

UNIT 0x09

NORMALFORMEN III

SYNTHESEALGORITHMUS

DEKOMPOSITIONSALGORITHMUS

Motivation/Erinnerung

Aufgabe

- Entwicklung einer 'guten' Datenbank [...] u.a. minimal, redundanzfrei [...]

Vorgehen

- Konzeptueller Entwurf [...] ER-Modells ✓
- Überführung in eine 'gute' Datenbank [...] dazu benötigen wir:
 - Implementationsentwurf [...] → Relationales Modell ✓
 - Mathematisches Modell [...] → Relationale Algebra ✓
 - Gütemaß → Normalformen (Anomalien, Verlustlosigkeit, Abhängigkeitserhaltung)
 - ♦ Kanonische Überdeckung ✓, funktionale Abhängigkeiten ✓, Attributhülle ✓,
 - ♦ Synthesealgorithmus (✓),
 - ♦ Dekompositionsalgorithmus

Anomalien

CRUD

- Unter CRUD versteht man die fundamentalen Datenbankoperationen:
 - **Create:** Datensatz anlegen,
 - **Read/Retrieve:** Datensatz lesen,
 - **Update:** Datensatz aktualisieren, und
 - **Delete/Destroy:** Datensatz löschen.
- Da der Lesezugriff (Read) die Daten nicht ändert, betrachtet man die folgenden Anomalien primär im Zusammenhang mit den anderen, die Daten verändernden, Operationen Create (Einfügung), Update und Delete (Lösung).
- Anmerkung: Die CRUD Operationen sind auch die, die in Persistenz-Frameworks häufig die konkreten Operationen auf dem DBMS abstrahieren.

Anomalien

Definition

- Es gibt:
 - **Update-Anomalien**
 - **Einfüge-Anomalien**
 - **Löschen-Anomalien.**
- Ausgangspunkt der Beispiele ist immer eine Relation, die diese Anomalien auch zuläßt und in diesem Sinne keinen "guten Datenbankentwurf" darstellt. Ein Datenbankentwurf, der bestimmte Normalformen erfüllt, vermeidet diese.

Anomalien

Update-Anomalie

- Angenommen, Vorlesungen und Professoren sind in einer Tabelle realisiert.
- Durch das Update einer falschen Raumnr. werden die Daten inkonsistent, da Sokrates kein eindeutiger Raum mehr zugeordnet ist.
- Generell tritt diese Anomalie auf, wenn redundante Daten in einem Tupel nur teilweise (falsch) aktualisiert werden.

ProfVorl						
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
...
2132	Popper	C3	52	5259	Der Wiener Kreis	2
2137	Kant	C4	7	4630	Die 3 Kritiken	4

- **Update-Anomalie**

```
update ProfVorl
  set Raum=111
where Titel="Ethik"
```

Unterlagen Prof. Striegnitz

Anomalien

Einfüge-Anomalie

- Angenommen, Vorlesungen und Professoren sind in einer Tabelle realisiert *und* VorlNr darf nicht NULL sein!
- Wenn Fr. Curie noch keine Vorlesung hält, ist Einfügen nicht möglich.
- Generell sind bei dieser Anomalie Daten so verbunden, dass sie nicht ohne andere (not NULL) eingegeben werden können.

ProfVorl						
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
...
2132	Popper	C3	52	5259	Der Wiener Kreis	2
2137	Kant	C4	7	4630	Die 3 Kritiken	4

- **Einfüge-Anomalie**

```
insert into ProfVorl
values (2239,Curie,C4,112)
```

Unterlagen Prof. Striegnitz

Anomalien

Löschen-Anomalie

- Angenommen, Vorlesungen und Professoren sind in einer Tabelle realisiert.
- Beim Löschen der Vorlesung "Der Wiener Kreis" wird auch Hr. Popper gelöscht, wenn das seine einzige Vorlesung ist.
- Generell müssen bei dieser Anomalie alle Daten eines Tupels gelöscht werden, obwohl eine Teilmenge gereicht hätte.

ProfVorl						
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
...
2132	Popper	C3	52	5259	Der Wiener Kreis	2
2137	Kant	C4	7	4630	Die 3 Kritiken	4

- **Löschen-Anomalie**

```
delete ProfVorl
where Titel="Der Wiener Kreis"
```

Unterlagen Prof. Striegnitz

Verlustlosigkeit und Abhängigkeitserhaltung

Lösungsansatz

- Offensichtlich wurden in der Relation zwei Sachverhalte modelliert.
- Trennt man die Relation auf (Dekomposition), so muss natürlich später die Zusammenführung möglich sein (Rekomposition). Dazu benötigt man einen gemeinsamen, ggf. neuen, Bereich, der in beiden Relationen vorkommt.

ProfVorl						
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
...
2132	Popper	C3	52	5259	Der Wiener Kreis	2
2137	Kant	C4	7	4630	Die 3 Kritiken	4

Professoren

PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

VorlNr	Titel	SWS	gelesenVon
4052	Logik	4	2125
4630	Die 3 Kritiken	4	2137
5001	Grundzuege	4	2125
5022	Glaube und Wissen	2	2134
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Maeeutik	2	2125

Vorlesungen

Verlustlosigkeit und Abhängigkeitserhaltung

Definition

Gegeben sei eine Relation R mit einer Menge funktionaler Abhängigkeiten. Diese Relation soll in neue Relationen R_i aufgeteilt werden, z.B. um Anomalien zu vermeiden.

- **Verlustlosigkeit/Verbundtreue:** Die in der ursprünglichen Relation R enthaltenen Informationen müssen aus den neuen Relationen R_1, \dots, R_n mittels natürlichen Verbunds (Natural Join) rekonstruierbar sein.
- **Abhängigkeitserhaltung:** Die ursprünglich geltenden funktionalen Abhängigkeiten müssen auch auf der Zerlegung, also den neuen Relationen, gelten.

Q&A

- Fallen Ihnen Aufteilungen ein, wo man Informationen oder Abhängigkeiten verliert?

Relation ist hier im Sinne eines *Schemas* gemeint, also nicht auf eine konkrete Ausprägung bezogen.



Verlustlosigkeit und Abhängigkeitserhaltung

Hinreichendes Kriterium für Verlustlosigkeit

- Eine Zerlegung einer Relation R in zwei Relationen R_1 und R_2 ist *verlustlos*, wenn man mind. eine der beiden aus dem gemeinsamen Bereich (Überlappung der Attribute) wieder ableiten kann. D.h. es gilt:
 - ♦ $R_1 \cap R_2 \rightarrow R_1$, oder
 - ♦ $R_1 \cap R_2 \rightarrow R_2$.

