DATA DRIVEN DECISIONS

Sommersemester 2026

Datenmanagement & -analyse

Prof. Dr. Christoph M. Flath
Lehrstuhl fuer Wirtschaftsinformatik und Business Analytics

Julius-Maximilians-Universitaet Wuerzburg

Agenda

Rueckblick: Der Weg zu SQL-Tabellen

Letzte Session:

- ER-Modell → Relationales Schema
- Primaer- und Fremdschluessel
- CREATE TABLE mit Constraints

Diese Session:

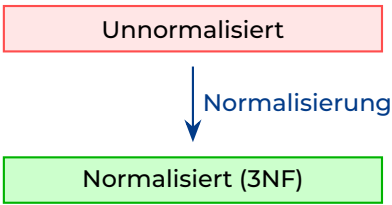
- Wann ist ein Schema “gut”?
- Systematische Qualitaetspruefung: **Normalformen**

Motivation: Wann ist ein Schema “gut”?

Probleme bei schlechtem Design:

- **Redundanz** – Daten mehrfach gespeichert
- **Aenderungsanomalie** – Inkonsistenz bei Updates
- **Einfuegeanomalie** – Fehlende Daten blockieren
- **Loeschanomalie** – Ungewollter Datenverlust

Ziel:
Schema so gestalten, dass diese Probleme **nicht auftreten koennen**.



Normalisierung = systematisches Verfahren zur Beseitigung von Redundanzen

Beispiel: Unnormalisierte Tabelle

Bestellung_Unnorm

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

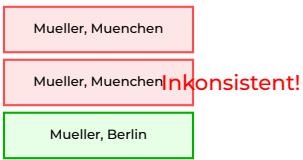
Probleme:

- “Mueller, Muenchen” steht **dreimal** (Redundanz)
- “Laptop, 999” steht **zweimal** (Redundanz)
- Wenn Mueller umzieht: **3 Zeilen** aendern (Aenderungsanomalie)
- Neuer Kunde ohne Bestellung? **Nicht moeglich** (Einfuegeanomalie)

Die drei Anomalien im Detail

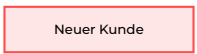
DATA DRIVEN DECISIONS

Aenderungsanomalie:



Update vergessen → Widerspruch in den Daten

Einfuegeanomalie:



Bestellung = ?

NULL nicht erlaubt!

Neuer Kunde ohne Bestellung nicht speicherbar

Loeschanomalie:

von Schmidt

Kundendaten weg!

Bestellung loeschen → Kundendaten verloren

Kernproblem

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

Weiteres Beispiel: Bibliothekssystem

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	Mueller	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 loeschen?

Agenda

Funktionale Abhaengigkeit (FD)

DATA DRIVEN DECISIONS

Definition

Ein Attribut B ist **funktional abhaengig** von A (geschrieben: A → B), wenn zu jedem Wert von A **genau ein** Wert von B gehoert.

Beispiele:

- Matrikelnr → Studentenname ✓
- PLZ → Ort ✓ (in Deutschland)
- Ort → PLZ ✗ (Muenchen hat viele PLZ)
- ISBN → Buchtitel ✓

“Wenn ich A kenne, kenne ich auch B.”

Verschiedene Notationen (alle aequivalent):

Schreibweise	Bedeutung
$A \rightarrow B$	A bestimmt B
$A \rightarrow B$	A bestimmt B
B ist FD von A	B haengt 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 Abhaengigkeit
- Die linke Seite heisst **Determinante**

FDs in unserem Beispiel

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

Funktionale Abhaengigkeiten:

- Best_Nr, Produkt → Menge (Schluessel!)
- 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 Schluessel** abhaengen!
Das ist die Ursache fuer Redundanz.

