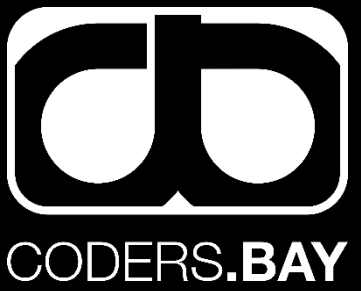
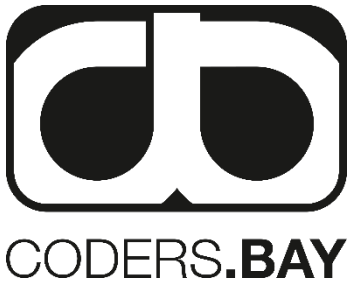
DIE WELT DER DATENBANKEN



WIEDERHOLUNG

NORMALFORMEN

ZIEL:
ANOMALIEN VERMEIDEN

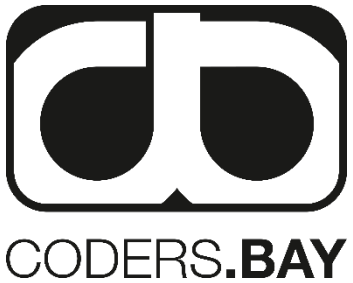


- **Änderungsanomalie:** **Bsp. Sokrates zieht um**
- **Einfügeanomalie:** Bsp. Curie ist neu und liest noch keine Vorlesung
- **Löschanomalie:** Bsp. “Die 3 Kritiken” fällt weg.

Professoren

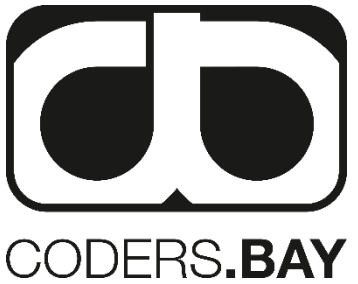
PersNr	Name	Rang	Raum	VorlNr	Titel	SWS
2125	Sokrates	C4	226	5041	Ethik	4
2125	Sokrates	C4	226	5049	Mäeutik	2
2125	Sokrates	C4	226	4052	Logik	4
...	...	...	...	...	...	...
2133	Popper	C3	52	5295	Der Wiener Kreis	2
2137	Kant	C4	7	4630	Die 3 Kritiken	4

NORMALFORMEN



- Legen **Eigenschaften** von Relationsschemata fest
- **Verbieten** bestimmte **Kombinationen** in Relationen
- Sollen Redundanzen und Anomalien vermeiden

NORMALFORMEN
ERSTE NORMALFORM



- Erlaubt nur **atomare Attribute** in den Relationsschemata. D.h. Attributwerte sind Elemente von **Standard-Datentypen** wie *integer* oder *string*, aber keine Mengenwerte wie *array* oder *set*

Nicht in 1NF:

Eltern

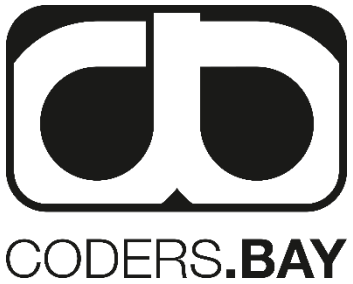
Vater	Mutter	Kinder
Johann	Martha	{Else, Lucie}
Heinz	Martha	{Cleo}
...	...	...

in 1NF (= flache Relation)

Eltern

Vater	Mutter	Kinder
Johann	Martha	Else
Johann	Martha	Lucie
Heinz	Martha	Cleo

NORMALFORMEN
ZWEITE NORMALFORM



- **Partielle Abhängigkeit** liegt vor, wenn ein Attribut funktional nur von einem Teil des Schlüssel abhängt.

- Verstoß gegen 2NF deutet darauf hin, dass in der Relation Informationen über mehr als ein Konzept modelliert werden.

- Zweite Normalform eliminiert **partielle Abhängigkeiten** bei Nichtschlüsselattributen.

