

Skript zur Vorlesung:

Einführung in die Informatik: Systeme und Anwendungen

Sommersemester 2014

Kapitel 3: Datenbanksysteme

Vorlesung: Prof. Dr. Christian Böhm

Übungen: Sebastian Goebel

Skript © 2004 Christian Böhm, Peer Kröger

<http://www.dbs.ifi.lmu.de/Lehre/InfoNF>



Überblick

3.1 Einleitung

3.2 Das Relationale Modell

3.3 Die Relationale Algebra

3.4 Mehr zu SQL

3.5 Das E/R-Modell

3.6 Normalformen

Relationaler Datenbank-Entwurf

- Schrittweises Vorgehen:
 - Informelle Beschreibung: **Pflichtenheft**
 - Konzeptioneller Entwurf: **E/R-Diagramm**
 - Relationaler DB-Entwurf: **Relationenschema**
- In diesem Kapitel:
 - Normalisierungstheorie als formale Grundlage für den relationalen DB-Entwurf**
- Zentrale Fragestellungen:
 - Wie können Objekte und deren Beziehungen ins relationale Modell überführt werden
 - Bewertungsgrundlagen zur Unterscheidung zwischen „guten“ und „schlechten“ relationalen DB-Schemata

Motivation Normalisierung

- **Relation Lieferant:**

<u>LNr</u>	LName	LStadt	LLand	<u>Ware</u>	Preis
103	Huber	Berlin	Deutschland	Schraube	50
103	Huber	Berlin	Deutschland	Draht	20
103	Huber	Berlin	Deutschland	Nagel	40
...
762	Maier	Zürich	Schweiz	Nagel	45
762	Maier	Zürich	Schweiz	Schraube	55

- **Hier gibt es offensichtlich einige Redundanzen:**

- Wenn bei mehreren Tupeln der Attributwert LNr gleich ist, dann müssen auch die Attributwerte von LName, LStadt und LLand gleich sein.
- Wenn (auch bei verschiedenen LNr) LStadt gleich ist, dann muss auch LLand gleich sein.

- **Redundanzen durch funktionale Abhängigkeiten**

- Wir sagen: Die Attribute LName, LStadt und LLand sind funktional abhängig vom Attribut LNr (ebenso LLand von LStadt).

Motivation Normalisierung

• Relation Lieferant:

<u>LNr</u>	LName	LStadt	LLand	<u>Ware</u>	Preis
103	Huber	Berlin	Deutschland	Schraube	50
103	Huber	Berlin	Deutschland	Draht	20
103	Huber	Berlin	Deutschland	Nagel	40
...
762	Maier	Zürich	Schweiz	Nagel	45
762	Maier	Zürich	Schweiz	Schraube	55

- Redundanzen führen zu Speicherplatzverschwendung.
- **Das eigentliche Problem sind aber Anomalien (Inkonsistenzen durch Änderungsoperationen) und, dass das Schema nicht intuitiv ist:**
 - **Update-Anomalie:** Änderung der Adresse (LName, LStadt, LLand) in nur *einem* Tupel statt in *allen* Tupeln zu einer LNr.
 - **Insert-Anomalie:** Einfügen eines Tupels mit inkonsistenter Adresse; Einfügen eines Lieferanten erfordert Ware
 - **Delete-Anomalie:** Löschen der letzten Ware löscht die Adresse.

Verbesserung

- Neues Datenbankschema:

LieferantAdr	(<u>LNr</u> , LName, LStadt)
Stadt	(<u>LStadt</u> , LLand)
Angebot	(<u>LNr</u> , <u>Ware</u> , Preis)

- Vorteile:
 - keine Redundanz
 - keine Anomalien
- Nachteil:
 - Um zu einer Ware die Länder der Lieferanten zu finden, ist ein zweifacher Join nötig (teuer auszuwerten und umständlich zu formulieren)

Ursprüngliche Relation

- Die ursprüngliche Relation Lieferant kann mit Hilfe einer View simuliert werden:

```
create view Lieferant as  
  select  L.LNr, LName, L.LStadt, LLand, Ware, Preis  
  from    LieferantAdr L, Stadt S, Angebot A  
  where   L.LNr = A.LNr and L.LStadt = S.LStadt
```