Uebung: FDs identifizieren

DATA DRIVEN DECISIONS

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

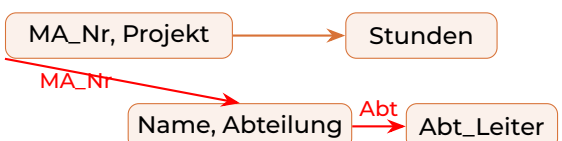
Welche FDs gibt es? (Ankreuzen)

- ☐ 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 Abhaengigkeiten:

- ✓ 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 – Schluessel-FD
- ✗ Projekt → Stunden – Falsch! Verschiedene MAs arbeiten unterschiedlich lange



Volle vs. Partielle Abhaengigkeit

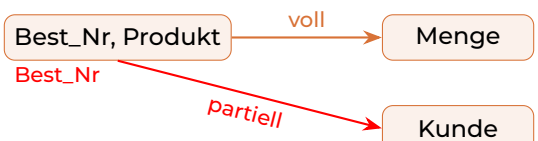
Volle funktionale Abhaengigkeit:
B ist **voll** funktional abhaengig von A, wenn B von A abhaengt, aber **nicht** von einer echten Teilmenge von A.

Beispiel:
Best_Nr, Produkt → Menge
→ Menge haengt vom **gesamten** Schluessel ab.

Partielle Abhaengigkeit:
B haengt nur von einem **Teil** des Schluessels ab.

Beispiel:
Best_Nr → Kunde
→ Kunde haengt nur von Best_Nr ab, nicht von Produkt.

Das verursacht Redundanz!



Regeln zur Herleitung von FDs:

1. Reflexivitaet

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

Beispiel: Vorname, Nachname → Nachname

2. Augmentation (Erweiterung)

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

Beispiel: Aus MA_Nr → Name folgt MA_Nr, Projekt → Name, Projekt

3. Transitivitaet

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

Beispiel: MA_Nr → Abteilung und Abteilung → Leiter ⇒ MA_Nr → Leiter

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

Abgeleitete Regeln aus Armstrong's Axiomen

DATA DRIVEN DECISIONS

Nuetzliche Zusatzregeln (ableitbar aus den Axiomen):

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

Beispiel:
ISBN → Titel und ISBN → Autor
⇒ ISBN → Titel, Autor

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

Beispiel:
MA_Nr → Name, Abteilung
⇒ MA_Nr → Name und MA_Nr → Abteilung

Pseudotransitivitaet

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 koennen abgeleitet werden?

FD	Ableitbar?	Begruendung
$A \rightarrow C$		

Quiz: Welche FDs sind ableitbar?

Gegeben:

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

Welche FDs koennen abgeleitet werden?

FD	Ableitbar?	Begruendung
$A \rightarrow C$	✓	Transitivitaet: A→B→C
$A, D \rightarrow E$		

Quiz: Welche FDs sind ableitbar?

Gegeben:

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

Welche FDs koennen abgeleitet werden?

FD	Ableitbar?	Begruendung
$A \rightarrow C$	✓	Transitivitaet: A→B→C
$A, D \rightarrow E$	✓	Pseudotrans.: A→C, CD→E
$B \rightarrow E$		

Quiz: Welche FDs sind ableitbar?

Gegeben:

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

Welche FDs koennen abgeleitet werden?

FD	Ableitbar?	Begruendung
$A \rightarrow C$	✓	Transitivitaet: A→B→C
$A, D \rightarrow E$	✓	Pseudotrans.: A→C, CD→E
$B \rightarrow E$	✗	D fehlt fuer C,D→E
$A \rightarrow B, C$		

Quiz: Welche FDs sind ableitbar?

Gegeben:

- A → B
- B → C
- C, D → E

Welche FDs koennen abgeleitet werden?