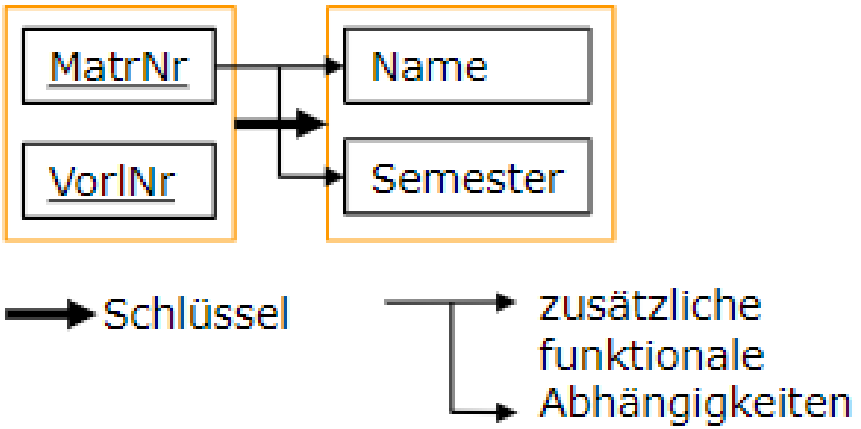
NORMALFORMEN
ZWEITE NORMALFORM
(NEGATIVBEISPIEL)

CODERS.BAY

StudentenBelegung

MatrNr	VorlNr	Name	Semester
26120	5001	Fichte	10
27550	5001	Schopenhauer	6
27550	4052	Schopenhauer	6
28106	5041	Carnap	3
28106	5052	Carnap	3
28106	5216	Carnap	3
28106	5259	Carnap	3
...	...		...

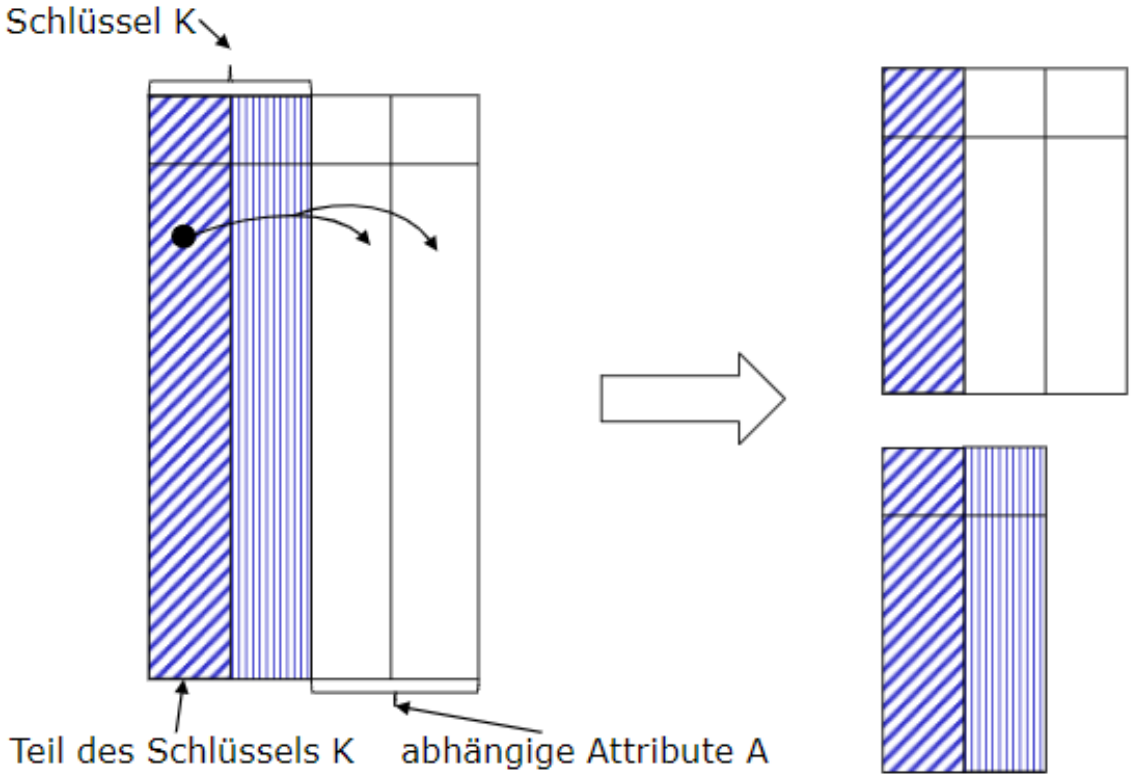
- {MatrNr} → {Name} und
- {MatrNr} → {Semster}