Schema-Zerlegung

- Anomalien entstehen durch Redundanzen
- Entwurfsziele:
 - Vermeidung von Redundanzen
 - Vermeidung von Anomalien
 - evtl. Einbeziehung von Effizienzüberlegungen
- Vorgehen:
Schrittweises Zerlegen des gegebenen Schemas (Normalisierung) in ein äquivalentes Schema ohne Redundanz und Anomalien
- Formalisierung von Redundanz und Anomalien:
Funktionale Abhängigkeit

Funktionale Abhängigkeit

(engl. Functional Dependency, FD)

- beschreibt Beziehungen zwischen den Attributen einer Relation
- Schränkt das Auftreten gleicher bzw. ungleicher Attributwerte innerhalb einer Relation ein
→ spezielle Integritätsbedingung (nicht in SQL)

Wiederholung Integritätsbedingungen in SQL:

- Primärschlüssel
- Fremdschlüssel (referenzielle Integrität)
- **not null**
- **check**

Wiederholung *Schlüssel*

Definition:

- Eine Teilmenge S der Attribute eines Relationenschemas R heißt ***Schlüssel***, wenn gilt:
 - ***Eindeutigkeit***
Keine Ausprägung von R kann zwei verschiedene Tupel enthalten, die sich in ***allen*** Attributen von S gleichen.
 - ***Minimalität***
Keine echte Teilmenge von S erfüllt bereits die Bedingung der Eindeutigkeit

Konventionen zur Notation

- Ab jetzt gilt die Notation:
 - A, B, C bezeichnen einzelne Attribute
 - X, Y, Z bezeichnen Mengen von Attributen
- Zur Vereinfachung gilt außerdem:
 - $A, B \rightarrow C$ bezeichne $\{A, B\} \rightarrow \{C\}$
 - $X \rightarrow Y, Z$ bezeichne $X \rightarrow Y \cup Z$
 - $t.A$ bezeichne das Attribut A des Tupels t
 - $t.X$ bezeichne die Menge X von Attributen des Tupels t
 - $t.X = r.X$ bezeichne $\forall A \in X : t.A = r.A$

Definition: *funktional abhängig*

- **Gegeben:**

- Ein Relationenschema R
- X, Y : Zwei Mengen von Attributen von R ($X, Y \subseteq R$)

- **Definition:**

Y ist von X funktional abhängig ($X \rightarrow Y$)

gdw. \forall Tupel t und $r : t.X = r.X \Rightarrow t.Y = r.Y$

(für alle möglichen Ausprägungen von R gilt:

Zu jedem Wert in X existiert genau ein Wert von Y .)

- Bsp.: **Lieferant** (LNr, LName, LStadt, LLand, Ware, Preis):

- $LNr \rightarrow LName$
- $LNr \rightarrow LStadt$
- $LStadt \rightarrow LLand$
- $LNr, Ware \rightarrow LName$
- $LNr, Ware \rightarrow LStadt$
- $LNr, Ware \rightarrow Preis$

Vergleich mit *Schlüssel*

- Gemeinsamkeiten zwischen dem *Schlüssel* im relationalen Modell und *Funktionaler Abhängigkeit*:
 - Definitionen ähnlich
 - Für alle Schlüsselkandidaten $S = \{A, B, \dots\}$ gilt:
Alle Attribute der Relation sind von S funktional abhängig:
 $\{A, B, \dots\} \rightarrow R$
- Unterschied:
 - Aber es gibt u.U. weitere funktionale Abhängigkeiten:
Ein Attribut B kann z.B. auch funktional abhängig sein
 - von Nicht-Schlüssel-Attributen
 - von nur einem Teil des Schlüssels (nicht vom gesamten Schlüssel)
- FD ist Verallgemeinerung des Schlüssel-Konzepts

Vergleich mit *Schlüssel*

- Wie der Schlüssel ist auch die funktionale Abhängigkeit eine ***semantische Eigenschaft*** des Schemas:
 - FD nicht aus aktueller DB-Ausprägung entscheidbar
 - sondern muss für alle möglichen Ausprägungen gelten

Prime Attribute

- Definition:
Ein Attribut heißt ***prim***,
wenn es Teil eines Schlüsselkandidaten ist