FD	Ableitbar?	Begruendung
A → C	✓	Transitivitaet: A→B→C
A, D → E	✓	Pseudotrans.: A→C, CD→E
B → E	✗	D fehlt fuer C,D→E
A → B, C	✓	Vereinigung: A→B, A→C

Hands-on

Funktionale Abhaengigkeiten erkennen

marimo: 08-normalisierung.py

Aufgabe 8.1: FDs aus Beispieldaten ableiten

Agenda

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" enthaelt eine **Liste**

In 1NF:

Student	Kurs
Anna	DMA
Anna	BWL
Anna	Statistik
Ben	DMA

✓ Jede Zelle enthaelt **einen** Wert

Merke

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

1NF-Verletzungen erkennen

Typische Anzeichen:

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

Loesung:

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, Mueller
BWL01	Einfuehrung BWL	Schmidt

Schritt 1: Wiederholende Werte identifizieren → “Dozenten”

Schritt 2: Aufteilen in atomare Werte:

Kurs_ID	Kursname	Dozent
DMA01	Datenmanagement	Flath
DMA01	Datenmanagement	Mueller
BWL01	Einfuehrung BWL	Schmidt

Hinweis: Der Schluessel hat sich geaendert! Jetzt: (Kurs_ID, Dozent)

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:

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

Agenda

Definition: 2NF

Eine Relation ist in **2NF**, wenn sie in 1NF ist und jedes Nicht-Schluesselattribut **voll funktional abhaengig** vom **gesamten** Primaerschluessel ist.

Anders gesagt: Keine partiellen Abhaengigkeiten!

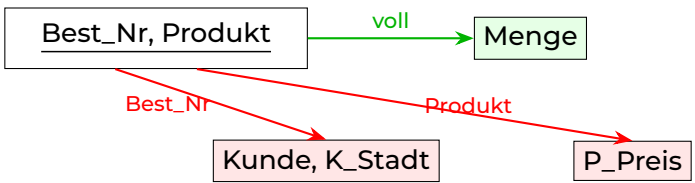
Wichtig

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

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

Funktionale Abhaengigkeiten:

- Best_Nr, Produkt → Menge ✓ voll abhaengig
- Best_Nr → Kunde, K_Stadt ✗ **partiell!**
- Produkt → P_Preis ✗ **partiell!**



2NF herstellen: Zerlegung

DATA DRIVEN DECISIONS

Loesung: Attribute, die nur von einem Teil des Schluessels abhaengen, 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 Schluessel abhaengen.

Ausgangstabelle: Vorlesung(VL_Nr, Dozent_ID, VL_Name, Raum, Dozent_Name)

Schritt 1: FDs identifizieren

- VL_Nr, Dozent_ID → Raum – voll (Dozent kann unterschiedliche Raeume haben)
- VL_Nr → VL_Name – **partiell!**
- Dozent_ID → Dozent_Name – **partiell!**

Schritt 2: Partielle FDs auslagern

- Neue Tabelle fuer VL_Nr → VL_Name
- Neue Tabelle fuer 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) Primaerschluessel gibt es **keine partiellen Abhaengigkeiten**.

Beispiel:

Kunde(<u>Kunden_ID</u>, Name, Stadt, PLZ)

- Kunden_ID → Name – voll (Schluessel hat nur 1 Attribut)
- Kunden_ID → Stadt – voll
- Kunden_ID → PLZ – voll

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

Aber: Es koennte trotzdem eine 3NF-Verletzung geben (transitive Abhaengigkeit: 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?

DATA DRIVEN DECISIONS

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 Abhaengigkeiten vorhanden (Hotel_Name, Gast_Name)

Hands-on

1NF und 2NF anwenden

marimo: 08-normalisierung.py

Aufgaben 8.2 – 8.3

Pause

15 Minuten

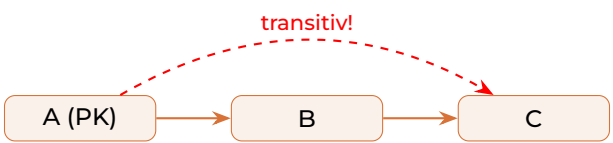
Kaffee holen!