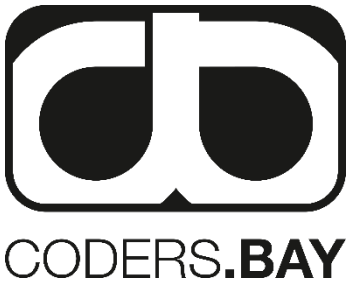
NORMALFORMEN
ZWEITE NORMALFORM

CODERS.BAY

- Eliminierung partieller Abhängigkeiten



NORMALFORMEN
ZWEITE NORMALFORM



- Eliminierung partieller Abhängigkeiten

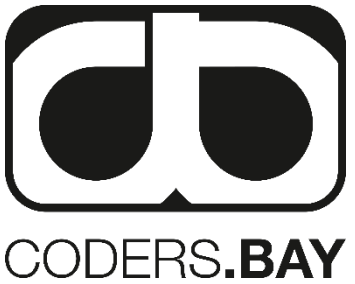
Relation in 2NF:

StudentenBelegung: {MatrNr, VorlNr, Name, Semester}

hören: {MatrNr, VorlNr}

Studenten: {MatrNr, Name, Semester}

NORMALFORMEN
DRITTE NORMALFORM



- Eliminiert (zusätzlich) transitive Abhängigkeiten

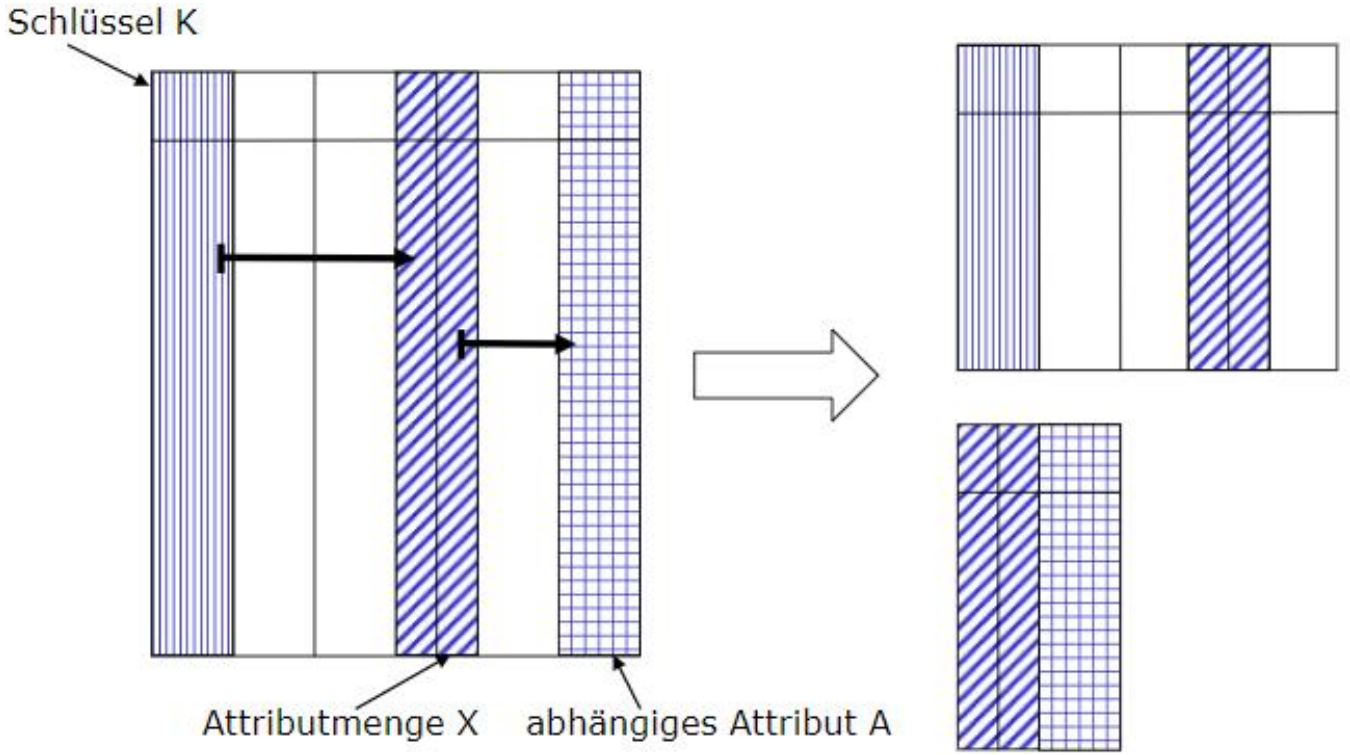
- Beispiel:
  - R = {PersNr, Name, Raum, Rang, PLZ, Ort, Straße}
  - {PersNr} → {PLZ} und {PLZ} → {Ort}

- Man beachte: 3.NF betrachtet **nur** Nichtschlüsselattribute als Endpunkt transitiver Abhängigkeiten.