Partielle und volle FD

- Ist ein Attribut B funktional von A abhängig, dann auch von jeder Obermenge von A.
Man ist interessiert, minimale Mengen zu finden, von denen B abhängt (vgl. Schlüsseldefinition)
- **Definition:**
 - Gegeben: Eine funktionale Abhängigkeit $X \rightarrow Y$
 - Wenn es keine echte Teilmenge $X' \subset X$ gibt, von der Y ebenfalls funktional abhängt,
 - dann heißt $X \rightarrow Y$ eine **volle funktionale Abhängigkeit** ($X \twoheadrightarrow Y$)
 - andernfalls eine **partielle funktionale Abhängigkeit**

Partielle und volle FD

- Beispiele:

- $LNr \rightarrow Lname$ voll funktional abhängig (von einem Element)
- $LNr, Ware \rightarrow LName$ partiell funktional abhängig ($LNr \rightarrow LName$)
- $Ware \text{ ? } Preis$ nicht funktional abhängig (siehe Beispiel)
- $LNr \text{ ? } Preis$ nicht funktional abhängig (siehe Beispiel)
- $LNr, Ware \rightarrow Preis$ voll funktional abhängig (weder von LNr noch von $Ware$ alleine funktional abhängig)

Herleitung funktionaler Abhängigkeit

Armstrong Axiome

- Reflexivität (R): Falls Y eine Teilmenge von X ist ($Y \subseteq X$), dann gilt immer $X \rightarrow Y$.
Inbesondere gilt also immer $X \rightarrow X$.
- Verstärkung (VS): Falls $X \rightarrow Y$ gilt, dann gilt auch $XZ \rightarrow YZ$.
Hierbei steht XZ für $X \cup Z$.
- Transitivität (T): Falls $X \rightarrow Y$ und $Y \rightarrow Z$ gilt, dann gilt auch $X \rightarrow Z$.

Diese Axiome sind **vollständig** und **korrekt** :

Sei F eine Menge von FDs:

- es lassen sich nur FDs von F ableiten, die von jeder Relationenausprägung erfüllt werden, für die auch F erfüllt ist.
- alle FDs ableitbar, die durch F impliziert sind.

Herleitung funktionaler Abhängigkeit

Triviale funktionale Abhängigkeit:

- Ein Attribut ist immer funktional abhängig:
 - von sich selbst
 - und von jeder Obermenge von sich selbst

Solche Abhängigkeiten bezeichnet man als *trivial*

Hülle einer Attributmeng

- Eingabe: eine Menge F von FDs und eine Menge von Attributen X .
- Ausgabe: die vollständige Menge von Attributen X^+ , für die gilt $X \rightarrow X^+$.

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 $X^+ = \text{Erg}$

Hülle einer Attributmeng

- **Beispiel:** AttrHülle($F, \{LNr\}$) mit
 $F = \{LNr \rightarrow LName; LNr \rightarrow LStadt; LStadt \rightarrow LLand; LNr, Ware \rightarrow Preis\}$
- $Erg_i = Erg$ nach i -tem Durchlauf der While-Schleife
- $Erg_0 = \{ LNr \}$
- $Erg_1 = \{ LNr, LName, LStadt \}$
- $Erg_2 = \{ LNr, LName, LStadt, LLand \}$
- $Erg_3 = \{ LNr, LName, LStadt, LLand \} = Erg_2$

Herleitung funktionaler Abhängigkeit

Weitere Axiome

- Vereinfachen Herleitungsprozess, aber nicht notwendig (da Armstrong-Axiome vollständig)
- Vereinigungsregel (VE):
Falls $X \rightarrow Y$ und $X \rightarrow Z$ gilt, dann gilt auch $X \rightarrow YZ$.
- Dekompositionsregel (D):
Falls $X \rightarrow YZ$ gilt, dann gilt auch $X \rightarrow Y$ und $X \rightarrow Z$.
- Pseudotransitivitätsregel (P):
Falls $X \rightarrow Y$ und $ZY \rightarrow V$ gilt, dann gilt auch $XZ \rightarrow V$.

Herleitung funktionaler Abhängigkeit

Beispiel: Zeige, dass (LNr, Ware) Schlüsselkandidat ist.