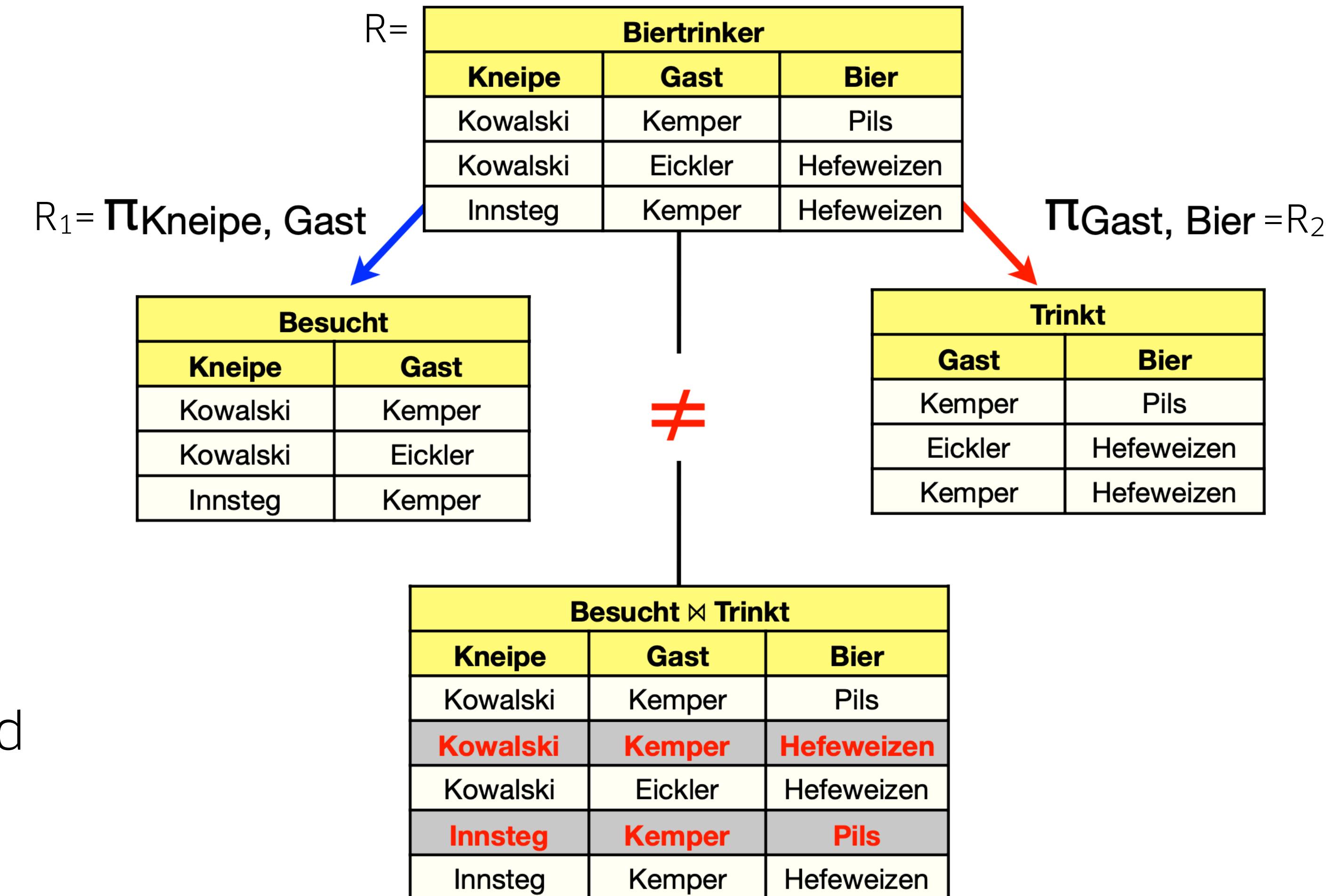


So findet man es gerne in der Literatur. Gemeint sind hier wieder die Schemata und nicht eine konkrete Ausprägung der Daten, denn die funktionalen Abhängigkeiten gelten für alle Ausprägungen. Vgl. Diskussion um funktionale Abhängigkeiten.

Verlustlosigkeit und Abhängigkeitserhaltung

Beispiel Verletzung der Verlustlosigkeit

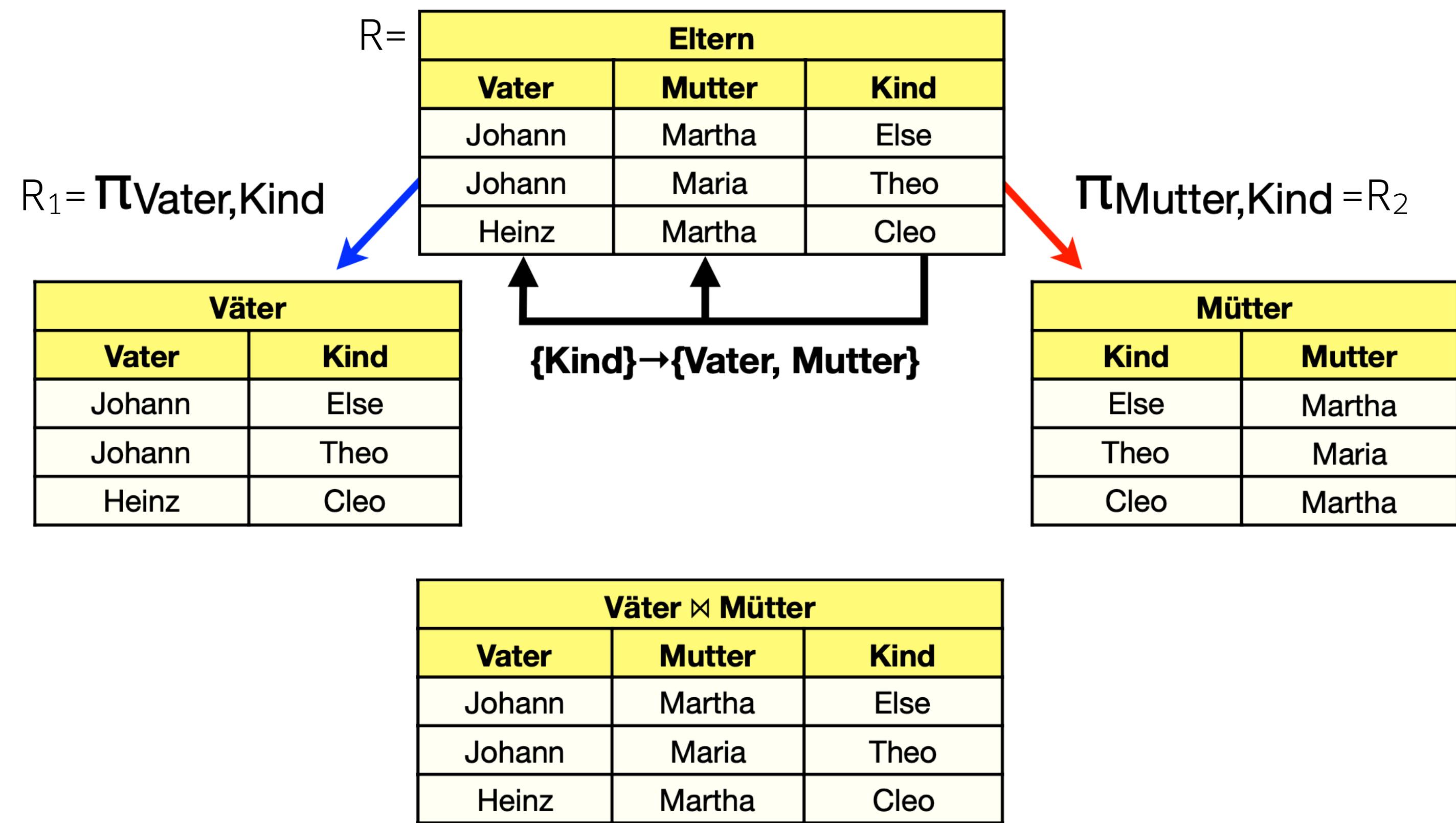
- Die Aufteilung in R_1 und R_2 ist *nicht verlustlos*, denn die Rekomposition durch den natürlichen Verbund enthält hier *mehr* Daten als in der ursprünglichen Relation!
- Insbesondere gilt:
 - $R_1 \cap R_2 = \text{Gast} \not\rightarrow R_1$, und
 - $R_1 \cap R_2 = \text{Gast} \not\rightarrow R_2$.