DATA DRIVEN DECISIONS

Definition: 3NF

Eine Relation ist in **3NF**, wenn sie in 2NF ist und kein Nicht-Schluesselattribut **transitiv** vom Primaerschluessel abhaengt.

Transitive Abhaengigkeit:
$A \rightarrow B$ und $B \rightarrow C \Rightarrow C$ haengt **transitiv** von A ab.



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

Bestellung(Best_Nr, Kunde, K_Stadt)

Funktionale Abhaengigkeiten:

- Best_Nr → Kunde ✓
- Kunde → K_Stadt (Kunde bestimmt Stadt)
- ⇒ Best_Nr → K_Stadt ✗ **transitiv!**

Redundanz: Wenn Mueller 5 Bestellungen hat, steht “Muenchen” 5x in der Tabelle.

3NF herstellen: Zerlegung

Loesung: Die transitive Abhaengigkeit 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-Schluesselattribut haengt **direkt** (nicht transitiv) vom Schluessel ab.

Ausgangstabelle (2NF): Mitarbeiter(MA_Nr, Name, Abt_ID, Abt_Name, Abt_Ort)

Schritt 1: Transitive FDs finden

- MA_Nr → Abt_ID – direkt
- Abt_ID → Abt_Name – nicht vom PK!
- Abt_ID → Abt_Ort – nicht vom PK!
- ⇒ MA_Nr → Abt_Name – **transitiv!**
- ⇒ MA_Nr → 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: Universitaetssystem

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

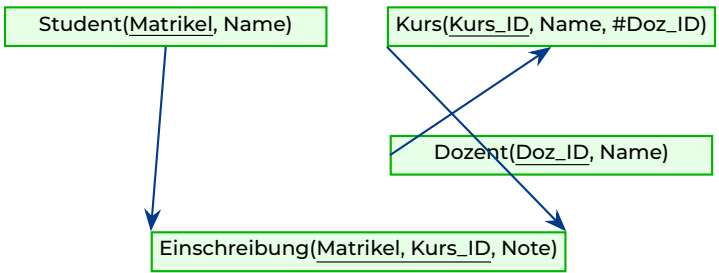
FDs:

- Matrikel → Student_Name
- Kurs_ID → Kurs_Name, Dozent_ID
- Dozent_ID → Dozent_Name
- Matrikel, Kurs_ID → Note

Probleme:

- Partielle Abhaengigkeit: Student_Name von Matrikel (nicht 2NF)
- Partielle Abhaengigkeit: Kurs_Name, Dozent_ID von Kurs_ID (nicht 2NF)
- Transitive Abhaengigkeit: Dozent_Name ueber Dozent_ID (nicht 3NF)

Normalisierte Tabellen (3NF):

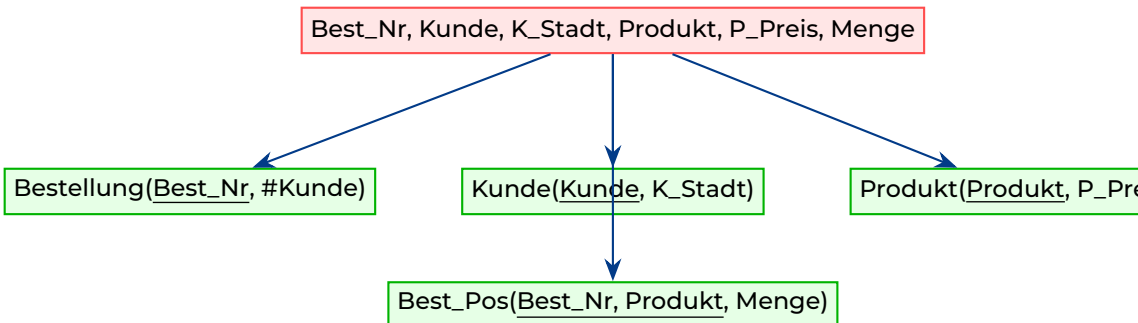


Vorteile:

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

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(ID, Titel, Regisseur, Genre, Genre_Beschreibung)	

Quiz: Welche Normalform?