- Gegeben: $F = \{ \text{LNr} \rightarrow \text{LName}; \text{LNr} \rightarrow \text{LStadt}; \text{LStadt} \rightarrow \text{LLand}; \text{LNr, Ware} \rightarrow \text{Preis} \}$.
- Zu zeigen: (LNr, Ware) eindeutig und minimal.
- Beweis:
 - (I) (LNr, Ware) eindeutig gdw.
 $(\text{LNr, Ware} \rightarrow \text{LNr, LName, LStadt, LLand, Ware, Preis}) \in F^+$,
d.h. ist aus F herleitbar.
 - (a) $(\text{LNr} \rightarrow \text{LName}) \in F \subseteq F^+ \xrightarrow{\text{VS}} (\text{LNr, Ware} \rightarrow \text{LName, Ware}) \in F^+$
 - (b) $(\text{LNr} \rightarrow \text{LStadt}) \in F^+ \xrightarrow{\text{VS}} (\text{LNr, Ware} \rightarrow \text{LStadt, Ware}) \in F^+$
 - (c) $(\text{LNr} \rightarrow \text{LStadt}) \in F^+$ und $(\text{LStadt} \rightarrow \text{LLand}) \in F^+$
 $\xrightarrow{\text{T}} (\text{LNr} \rightarrow \text{LLand}) \in F^+ \xrightarrow{\text{VS}} (\text{LNr, Ware} \rightarrow \text{LLand, Ware}) \in F^+$
 - (d) Wegen R gilt $(\text{LNr} \rightarrow \text{LNr}) \in F^+ \xrightarrow{\text{VS}} (\text{LNr, Ware} \rightarrow \text{LNr, Ware}) \in F^+$
 - (e) aus (a) bis (d) und $(\text{LNr, Ware} \rightarrow \text{Preis})$ folgt (I) nach VE, qed.

Herleitung funktionaler Abhängigkeit

Beispiel: Zeige, dass $(LNr, Ware)$ Schlüsselkandidat ist.

- Gegeben: $F = \{ LNr \rightarrow LName; LNr \rightarrow LStadt; LStadt \rightarrow LLand; LNr, Ware \rightarrow Preis \}$.
- Zu zeigen: $(LNr, Ware)$ eindeutig und minimal.
- Beweis:
 - (II) $(LNr, Ware)$ minimal gdw.
 - $(LNr \rightarrow LNr, LName, LStadt, LLand, Ware, Preis)$ und
 - $(Ware \rightarrow LNr, LName, LStadt, LLand, Ware, Preis)$ gelten nicht.
 - (a) Preis ist nur von LNr und Ware gemeinsam funktional abhängig.
 - (b) Weder LNr kann aus Ware hergeleitet werden, noch Ware von LNr.
 - (c) aus (a) und (b) folgt $(LNr, Ware)$ minimal, qed.

Normalisierung

- In einem Relationenschema sollen also möglichst keine funktionalen Abhängigkeiten bestehen, außer vom gesamten Schlüssel
- Verschiedene Normalformen beseitigen unterschiedliche Arten von funktionalen Abhängigkeiten bzw. Redundanzen/Anomalien
 - 1. Normalform
 - 2. Normalform
 - 3. Normalform
 - Boyce-Codd-Normalform
- Herstellung einer Normalform durch verlustlose Zerlegung des Relationenschemas

1. Normalform

- Keine Einschränkung bezüglich der FDs
- Ein Relationenschema ist in erster Normalform, wenn alle Attributwerte **atomar** sind
- In relationalen Datenbanken sind nicht-atomare Attribute ohnehin nicht möglich
- Nicht-atomare Attribute z.B. durch **group by**

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

← „nested relation“
non first normal form
In SQL nur temporär erlaubt

2. Normalform

- Motivation:
Man möchte verhindern, dass Attribute nicht vom gesamten Schlüssel voll funktional abhängig sind, sondern nur von einem Teil davon.
- Beispiel:

<u>LNr</u>	LName	LStadt	LLand	<u>Ware</u>	Preis
103	Huber	Berlin	Deutschland	Schraube	50
103	Huber	Berlin	Deutschland	Draht	20
103	Huber	Berlin	Deutschland	Nagel	40
...
762	Maier	Zürich	Schweiz	Nagel	45
762	Maier	Zürich	Schweiz	Schraube	55

Anomalien?