Verlustlosigkeit und Abhängigkeitserhaltung

Beispiel Verlustlosigkeit

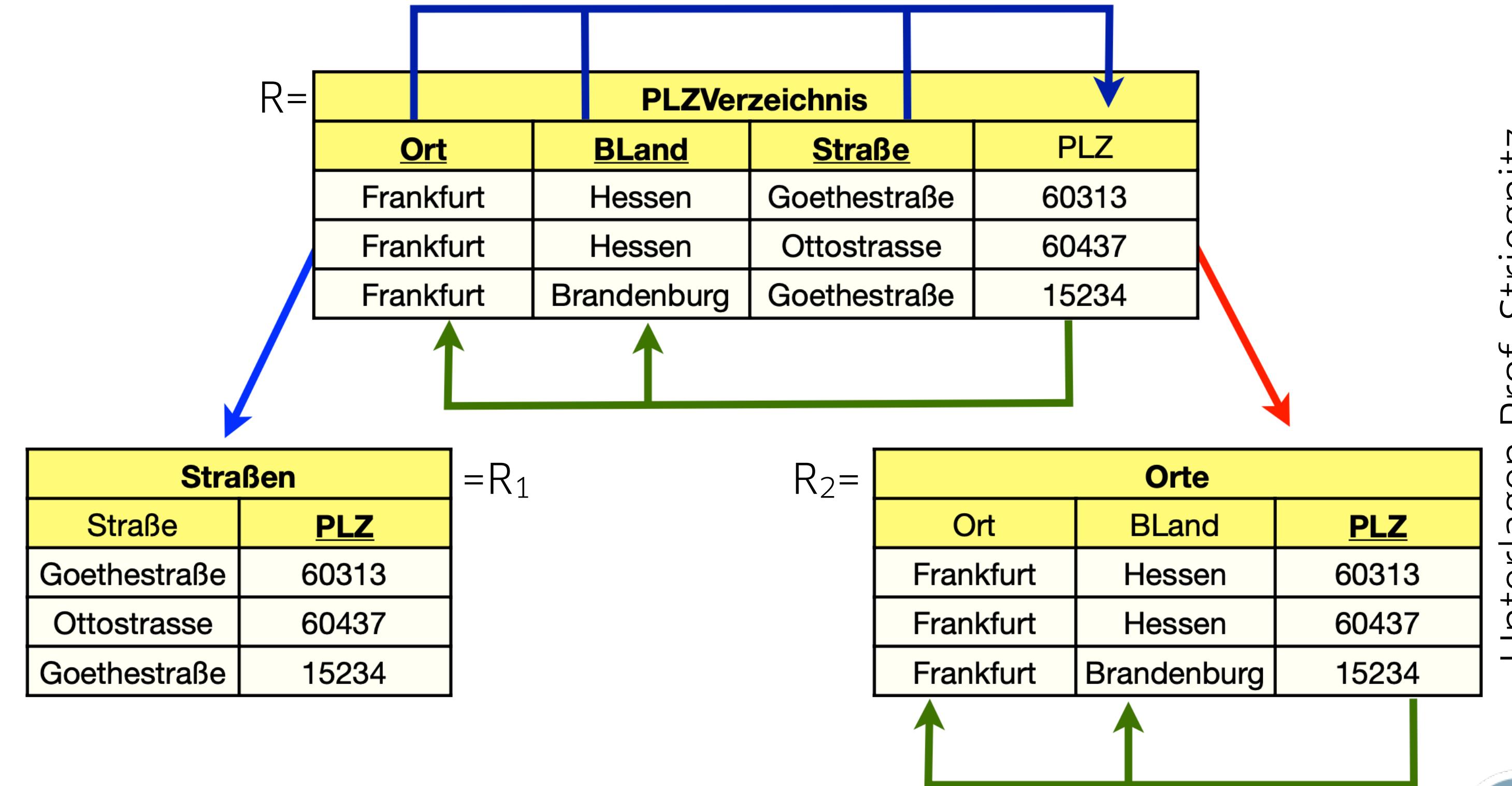
- Die Aufteilung in R_1 und R_2 ist *verlustlos*, denn die Rekomposition durch den natürlichen Verbund ergibt hier wieder die ursprünglichen Relation!
- Insbesondere gilt:
 - ♦ $R_1 \cap R_2 = \text{Kind} \rightarrow R_1$, und
 - ♦ $R_1 \cap R_2 = \text{Kind} \rightarrow R_2$.



Verlustlosigkeit und Abhängigkeitserhaltung

Beispiel Verlust der Abhängigkeitserhaltung

- Orte sind durch Namen und Bundesland eindeutig identif.
- Innerhalb einer Straße ändert sich die Postleitzahl nicht.
- Postleitzahlengebiete gehen nicht über Ortsgrenzen.
- Funktionale Abhängigkeiten:
 - PLZ → Ort,BLand
 - Straße, Ort, BLand → PLZ



Die Zerlegung ist verlustfrei, aber die funktionale Abhängigkeit $\text{Straße}, \text{Ort}, \text{BLand} \rightarrow \text{PLZ}$ geht verloren.



Verlustlosigkeit und Abhängigkeitserhaltung

Beispiel Verlust der Abhängigkeitserhaltung

- Durch den Verlust der funktionalen Abhängigkeit sind diese neuen Einfügungen in R_1 und R_2 möglich, führen aber zur Verletzung bei der Rekombination.

$R =$

PLZVerzeichnis			
Ort	BLand	Straße	PLZ
Frankfurt	Hessen	Goethestraße	60313
Frankfurt	Hessen	Ottostrasse	60437
Frankfurt	Brandenburg	Goethestraße	15234

$R_1 =$

Straßen	
Straße	PLZ
Goethestraße	60313
Ottostrasse	60437
Goethestraße	15234
Goethestraße	15235

$R_2 =$

Orte		
Ort	BLand	PLZ
Frankfurt	Hessen	60313
Frankfurt	Hessen	60437
Frankfurt	Brandenburg	15234
Frankfurt	Brandenburg	15235

$R_1 \bowtie R_2 =$

Straßen \bowtie Orte			
Ort	BLand	Straße	PLZ
Frankfurt	Hessen	Goethestraße	60313
Frankfurt	Hessen	Galgenstraße	60437
Frankfurt	Brandenburg	Goethestraße	15234
Frankfurt	Brandenburg	Goethestraße	15235

Offenbar kann man eine Relation nicht beliebig aufspalten... !



Funktionale Abhängigkeit verletzt.



Normalformen

Roter Faden

- Zur Motivation der Normalformen betrachtet man *Anomalien*, die durch die jeweilige Normalformen verhindert werden.
- Verändert man ein Schema, um einer Normalform zu genügen, muss man auf *Verlustlosigkeit und Abhängigkeitserhaltung* achten. Synthese- und Dekompositions-algorihmus überführen jeweils eine Schema in eine bestimmte Normalform.

Normalformen

- Erste Normalform (1NF, manchmal auch NF1),
- Zweite Normalform (2NF/2NF),
- Dritte Normalform (3NF/NF3),
- Boyce-Codd-Normalform (BCNF),
- Vierte und Fünfte Normalform (hier nicht relevant).



Erste Normalform

Definition

- Ein Schema in erster Normalform (1NF, auch NF1) besitzt nur atomare/elementare Attribute, d.h. kein Attribut ist zusammengesetzt oder mehrwertig.
- Funktionale Abhängigkeiten spielen keine Rolle.

Anmerkungen

- Die Atomarität hatten wir schon beim Übergang vom ER-Diagramm in das Implementationsmodell gefordert.
- Man findet auch die Bezeichnung 'Nullte Normalform' für ein Schema nicht in 1NF.
- Die erste Normalform *ist immer gegeben*, wenn das Schema in der Form anonymer Attribute (A,B,C,...) vorliegt. Nur bei Attributen mit Bedeutung können diese zusammengesetzt oder mehrwertig sein.

Erste Normalform

Beispiel zu Erster Normalform

Nr (EAN,ISBN)	Zeitschrift	Verlags- gründung	Themen
...aa11	Heise - c't - 11/18	1949	S.15 C++, S.24 C#
...bb22	Heise - c't - 12/18	1949	S.12 Python, S.29 C#
...cc33	Heise - ix - 08/18	1949	S.9 RAID, S.23 gpio
...dd44	Computec - buffered - 08/18	1989	S.11 WoW
...ee55	Webedia - GameStar - 08/18	2007	S.13 LoL, S.20 WoW

Nicht in erster Normalform

Nr (N)	Verlag (V)	Magazin (M)	Ausgabe (A)	Gründung (G)	Seite (S)	Thema (T)
aa11	Heise	c't	11/18	1949	15	C++
aa11	Heise	c't	11/18	1949	24	C#
bb22	Heise	c't	12/18	1949	12	Python
bb22	Heise	c't	12/18	1949	29	C#
cc33	Heise	iX	08/18	1949	9	RAID
cc33	Heise	iX	08/18	1949	23	gpio
dd44	Computec	buffered	08/18	1989	11	WoW

Erfüllt erste Normalform

Überführung in erste Normalform

- Um 1NF zu erreichen, müssen Sachverhalte in mehrere Attribute getrennt und Mehrwertigkeiten, typischerweise in eine 1:N-Relation, aufgespalten werden.