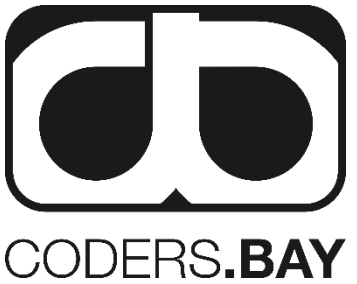
NORMALFORMEN
DRITTE NORMALFORM

CODERS.BAY

- Eliminierung transitiver Abhängigkeiten durch Verschiebung transitiv abhängiger Attribute in ein neues Relationenschema.



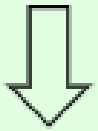
NORMALFORMEN
DRITTE NORMALFORM



- Eliminierung transitiver Abhängigkeiten

Relation in 3NF:

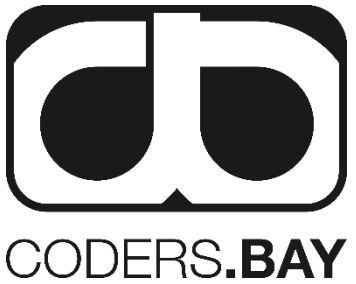
Professoren: {ProfNr, Name, Raum, Rang, PLZ, Ort, Straße}



Professoren: {ProfNr, Name, Raum, Rang, PLZ, Straße}

Orte: {PLZ, Ort}

NORMALFORMEN

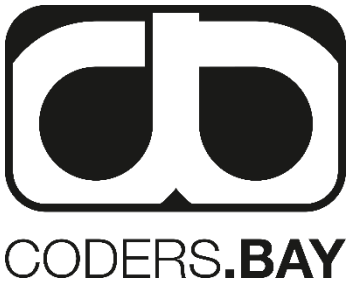


- **1NF:** Ein Relationenschema ist in 1. Normalform, wenn dessen Wertebereiche atomar sind.

- **2NF:** Ein Relationenschema ist in 2. Normalform, wenn es in 1. Normalform ist und jedes Nichtschlüsselattribut voll funktional vom Primärschlüssel abhängig ist.

- **3NF:** Ein Relationenschema ist in 3. Normalform, wenn es sich in 2. Normalform befindet, und kein Nichtschlüsselattribut vom Primärschlüssel transitiv abhängig ist.

SCHLÜSSEL VON 1:N-BEZIEHUNGEN



- Relationen **mit gleichem Schlüssel** kann man zusammenfassen. Aber nur diese. Und keine anderen!

Initialentwurf:

Professoren: {[PersNr:integer, Name:string, Rang:string, Raum:integer]}
Vorlesungen: {[VorlNr:integer, Titel:string, SWS:integer]}
lesen: {[PersNr:integer, VorlNr:integer]} (1:N)

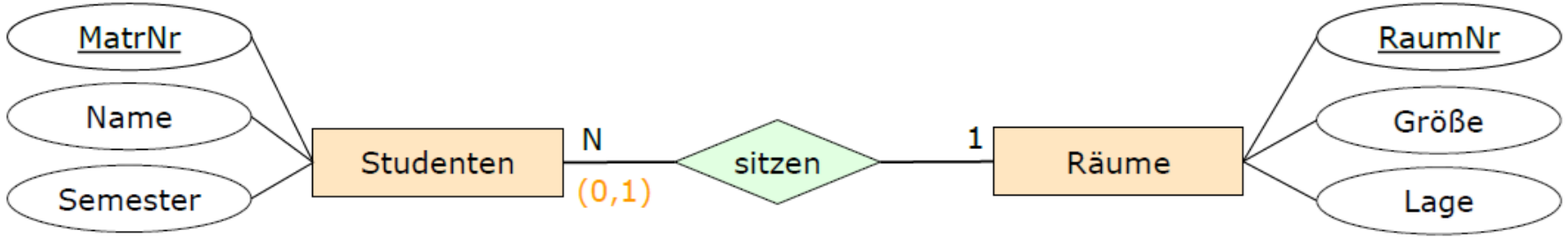
Verfeinerung durch Zusammenfassung von Relationen:

Professoren: {[PersNr:integer, Name:string, Rang:string, Raum:integer]}
Vorlesungen: {[VorlNr:integer, Titel:string, SWS:integer, gelesenVon:integer]}

NULL - WERTE VERMEIDEN (1:N)

CODERS.BAY

- **Beispiel:** Studierende, die als Assistenten arbeiten, bekommen einen Arbeitsraum. Es gibt 25.000 Studierende und 200 davon sind Assistenten