Konsequenz: In den abhängigen Attributen muss dieselbe Information immer wiederholt werden


2. Normalform

- Dies fordert man vorerst nur für Nicht-Schlüssel-Attribute (für die anderen z.T. schwieriger)
- Definition
Ein Schema ist in zweiter Normalform, wenn jedes Attribut
 - voll funktional abhängig von **allen** Schlüsselkandidaten
 - oder
 - prim ist
- Beobachtung:
Zweite Normalform kann nur verletzt sein, wenn...
...ein Schlüssel(kandidat) zusammengesetzt ist

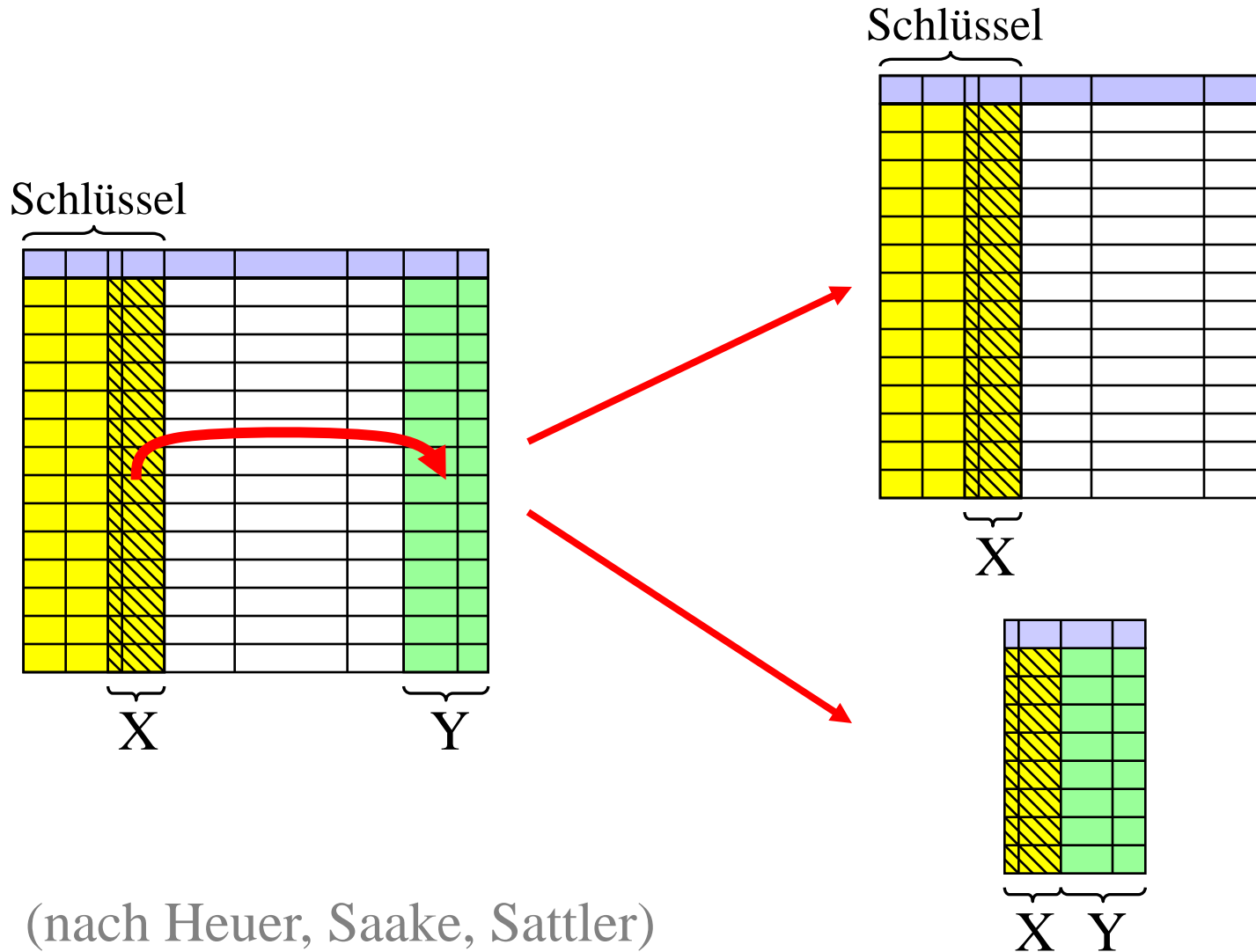
2. Normalform

- Zur Transformation in 2. Normalform spaltet man das Relationenschema auf:
 - Attribute, die voll funktional abhängig vom Schlüssel sind, bleiben in der Ursprungsrelation R
 - Für alle Abhängigkeiten $X_i \rightarrow Y_i$ von einem Teil eines Schlüssels ($X_i \subsetneq S$) geht man folgendermaßen vor:
 - Lösche die Attribute Y_i aus R
 - Gruppier die Abhängigkeiten nach gleichen linken Seiten X_i
 - Für jede Gruppe führe eine neue Relation ein mit allen enthaltenen Attributen aus X_i und Y_i
 - X_i wird Schlüssel in der neuen Relation