Bewerten Sie jede Relation:

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

Quiz: Welche Normalform?

DATA DRIVEN DECISIONS

Bewerten Sie jede Relation:

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

Quiz: Welche Normalform?

Bewerten Sie jede Relation:

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

Quiz: Welche Normalform?

Bewerten Sie jede Relation:

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

Erklaerungen:

- Film: Genre → Genre_Beschreibung ist transitiv
- Buch: Wiederholungsgruppe (mehrere Autor-Spalten)
- Bestellung: Preis haengt nur von Artikel ab (partiell)
- Mitarbeiter: Alle Attribute haengen direkt vom Schluessel ab

Haeufige Fehler bei der Normalisierung

DATA DRIVEN DECISIONS

Fehler 1: FDs uebersehen

- Alle Geschaeftsregeln beachten!
- “Jeder Kunde hat genau eine Adresse” → FD

Fehler 2: Schluessel falsch bestimmt

- Schluessel muss minimal sein
- Alle Attribute bestimmen

Fehler 3: Zu frueh aufhoeren

- 2NF reicht nicht!
- Transitive FDs pruefen

Fehler 4: Zu viel zerlegen

- Nur bei echten Verletzungen zerlegen
- JOINs kosten Performance

Tipp

Bei Unsicherheit: Beispieldaten durchspielen! Tritt Redundanz auf?

Agenda

Boyce-Codd-Normalform (BCNF)

DATA DRIVEN DECISIONS

Definition: BCNF

Eine Relation ist in **BCNF**, wenn fuer jede nicht-triviale FD X → Y gilt: X ist ein **Superschluessel**.

BCNF ist strenger als 3NF:

- 3NF erlaubt: Schluesselattribut haengt von Nicht-Schluessel ab
- BCNF verbietet das

In der Praxis:

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

Merksatz

"Jedes Attribut haengt vom Schluessel ab, vom ganzen Schluessel, und von nichts ausser dem Schluessel." (Codd)

3NF vs. BCNF: Der Unterschied

DATA DRIVEN DECISIONS

Beispiel: Kurs(Student, Fach, Dozent)

FDs:

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

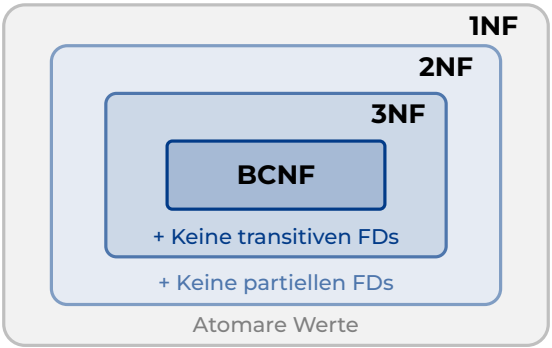
3NF-Pruefung:

- Keine partiellen Abhaengigkeiten (Dozent haengt vom ganzen Schluessel ab)
- Keine transitiven Abhaengigkeiten (Dozent ist kein Nicht-Schluesselattribut in der zweiten FD)
- ⇒ ✓ Ist in 3NF!

BCNF-Pruefung:

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

Normalformen: Uebersicht



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

Systematisches Vorgehen:

1. **1NF pruefen:**
   - Sind alle Werte atomar? (Keine Listen, keine Wiederholungsgruppen)
2. **Schluessel bestimmen:**
   - Welche Attributkombination bestimmt alle anderen?
3. **2NF pruefen:** (nur bei zusammengesetztem Schluessel)
   - Haengen alle Nicht-Schluesselattribute vom *ganzen* Schluessel ab?
4. **3NF pruefen:**
   - Gibt es FDs zwischen Nicht-Schluesselattributen?
   - Falls ja: transitive Abhaengigkeit!