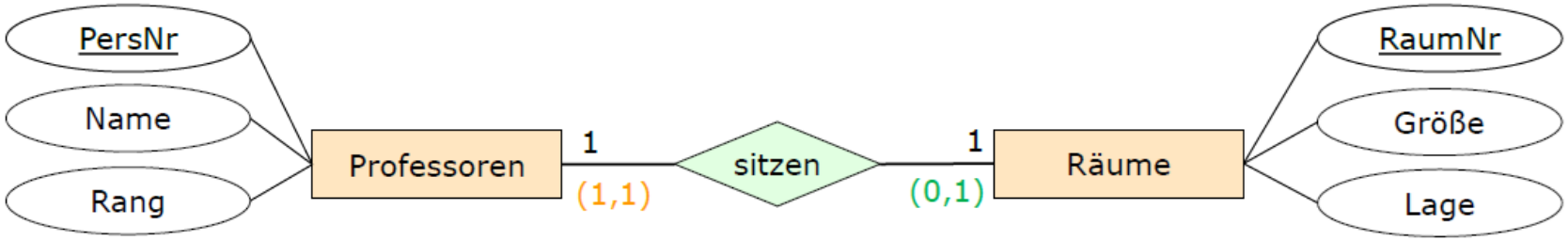
Logischer Entwurf:

Räume: {[RaumNr:integer, Größe:decimal, Lage:string]}

~~Studenten: {[MatrNr:integer, Name:string, Semester:integer, RaumNr:integer]}~~

- Hier **nicht** zusammenfassen! NULL-Werte vermeiden!

NULL-WERTE VERMEIDEN (1:1)



Logischer Entwurf:

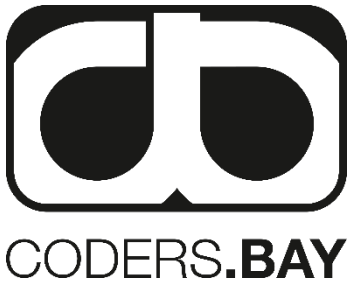
Professoren: {[PersNr:integer, Name:string, Rang:string]}
Räume: {[RaumNr:integer, Größe:decimal, Lage:string]}
sitzen: {[PersNr:integer, RaumNr:integer]} oder
sitzen: {[PersNr:integer, RaumNr:integer]}

Professoren: {[PersNr:integer, Name:string, Rang:string, RaumNr:integer]}
Räume: {[RaumNr:integer, Größe:decimal, Lage:string]}

Professoren: {[PersNr:integer, Name:string, Rang:string]}
Räume: {[RaumNr:integer, Größe:decimal, Lage:string, PersNr:integer]}

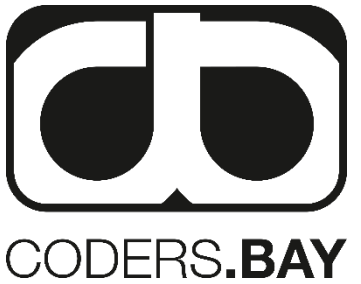
EMPLOYEE DATENBANK

ANFORDERUNGEN AN DIE EMPLOYEE DATABASE



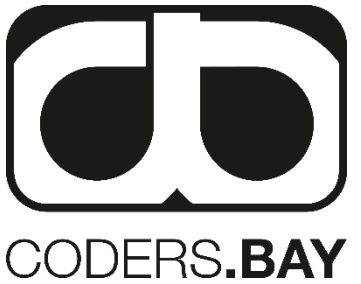
- Folgendes soll zu den Mitarbeitern gespeichert werden:
  - First_name
  - Last_name
  - Email
  - Phone_number
  - Hire_date
  - Salary
  - Commission_pct

ANFORDERUNGEN AN DIE EMPLOYEE DATABASE



- Jeder Mitarbeiter hat einen Manager, der selbst wieder ein Mitarbeiter ist
- Jeder Mitarbeiter arbeitet in einer Abteilung
- Jeder Mitarbeiter hat einen job_title (zu diesem werden wiederrum min_salary, max_salary gespeichert)
- Für jeden Mitarbeiter wird eine Job History geführt (start_date, end_date) – dabei soll hervorgehen in welcher Position (job_title) und welcher Abteilung der Mitarbeiter gearbeitet hat

ANFORDERUNGEN AN DIE EMPLOYEE DATABASE



- Eine Abteilung hat einen Manager und einen Namen und befindet sich an einem Standort
- Ein Standort liegt in einem Land
- Das Land ist einer bestimmten Region zugeordnet

ENDE

CODERS.BAY