Zweite Normalform

Motivation

- Das nebenstehende Schema besitzt den Schlüssel
 - MatrNr,VorlNr. → ident aber es gilt
 - MatrNr → Name, Semester d.h. Attribute sind abhängig von *einem Teil* des Schlüssels.
- Hier sind Redundanzen sichtbar und Update-Anomalien möglich, etwa die Änderung eines Names in nur einem Tupel.

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

Zweite Normalform

Definition

- Ein Schema R ist in zweiter Normalform (2NF, auch NF2), wenn es in 1NF vorliegt und jedes Attribut entweder
 - prim ist, oder
 - voll funktional abhängig von jedem Schlüsselkandidaten ist.

Anmerkungen

- Für den Test, ob 2NF vorliegt, benötigt man *alle* Schlüsselkandidaten.
- Wenn jedes Attribut in irgendeinem Schlüsselkandidaten vorkommt, ist 2NF erfüllt, da es nur prim Attribute gibt.
- 2NF kann nur verletzt sein, wenn ein Schlüsselkandidat zusammengesetzt ist.

Zweite Normalform

Überführung in zweite Normalform

- Die Relation kann verlustfrei und abhängigkeitsbewahrend in zwei Relationen aufgeteilt werden.

Man sieht, dass eine Relation hören sowohl aus einer Überführung in 2NF als auch aus einer guten Modellierung folgt.

StudentenBelegung			
<u>MatrNr</u>	<u>VorlNr</u>	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
...

Nicht in zweiter Normalform

hören	
<u>MatrNr</u>	<u>VorlNr</u>
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
...	...

Studenten		
<u>MatrNr</u>	Name	Semester
26120	Fichte	10
27550	Schopenhauer	6
28106	Carnap	3
...

Erfüllt zweite Normalform



Wieviele Attribute sind es?

Zweite Normalform

Beispiele

- Welche Normalform liegt vor?
 - Schlüsselkandidaten
 - Bedingungen NF
- Normalisieren... (später)

Bsp.1

$$\mathcal{U} = A B C D$$

$$\mathcal{F}D = \{ A \rightarrow BC, B \rightarrow A \}$$

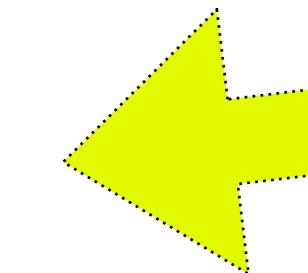
Von welcher NF?

\Rightarrow Schlüsselkand.: D fällt auf rechter Seite (nur in Schl.)

$$\Rightarrow AD, BD$$

\Rightarrow Nicht-Prin: C

\Rightarrow vgl. $A \rightarrow C$ und A Teil einer Schl. kand. Nicht NF2 oder ?NF



Nur diese für 2NF...

Bsp.2

$$\mathcal{U} = A B C D, \mathcal{F}D = \{ A \rightarrow B, B \rightarrow A, AD \rightarrow C \}$$

\Rightarrow Schl. kand.: AD, BD

\Rightarrow Nicht-Prin: C aber vgl. $AD \rightarrow C$ liegt 2NF vor



Dritte Normalform

Motivation

- Das nebenstehende Schema besitzt die funktionalen Abhängigkeiten
 - $\text{Name} \rightarrow \text{Plz}, \text{Ort}, \text{Strasse}$
 - $\text{Plz} \rightarrow \text{Ort}$
 mit Name als Schlüssel.
- Das Schema ist in 2NF, da der Schlüssel nur aus einem Attribut besteht.
- Hier sind ebenfalls Redundanzen sichtbar und Update-Anomalien möglich, etwa die Änderung eines Ortes in nur einem Tupel (vgl. Motivation Zweite Normalform). Das Problem sind funktionale Abhängigkeiten von Attributen, die nicht prim sind (Ort) und nur transitiv vom Schlüssel abhängen (über Plz).

\mathcal{R}			
Name	Plz	Ort	Strasse
Kurt Hanf	52080	Aachen	Von-Coels-Str.
Heidi Panne	52080	Aachen	Karlsstr.

Die 2NF verhindert offenbar nicht alle Redundanzen.



Dritte Normalform

Definition

- Ein Schema R ist in dritter Normalform (3NF, auch NF3), wenn es in 2NF vorliegt und jedes nicht-prim Attribut direkt, also nicht transitiv, von einem Schlüsselkandidaten abhängt.

Anmerkungen

- Gemeint ist, dass bei einer vorliegenden transitiven Abhängigkeit b von β , also $\beta \rightarrow a \rightarrow b$, wobei β ein Schlüsselkandidat ist, hier a kein Schlüsselkandidat ist.

Überführung in dritte Normalform

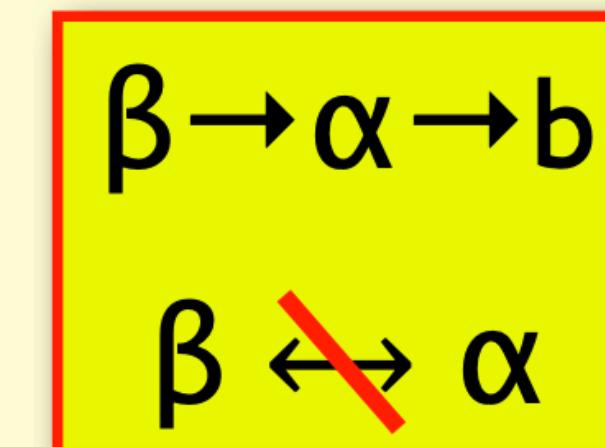
- Der Synthesealgorithmus überführt R verlustlos und abhängigkeits-erhaltend in 3NF.

Dritte Normalform

Definition Transitive / direkte Abhangigkeit

Seien $\mathcal{R} = \{b_1, \dots, b_n\}$ ein vereinfachtes Schema, F eine Menge von funktionalen Abhangigkeiten zu \mathcal{R} , $\beta \subseteq \mathcal{R}$ und $b \in \mathcal{R}$. b ist **transitiv abhangig** von β , falls gilt

- $\beta \rightarrow b \in F^+ \wedge b \notin \beta$
- $\exists \alpha \subseteq \mathcal{R} :$
 - $\alpha \rightarrow b \in F^+ \wedge b \notin \alpha \wedge$
 - $\beta \rightarrow \alpha \in F^+ \wedge$
 - $\alpha \rightarrow \beta \notin F^+$



Gilt $\beta \rightarrow b \in F^+$ und b ist nicht transitiv von β abhangig, so heit b **direkt abhangig** von β .

Dritte Normalform

Alternative Definition

- Ein Schema R mit einer Menge funktionaler Abhängigkeiten FD ist in dritter Normalform (3NF, auch NF3), wenn für jede funktionale Abhängigkeit $\alpha \rightarrow \beta \in FD^+$ mindestens eine der folgenden Bedingungen gilt:
 - $\alpha \rightarrow \beta$ ist trivial, also $\beta \subseteq \alpha$,
 - β enthält nur prim Attribute,
 - α ist Superschlüssel.

Auch wenn es technischer aussieht,
beim Test auf 3NF testet man einfach
alle Abhängigkeiten, ob eine der drei
Bedingungen zutrifft.

Anmerkungen

- Für den Test, ob 3NF vorliegt, benötigt man *alle* Schlüsselkandidaten.
- Die zweite Normalform ist eingeschlossen.
- Diese Definition ist auch für die BCNF-Normalform wichtig.



Dritte Normalform

Beispiele

- Welche Normalform liegt vor?
 - Schlüsselkandidaten
 - Bedingungen NF
- Normalisieren...

(später)

$\mathcal{R} = ABCD \quad FD = \{ A \rightarrow BC, C \rightarrow D \}$, welche NF?