Denormalisierung: Wann und Warum?

DATA DRIVEN DECISIONS

Normalisierung hat auch Nachteile:

- Viele Tabellen → viele JOINs noetig
- JOINs koennen **Performance kosten**
- Komplexere Abfragen

Denormalisierung = bewusste Einfuehrung von Redundanz

Wann sinnvoll?

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

Wichtig

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

Denormalisierung: Beispiele

DATA DRIVEN DECISIONS

Typische Denormalisierungsstrategien:

1. Berechnete Spalten speichern:

- Statt: SUM(Positionen) bei jeder Abfrage
- Speichern: Bestellung.Gesamtsumme
- Vorteil: Schneller Zugriff
- Nachteil: Bei Aenderung nachfuehren

2. Haeufige JOINs vorwegnehmen:

- Statt: Bestellung JOIN Kunde
- Speichern: Kundenname in Bestellung
- Vorteil: Kein JOIN noetig
- 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)	Datenintegritaet
Data Warehouse	1NF/2NF	Leseperformance
Reporting-DB	Denormalisiert	Schnelle Abfragen
Stammdaten	3NF/BCNF	Wenig Aenderungen
Bewegungsdaten	3NF	Viele Aenderungen

Faustregel:

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

Hands-on

3NF-Normalisierung durchfuehren

marimo: 08-normalisierung.py

Aufgaben 8.4 – 8.5

Zusammenfassung

Funktionale Abhaengigkeit:

- A → B = A bestimmt B
- Voll vs. partiell
- Transitiv: A→B→C

Normalformen:

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

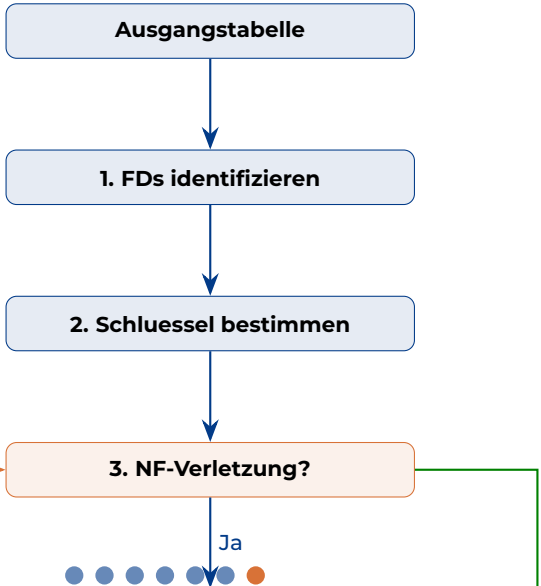
Normalisierungsalgorithmus:

1. FDs identifizieren
2. Schluessel bestimmen
3. Verletzungen finden
4. Tabelle zerlegen
5. Wiederholen bis 3NF

Denormalisierung:

- Bewusst Redundanz einfuehren
- Fuer Performance
- Nach Abwaegung!

Normalisierung: Der Algorithmus



NF	Anforderung	Prueffrage	Loesung
1NF	Atomare Werte	Listen in Zellen? Wiederholungsgruppen?	Aufteilen in Zeilen/Tabellen
2NF	Voll abhaengig vom PK	Haengt Attribut von Teil des Schluessels ab?	Teil-FD auslagern
3NF	Keine transitive FD	FD zwischen Nicht-Schluessel-Attributen?	Transitive FD auslagern
BCNF	Determinante = Superschluessel	Determinante kein Schluessel?	Auslagern

Merksatz (Codd):

> *"Jedes Attribut haengt ab vom Schluessel, vom ganzen Schluessel, und von nichts ausser dem Schluessel – so wahr mir Codd helfe!"*

Vorlesung 9: Joins

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

Jetzt haben wir gute Tabellen – in Session 9 lernen wir, sie wieder zusammenzufuehren!

Fragen?

christoph.flath@uni-wuerzburg.de