekomposition:

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

Beispiel:

$\text{MA_Nr} \rightarrow \text{Name, Abteilung}$
 $\Rightarrow \text{MA_Nr} \rightarrow \text{Name}$ und $\text{MA_Nr} \rightarrow \text{Abteilung}$

Pseudotransitivität

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

Gegeben:

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

Welche FDs können abgeleitet werden?

FD	Ableitbar?	Begründung
$A \rightarrow C$		

Gegeben:

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

Welche FDs können abgeleitet werden?

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

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.: $A \rightarrow C, CD \rightarrow E$
$B \rightarrow E$		

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

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 Abhängigkeiten erkennen

marimo: 08-normalisierung.py

Aufgabe 8.1: FDs aus Beispieldaten ableiten

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

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):

<u>Kurs_ID</u>	Kursname	Dozenten
DMA01	Datenmanagement	Flath, Müller
BWL01	Einführung BWL	Schmidt

Schritt 1: Wiederholende Werte identifizieren → “Dozenten”

Schritt 2: Aufteilen in atomare Werte:

<u>Kurs_ID</u>	Kursname	<u>Dozent</u>
DMA01	Datenmanagement	Flath
DMA01	Datenmanagement	Müller
BWL01	Einführung BWL	Schmidt

Hinweis: Der Schlüssel hat sich geändert! Jetzt: (Kurs_ID, Dozent)

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:

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:

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

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

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!



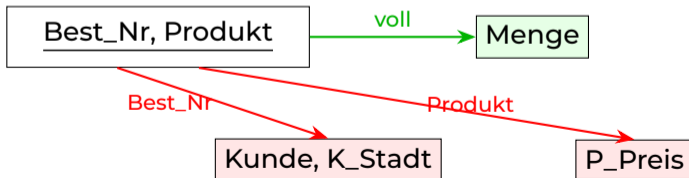
Wichtig

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!**



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.

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)

Gegeben: Buchung(Hotel_ID, Gast_ID, Datum, Zimmer, Hotel_Name, Gast_Name)

FDs:

- Hotel_ID, Gast_ID, Datum \rightarrow Zimmer
- Hotel_ID \rightarrow Hotel_Name
- Gast_ID \rightarrow 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

Gegeben: Buchung(Hotel_ID, Gast_ID, Datum, Zimmer, Hotel_Name, Gast_Name)

FDs:

- Hotel_ID, Gast_ID, Datum \rightarrow Zimmer
- Hotel_ID \rightarrow Hotel_Name
- Gast_ID \rightarrow 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!

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

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 Abhängigkeiten:

- $\text{Best_Nr} \rightarrow \text{Kunde}$ ✓
- $\text{Kunde} \rightarrow \text{K_Stadt}$ (Kunde bestimmt Stadt)
- $\Rightarrow \text{Best_Nr} \rightarrow \text{K_Stadt}$ ✗ **transitiv!**



Redundanz: Wenn Müller 5 Bestellungen hat, steht “München” 5x in der Tabelle.

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

- MA_Nr \rightarrow Abt_ID – direkt
- Abt_ID \rightarrow Abt_Name – nicht vom PK!
- Abt_ID \rightarrow Abt_Ort – nicht vom PK!
- \Rightarrow MA_Nr \rightarrow Abt_Name – **transitiv!**
- \Rightarrow MA_Nr \rightarrow 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)

Ausgangstabelle:

Einschreibung(Matrikel, Student_Name, Kurs_ID, Kurs_Name, Dozent_ID, Dozent_Name, Note)

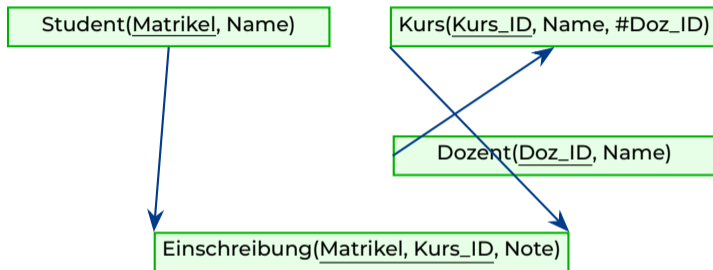
FDs:

- Matrikel \rightarrow Student_Name
- Kurs_ID \rightarrow Kurs_Name, Dozent_ID
- Dozent_ID \rightarrow Dozent_Name
- Matrikel, Kurs_ID \rightarrow 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):