\Rightarrow Schlüsselkand. A \checkmark Nicht-Prim B,C,D

\Rightarrow Trans. Abh. ? ja, sieht man: $A \rightarrow C, C \rightarrow D$
Weg 1 also Nicht 3NF

--- alternativ / immer ---

$A \rightarrow BC$:

Weg 2

Bed. 1: $BC \notin A \quad \text{X}$
Bed. 2: B,C nur Prim? X
Bed. 3: A Superfl. \checkmark

(d.h. $A \rightarrow BC$ reicht Bed. nicht)

$C \rightarrow D$:

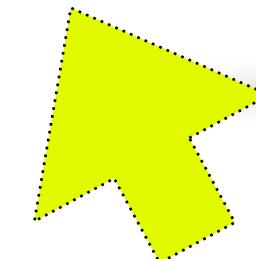


Weg 2 geht immer!

Bed. 1: X
Bed. 2: D Prim? X
Bed. 3: C Superfl.? X

also keine Bed. rüft m \Rightarrow Nicht 3NF

Weg 1: nur nicht-prim



Synthesealgorithmus

Definition

- 1. Bestimme die kanonische Überdeckung F^C zu F . Wiederholung:**
 - Linksreduktion
 - Rechtsreduktion
 - Entfernung von FDs der Form α
 - Zusammenfassung gleicher linker Seiten
- 2. Für jede funktionale Abhängigkeit $\alpha \rightarrow \beta \in F^C$:**
 - Kreiere ein Relationenschema $\mathcal{R}_\alpha := \alpha \cup \beta$
 - Ordne \mathcal{R}_α die FDs $F_\alpha := \{\alpha' \rightarrow \beta' \in F^C \mid \alpha' \cup \beta' \in \mathcal{R}_\alpha\}$ zu.
- 3. Falls eines der in Schritt 2. erzeugten Schemata einen Schlüsselkandidaten von \mathcal{R} bzgl. F^C enthält, sind wir fertig. Sonst wähle einen Schlüsselkandidaten $\kappa \in \mathcal{R}$ aus und definiere folgendes Schema:**
 - $\mathcal{R}_\kappa = \kappa$
 - $F_\kappa = \emptyset$
- 4. Eliminiere diejenigen Schemata \mathcal{R}_α , die in einem anderen Relationenschema \mathcal{R}_α' enthalten sind, d.h.:**
 - $\mathcal{R}_\alpha \subseteq \mathcal{R}_\alpha'$

Synthesealgorithmus

Anmerkungen

- Grundlage ist die kanonische Überdeckung FD^C (1), sie ergibt die Relationen (2).
- Ein Schlüssel muss enthalten sein (3), vgl. dazu auch das kommende Beispiel.
- Relationen, deren Attribute komplett in anderen Relationen enthalten sind, sind überflüssig (4).



**Der Synthesealgorithmus überführt
das Schema in die dritte Normalform!**

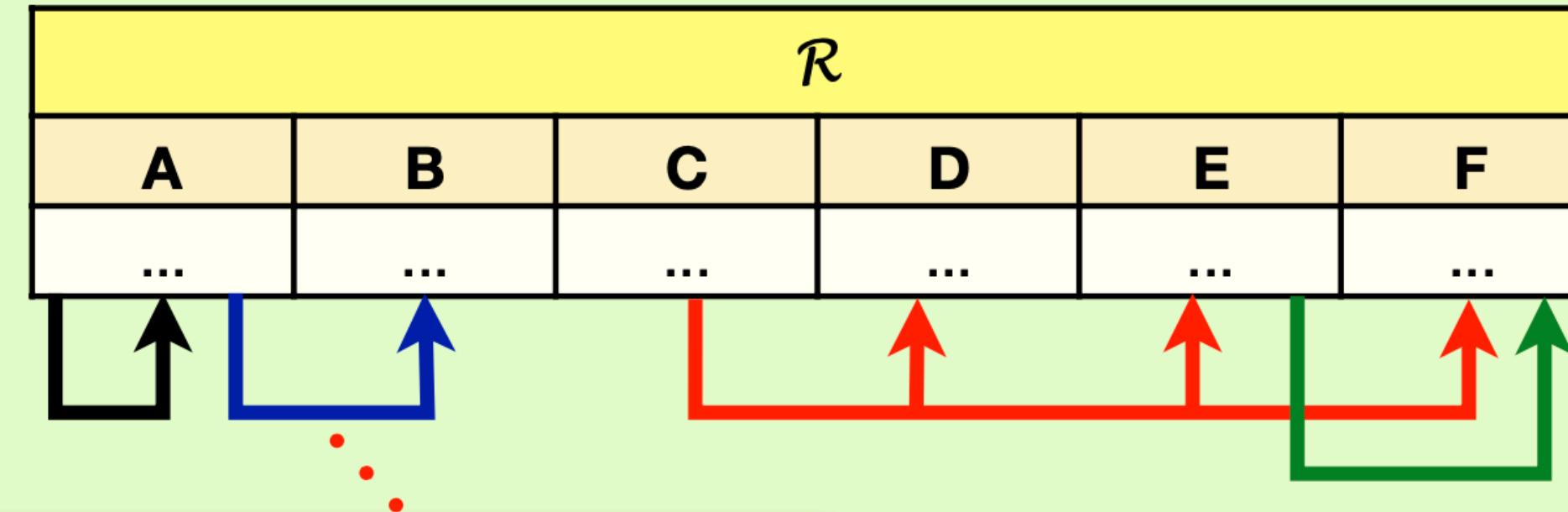
Synthesealgorithmus

Beispiel 1

Wir betrachten ein Beispiel zur Veranschaulichung der Notwendigkeit des 3. Schrittes im Synthesealgorithmus

Seien $\mathcal{R} = \{A, B, C, D, E, F\}$ und folgende FDs gegeben:

- ▶ 1: $\{A\} \rightarrow \{A\}$
- ▶ 2: $\{A\} \rightarrow \{B\}$
- ▶ 3: $\{C\} \rightarrow \{D, E, F\}$
- ▶ 4: $\{E\} \rightarrow \{F\}$



R nicht in 2. NF!

Schlüsselkandidaten:

$L: \beta \subseteq \text{AttrHülle}(F, \alpha - \{a\})$

Linke Seiten der FDs betrachten: $\{A, C, E\}$ ist offenbar Superschlüssel

Linksreduktion:

- Entfernen von A: $\{C, E\} \xrightarrow{3} \{C, D, E, F\}$ - nicht möglich
- Entfernen von C: $\{A, E\} \xrightarrow{2} \{A, B, E\} \xrightarrow{4} \{A, B, E, F\}$ - nicht möglich
- Entfernen von E: $\{A, C\} \xrightarrow{2} \{A, B, C\} \xrightarrow{3} \{A, B, C, D, E, F\}$ - möglich!