2. Normalform

- Beispiel:  partielle Abhängigkeiten
Lieferant (LNr, LName, LStadt, LLand, Ware, Preis)
- Vorgehen:
 - LName, LStadt und LLand werden aus Lieferant gelöscht
 - Gruppierung:
 - es könnten Attribute von Ware abhängen (2. Gruppe)
 - Erzeugen einer Relation mit LNr, LName, LStadt und LLand
- Ergebnis: **Lieferung** (LNr, Ware, Preis)
LieferAdr (LNr, LName, LStadt, LLand)

Grafische Darstellung




3. Normalform

- Motivation:
Man möchte zusätzlich verhindern, dass Attribute von nicht-primen Attributen funktional abhängen.

- Beispiel:

LieferAdr (LNr, LName, LStadt, LLand)



001	Huber	München	Deutschland
002	Meier	Berlin	Deutschland
003	Müller	Köln	Deutschland
004	Hinz	Wien	Österreich
005	Kunz	Salzburg	Österreich

- Redundanz: Land mehrfach gespeichert
- Anomalien?

3. Normalform

- Abhängigkeit von Nicht-Schlüssel-Attribut bezeichnet man häufig auch als **transitive** (d.h. durch Armstrong-Transitivitätsaxiom hergeleitete) **Abhängigkeit** vom Primärschlüssel
 - weil Abhängigkeit *über* ein drittes Attribut besteht:

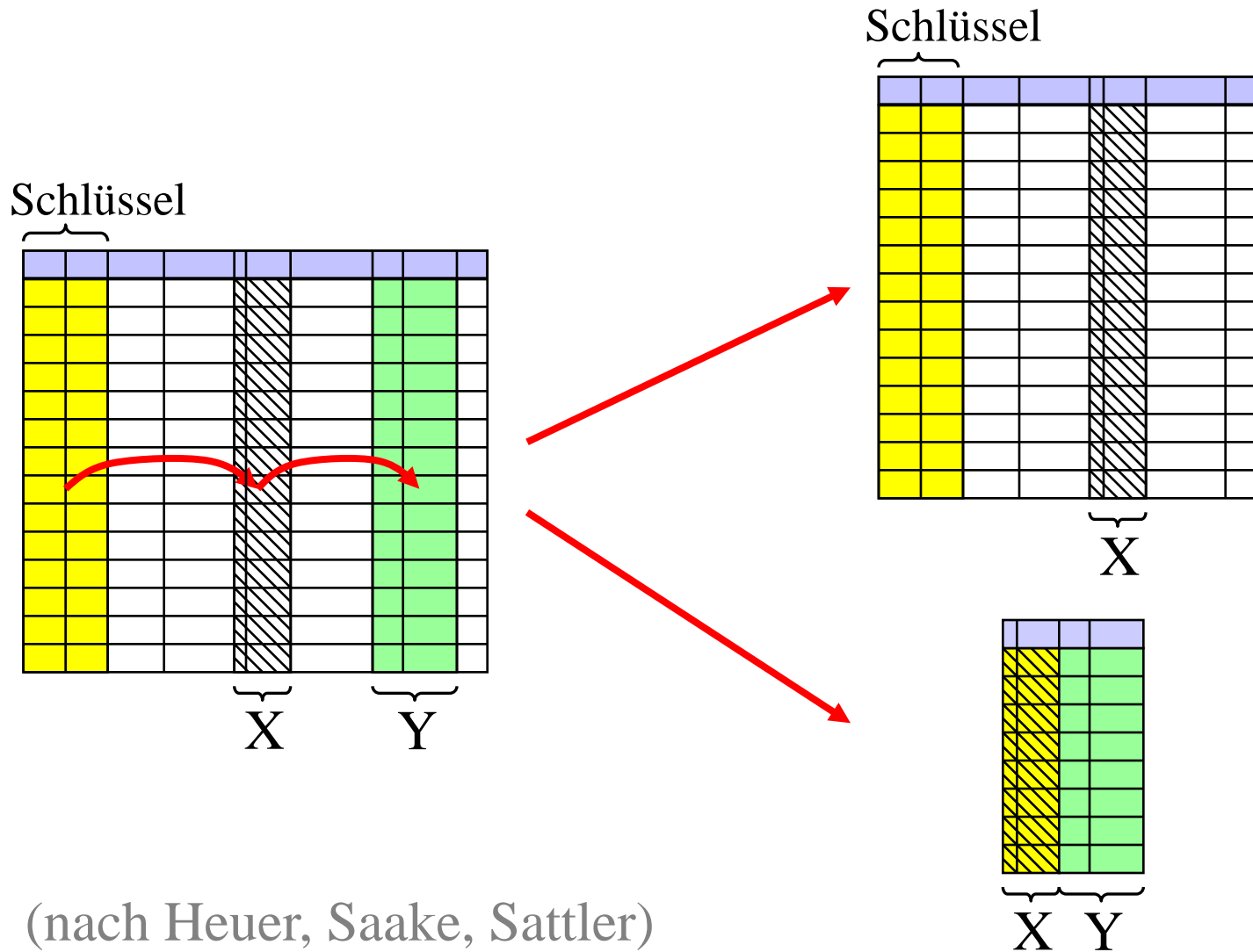


- Definition:
Ein Relationenschema ist in 3. Normalform, wenn für jede nichttriviale Abhängigkeit $X \rightarrow Y$ gilt:
 - X enthält einen Schlüsselkandidaten
 - oder Y ist prim
- Beobachtung: 2. Normalform ist mit impliziert

3. Normalform

- Transformation in 3. Normalform wie vorher
 - Attribute, die voll funktional abhängig vom Schlüssel sind, und nicht abhängig von Nicht-Schlüssel-Attributen sind, bleiben in der Ursprungsrelation R
 - Für alle Abhängigkeiten $X_i \rightarrow Y_i$ von einem Teil eines Schlüssels ($X_i \subset S$) oder von Nicht-Schlüssel-Attribut:
 - Lösche die Attribute Y_i aus R
 - Gruppier die Abhängigkeiten nach gleichen linken Seiten X_i
 - Für jede Gruppe führe eine neue Relation ein mit allen enthaltenen Attributen aus X_i und Y_i
 - X_i wird Schlüssel in der neuen Relation

Grafische Darstellung



Schlussbemerkungen

- Ein gut durchdachtes E/R-Diagramm liefert bereits weitgehend normalisierte Tabellen
- Normalisierung ist in gewisser Weise eine Alternative zum E/R-Diagramm
- Extrem-Ansatz: *Universal Relation Assumption*
 - Modelliere alles zunächst in einer Tabelle
 - Ermittle die funktionalen Abhängigkeiten
 - Zerlege das Relationenschema entsprechend (der letzte Schritt kann auch automatisiert werden: Synthesealgorithmus für die 3. Normalform)

Schlussbemerkungen

- Normalisierung kann schädlich für die Performanz sein, weil Joins sehr teuer auszuwerten sind
- Nicht *jede* FD berücksichtigen:
 - Abhängigkeiten zw. Wohnort, Vorwahl, Postleitzahl
 - Man kann SQL-Integritätsbedingungen formulieren, um Anomalien zu vermeiden (sog. Trigger)
- Aber es gibt auch Konzepte, Relationen so abzuspeichern, dass Join auf bestimmten Attributen unterstützt wird
 - ORACLE-Cluster

Zusammenfassung

Implikation

- 1. Normalform:
Alle Attribute atomar
- 2. Normalform:
Keine funktionale Abhängigkeit eines Nicht-Schlüssel-Attributs von Teil eines Schlüssels
- 3. Normalform:
Zusätzlich keine nichttriviale funktionale Abhängigkeit eines Nicht-Schlüssel-Attributs von Nicht-Schlüssel-Attributen
- Boyce-Codd-Normalform:
Zusätzlich keine nichttriviale funktionale Abhängigkeit unter den Schlüssel-Attributen