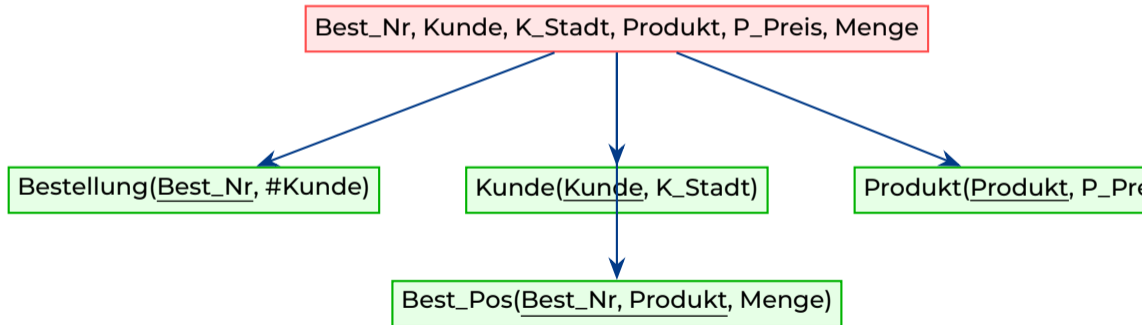


Vorteile:

- 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!

Bewerten Sie jede Relation:

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

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)	

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)	

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)	

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> , <u>Artikel</u> , Preis, Menge, Kunde_Name)	nicht 2NF
Mitarbeiter(<u>MA_ID</u> , Name, Gehalt)	3NF

Erklärungen:

- Film: Genre \rightarrow 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

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?

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

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:

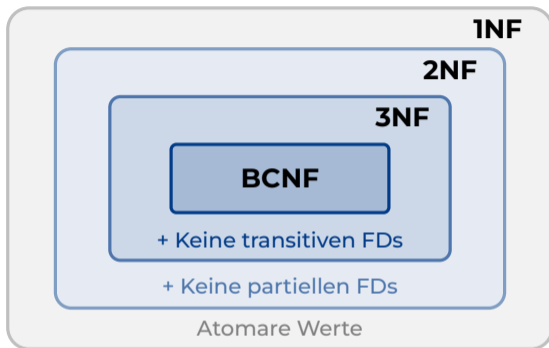
- Student, Fach \rightarrow Dozent – Schlüssel-FD
- Dozent \rightarrow Fach – Jeder Dozent unterrichtet nur ein Fach

3NF-Prüfung:

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

BCNF-Prüfung:

- Dozent \rightarrow Fach – Dozent ist kein Superschlüssel!
- \Rightarrow ✗ Ist NICHT in BCNF!



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.

Typische Denormalisierungsstrategien:

1. Berechnete Spalten speichern:

- Statt: SUM(Positionen) bei jeder Abfrage
- Speichern: 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"

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

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

Funktionale Abhängigkeit:

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

Normalformen:

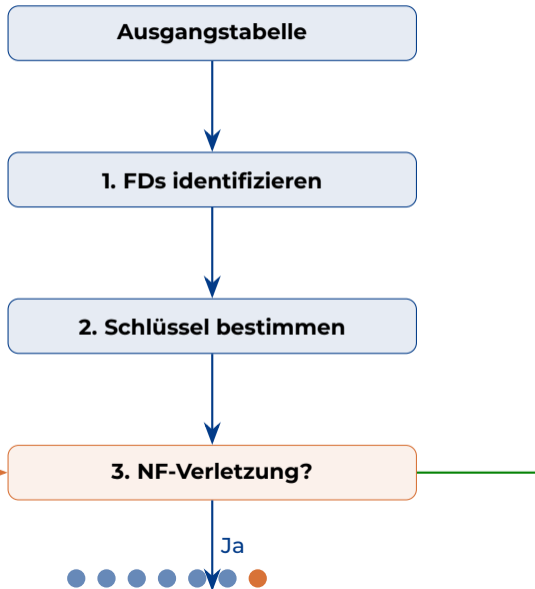
- 1NF: Atomare Werte
- 2NF: Keine partiellen Abhängigkeiten
- 3NF: Keine transitiven Abhängigkeiten

Normalisierungsalgorithmus:

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

Denormalisierung:

- Bewusst Redundanz einführen
- Für Performance
- Nach Abwägung!



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 = Super-schlüssel	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