{A,C} ist einziger Schlüsselkandidat

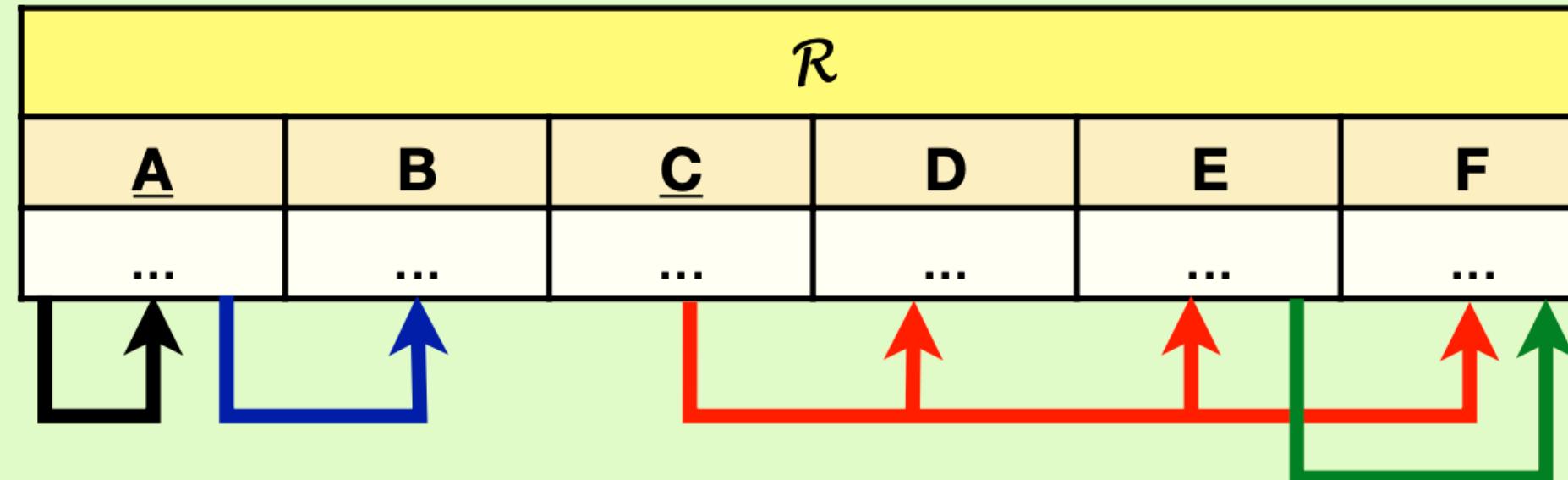
Synthesealgorithmus

Beispiel 1

Wir betrachten ein Beispiel zur Veranschaulichung der Notwendigkeit des 3. Schrittes im Synthesealgorithmus

Seien $\mathcal{R}=\{A, B, C, D, E, F\}$ und folgende FDs gegeben:

- ▶ 1: $\{A\} \rightarrow \{A\}$
- ▶ 2: $\{A\} \rightarrow \{B\}$
- ▶ 3: $\{C\} \rightarrow \{D, E, F\}$
- ▶ 4: $\{E\} \rightarrow \{F\}$



Kanonische Überdeckung:

Linksreduktion:

- nichts zu tun

Rechtsreduktion

- $\{A\} \rightarrow \emptyset$ (trivial)
- $\{A\} \rightarrow \{B\}$ - nichts zu tun
- $\{C\} \rightarrow \{D, E, F\}$ entferne F, denn $\{C\} \rightarrow_3 \{D, E\} \rightarrow_4 \{D, E, F\}$
- $\{E\} \rightarrow \{F\}$ - nichts zu tun

$$F^C = \{A \rightarrow B, C \rightarrow DE, E \rightarrow F\}$$

Synthesealgorithmus

Beispiel 1

Synthesealgorithmus:

$$F^C = \{A \rightarrow B, C \rightarrow DE, E \rightarrow F\}$$

Schlüsselkandidaten = { {A,C} }

Schemata erstellen:

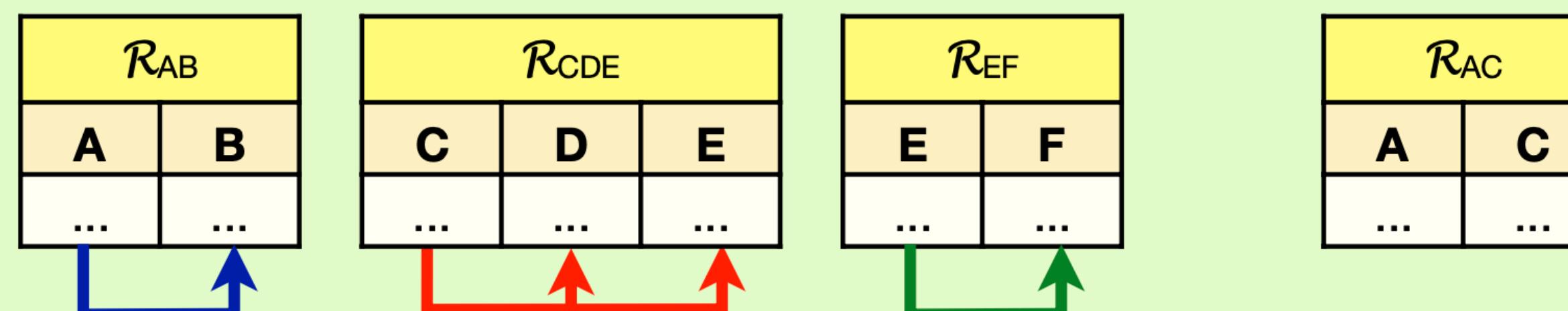
- $R_{AB} = \{A, B\}$ mit $F_{AB} = \{A \rightarrow B\}$
- $R_{CDE} = \{C, D, E\}$ mit $F_{CDE} = \{C \rightarrow DE\}$
- $R_{EF} = \{E, F\}$ mit $F_{EF} = \{E \rightarrow F\}$

Test auf Schlüsselkandidat

- Keine Relation enthält den einzigen Schlüsselkandidaten {A,C}
- $R_{AC} = \{A, C\}$ mit $F_{AC} = \emptyset$

Streichen von überflüssigen Relationen

- Hier nicht notwendig



Synthesealgorithmus

Beispiel 2

$$\mathcal{R} = \{A, B, C, D, E\}$$

$$\mathcal{F}_n = \{ ADE \rightarrow BE, B \rightarrow AC, D \rightarrow AC, ACE \rightarrow BD, C \rightarrow B \}$$

Schlüssel

- alle Attribut Werte ... (rechte Seite), d.h.

teste:

$$\text{wir brauche: } ? \rightarrow ABCDE \quad \text{min.}$$

$$\text{wir betrachten Attrib. Menge } (\mathcal{F}_n, ?) =: ?^+ \\ \text{also } A^+ = A$$

$$B^+ \rightarrow BACD, \text{ also } B^+ = ABCD$$

$$C \rightarrow CB \rightarrow ABCD$$

$$D \rightarrow DAC \rightarrow DACB \rightarrow ABCD$$

$$E \rightarrow E$$

"

- Feststeh.: E folgt nicht...
normal genauer hinweise: $ADE \rightarrow BE$, d.h.
ohne E kein E, also w/p E in der Schlüssel.

$$\begin{aligned} AE &\rightarrow AE \\ BE &\rightarrow ADCDE \quad \checkmark \\ CE &\rightarrow ABCDE \quad \checkmark \\ DE &\rightarrow ABCDE \quad \checkmark \end{aligned}$$

- BE, CE, DE Schlüsselnd. (nicht zu reduzieren)
Prim: B, C, DE , Nicht-Prim: A (w. $D \rightarrow A$ nicht zMF)

Synthesealgorithmus

Beispiel 2

Links red.

- Beinhaltet nur FDs mit mehr als einer Attributstr.

$$\underline{A \rightarrow DE} \rightsquigarrow \underline{BE}$$

A weglassen: $D\bar{E} \rightarrow D\bar{E}AC \rightarrow D\bar{E}ACB$
also $BE \subset D\bar{E}^+$

d.h. A nicht notwendig

$$\text{neue } F_n = \{ D\bar{E} \rightarrow BE, \dots \}$$

E weglassen: $D \rightarrow DAC \rightarrow DACB$ und $BE \not\subset ABC$
d.h. E notwendig

D weglassen: $E \rightarrow E$ und $BE \not\subset E$
d.h. D notwendig

- $\underline{ACE} \rightarrow BD$

A weglassen: $C\bar{E} \rightarrow C\bar{E}B \rightarrow C\bar{E}BA\bar{D}$ und $BD \subset ABC$
d.h. A nicht notwendig

neue $F_n = \{ \dots, C\bar{E} \rightarrow BD, \dots \}$

C weglassen: $E \rightarrow E$, d.h. $BD \not\subset E$ aber notwendig

E weglassen: $C \rightarrow CB \rightarrow CBA\bar{D}$ d.h. $BD \subset CAB$
also E nicht notwendig

neue $F_n = \{ \dots, C \rightarrow BD, \dots \}$

- flesant: $F_n = \{ D\bar{E} \rightarrow BE, B \rightarrow ACD, D \rightarrow AC,$
 $C \rightarrow BD, C \rightarrow B \}$

Synthesealgorithmus

Beispiel 2

Reduktion

betrachte $\tilde{F}_n = \{ DE \rightarrow B, DE \rightarrow E, B \rightarrow A, B \rightarrow C, \underline{B \rightarrow D}, \underline{D \rightarrow A}, \underline{D \rightarrow C}, C \rightarrow B, C \rightarrow D, C \rightarrow B \}$

$DE \rightarrow B$

Akh. Wille ($\tilde{F}_n \setminus \{ DE \rightarrow B \}, DE \right)$:

* $DE \rightarrow DEAC \rightarrow DEACB$ und $B \subset ABCDE$
 also $DE \rightarrow B$ nicht notwendig

neuer $\tilde{F}_n = \{ DE \rightarrow E, B \rightarrow A, \dots \}$ (ohne $DE \rightarrow B$)

$DE \rightarrow E$

Akh. Wille ($\tilde{F}_n \setminus \{ DE \rightarrow E \}, DE \right)$:

$DE \rightarrow DEAC \rightarrow DEACB \quad \text{und} \quad E \subset DEACB$

(also $DE \rightarrow E$ nicht notwendig)

(oft erschöpft)

neuer $\tilde{F}_n = \{ B \rightarrow A, B \rightarrow C, \dots \}$

$B \rightarrow A$

Akh. Wille ($\tilde{F}_n \setminus \{ B \rightarrow A \}, B \right)$:

$B \rightarrow BCD \rightarrow BCD\underline{A}$ also $A \subset BCDA$
 und $B \rightarrow A$ nicht notw.

neuer $\tilde{F}_n = \{ B \rightarrow C, B \rightarrow D, \dots \}$

Synthesealgorithmus

Beispiel 2

$B \rightarrow C$

Ath. Wölle ($\tilde{F}_n \setminus \{B \rightarrow C\}, B$):

$B \rightarrow B \rightarrow B \rightarrow A \rightarrow C$ d.h. $C \in BDAC$

und $B \rightarrow C$ nicht wahr.

neu: $\tilde{F}_n = \{B \rightarrow D, D \rightarrow A, \dots\}$

$B \rightarrow D$

Ath. Wölle ($\tilde{F}_n \setminus \{B \rightarrow D\}, B$):

$B \rightarrow B \rightarrow D$ als $B \rightarrow D$ wahr.

$D \rightarrow A$

Ath. Wölle ($\tilde{F}_n \setminus \{D \rightarrow A\}, D$):

$D \rightarrow D \rightarrow C \rightarrow D \rightarrow B$ also $D \rightarrow A$ wahr. ($A \notin DCB$)

$D \rightarrow C$

Ath. Wölle ($\tilde{F}_n \setminus \{D \rightarrow C\}, D$):

$D \rightarrow D \rightarrow A$ also $D \rightarrow C$ wahr. ($C \notin DA$)

$C \rightarrow B$

doppelt, also strikt

neu: $\tilde{F}_n = \{B \rightarrow D, D \rightarrow A, D \rightarrow C, C \rightarrow D, C \rightarrow B\}$

$C \rightarrow B$

Ath. Wölle ($\tilde{F}_n \setminus \{C \rightarrow B\}, C$):

$C \rightarrow CB \rightarrow CBD$ also nicht wahr. ($D \subset CB$)

neu: $\tilde{F}_n = \{B \rightarrow D, D \rightarrow A, D \rightarrow C, C \rightarrow B\}$

$C \rightarrow B$

Ath. Wölle ($\tilde{F}_n \setminus \{C \rightarrow B\}, C$):

$C \rightarrow C$ also wahr.

Synthesealgorithmus

Beispiel 2

- nach Mult. red.

$$\widehat{\mathcal{F}}_n = \{ B \rightarrow D, \overset{\uparrow}{D} \rightarrow AC, C \rightarrow B \}$$

mit. gefaßt

= Kanonisch Überdeckung

- M. schreibe

$$\mathcal{R}_1 = \{ \cancel{B \rightarrow D} \} \quad \mathcal{F}_1 = \{ B \rightarrow D \}$$

$$\mathcal{R}_2 = \{ \cancel{D \rightarrow AC} \} \quad \mathcal{F}_2 = \{ D \rightarrow AC \}$$

$$\mathcal{R}_3 = \{ BC \} \quad \mathcal{F}_3 = \{ C \rightarrow B \}$$

- mind. ei Sch. Kandidat att. ? (BE, CE, DE)

Nei, also himmfig, d.h.

$$\mathcal{R}_4 = \{ BE \}, \quad \mathcal{F}_4 = \{ \} \quad (\text{Sch. Kandidat beliebig})$$

Boyce-Codd Normalform

Motivation

- Alle Attribute sind prim. Damit ist 3NF erfüllt...

\mathcal{R}		
PLZ	Ort	Strasse
52080	Aachen	Von-Coels-Str.
52080	Aachen	Karlsstr.



\mathcal{R} ist in 3. Normalform mit folgenden FDs

- ▶ 1: $\{PLZ\} \rightarrow \{Ort\}$
- ▶ 2: $\{Ort, Strasse\} \rightarrow \{PLZ\}$

alle Attribute sind prim!

Schlüsselkandidaten

Linke Seiten der FDs: $\{Ort, Strasse, PLZ\}$ ist offenbar Superschlüssel

Linksreduktion (Beachte: nicht deterministisch!)

- Variante 1: $\{Strasse, PLZ\} \xrightarrow{1} \{Strasse, PLZ, Ort\}$ - keine weitere Reduktion möglich
- Variante 2: $\{Ort, Strasse\} \xrightarrow{2} \{Ort, Strasse, PLZ\}$ - keine weitere Reduktion möglich

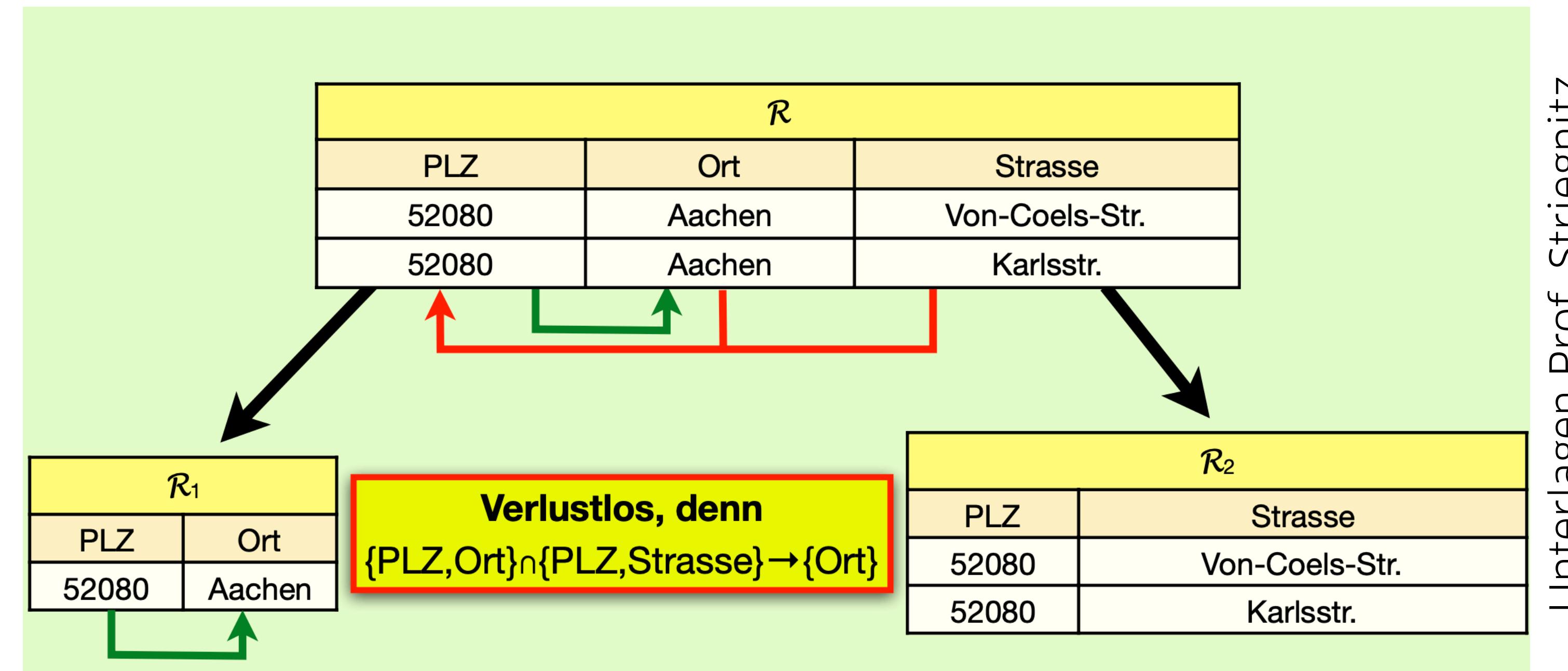
2 Schlüsselkandidaten: $\{Strasse, PLZ\}$ und $\{Ort, Strasse\}$

Die 3NF verhindert offenbar nicht alle Redundanzen.



Boyce-Codd Normalform

Motivation



Unterlagen Prof. Striegnitz

Boyce-Codd Normalform

Definition

- Ein Schema R mit einer Menge funktionaler Abhängigkeiten FD ist in Boyce-Codd Normalform (BCNF), wenn für jede funktionale Abhängigkeit $\alpha \rightarrow \beta \in FD^+$ mindesten eine der folgenden Bedingungen gilt:
 - $\alpha \rightarrow \beta$ ist trivial, also $\beta \subseteq \alpha$,
 - α ist Superschlüssel.

Beim Test auf BCNF testet man, wie bei 3NF, einfach alle Abhängigkeiten, ob eine der zwei Bedingungen zutrifft.



Anmerkungen

- Vgl. 3NF, bei BCNF fällt die zweite Bedingung weg.
- Die dritte Normalform ist eingeschlossen.

Boyce-Codd Normalform

Anmerkungen

- Man kann jedes Schema R mit funktionalen Abhängigkeiten FD so in Schemata R_1, \dots, R_n zerlegen, dass BCNF erfüllt ist.



Die Zerlegung nicht notwendigerweise abhängigkeitserhaltend!

- Der folgende *Dekompositionsalgorithmus* setzt genau das um und durch die Aufteilung der Relationen können funktionale Abhängigkeiten, wie im motivierenden Beispiel, verloren gehen.
- Die Voraussetzung $\alpha \rightarrow \beta \in FD^+$ ist wichtig, denn nicht immer sind alle Abhängigkeiten explizit gegeben - siehe Beispiele.

Dekompositionsalgorithmus

Definition

- **Starte mit $Z = \{\mathcal{R}\}$**
- **Solange es noch ein Relationenschema \mathcal{R}_i in Z gibt, das nicht in BCNF ist, mache folgendes:**

Anmerkung: Es gibt also eine für \mathcal{R}_i geltende funktionale Abhängigkeit $\alpha \rightarrow \beta$ mit $\alpha \cap \beta = \emptyset$ (nicht trivial) und $\neg(\alpha \rightarrow \mathcal{R}_i)$ (α kein Superschlüssel)

 - **Finde eine solche FD**

Anmerkung: Man sollte die FD so wählen, dass β alle von α funktional abhängigen Attribute $b \in (\mathcal{R}_i - \alpha)$ enthält, damit der Dekompositionsalgorithmus möglichst schnell terminiert.
 - **Zerlege \mathcal{R}_i in $\mathcal{R}_{i1} = \alpha \cup \beta$ und $\mathcal{R}_{i2} := \mathcal{R}_i - \beta$**

Die FDs verteilen sich entsprechend - kommen die Attribute in einer FD in keiner Relation mehr geschlossen vor, entfällt diese FD → Verlust von Abhängigkeiten!
 - **Entferne \mathcal{R}_i aus Z und füge \mathcal{R}_{i1} und \mathcal{R}_{i2} ein**

also setze $Z = (Z - \{\mathcal{R}_i\}) \cup \{\mathcal{R}_{i1}\} \cup \{\mathcal{R}_{i2}\}$

Dekompositionsalgorithmus

Beispiel 1

Bsp. BCNF

$$\mathcal{R} = \{ A, B, C, D \}$$

$$\mathcal{F}_R = \{ BC \rightarrow D, AB \rightarrow A \}$$

Schlüssel? ABC

Dekompositionsalg. (keine kan. Übereinst.)

Test ob für $\alpha \rightarrow \gamma$ entweder 1. $\gamma \subseteq \alpha$ oder 2. α SuperschlüsselFinde $\alpha \rightarrow \gamma$, die eine Bed. reicht ...

$$BC \rightarrow D$$

1. $D \not\subseteq BC$ 2. BC kein Superschl.

also

$$\mathcal{R}_1 = \underbrace{BCD}_{\alpha \quad \gamma}$$

$$\mathcal{F}_{\mathcal{R}_1} = \{ BC \rightarrow D \}$$

$$\mathcal{R}_2 = ABCD \setminus D = ABC$$

$$\mathcal{F}_{\mathcal{R}_2} = \{ AB \rightarrow A \}$$

Dekompositionsalgorithmus

Beispiel 2

Bsp. BCNF

$$\mathcal{R} = \{ A, B, C, D \}$$

$$\mathcal{F}_R = \{ C \rightarrow A, BD \rightarrow AC, C \rightarrow B \}$$

keine Schlüsselknd.

$$\begin{matrix} CD \\ BD \end{matrix}$$

Test BCNF:

$$\begin{aligned} C \rightarrow A : & \quad A \notin C \text{ also } \times \\ & C \text{ ke Superkl. also } \times \end{aligned}$$

also

$$\mathcal{R}_1 = AC$$

$$\mathcal{F}_{\mathcal{R}_1} = \{ C \rightarrow A \}$$

$$\mathcal{R}_2 = BD$$

achtg

$$\mathcal{F}_{\mathcal{R}_2} = \{ BD \rightarrow C \\ C \rightarrow B \}$$

↑ BCNF?

 \mathcal{R}_2 : End. Schlüssel BD,also für $C \rightarrow B$ ist C ke Superkl.d.h. \mathcal{R}_2 reicht BCNF

also

$$\mathcal{R}_{21} = BC$$

$$\mathcal{F}_{\mathcal{R}_{21}} = \{ C \rightarrow B \}$$

$$\mathcal{R}_{22} = \{ C \}$$

$$\mathcal{F}_{\mathcal{R}_{22}} = \{ \}$$

wert

Dekompositionsalgorithmus

Beispiel 3

Bsp. $\overline{FD} = \{ B \rightarrow A, D \rightarrow ACE, AC \rightarrow DE, D \rightarrow AB \}$
 $R = ABCDE$

$B \rightarrow A$ verletzt BCNF

$$\begin{array}{ll} \downarrow & \longrightarrow \\ R_1 = A \underline{B} & R_2 = R - A = \underline{BCDE} \\ FD_1 = \{ B \rightarrow A \} & FD_2 = \{ D \rightarrow BCE \} \end{array}$$

Verlust: $AC \rightarrow DE$

Dekompositionsalgorithmus

Beispiel 4

Bsp.

$$\text{FD} = \{ACD \rightarrow E, ABE \rightarrow CD, AC \rightarrow DE, BC \rightarrow E\}$$

$$\mathcal{R} = ABCDE$$

$$\text{SL. knd. : } A\bar{B}C, A\bar{B}E$$

$$\text{Syllese : } \text{FD}^c = \{ABE \rightarrow C, AC \rightarrow DC, BC \rightarrow E\}$$

$$\mathcal{R}_1 = \underline{ABC}\bar{E} \text{ ob } \underline{ABC}\bar{E}, \quad \text{FD}_1 = \{A\bar{B}E \rightarrow C\}$$

$$\mathcal{R}_2 = \underline{ACD}\bar{E}, \quad \text{FD}_2 = \{AC \rightarrow DE\}$$

$$\mathcal{R}_3 = \underline{BC}\bar{E}, \quad \text{FD}_3 = \{BC \rightarrow E\}$$

hier $\mathcal{R}_3 \subset \mathcal{R}_1$ also \mathcal{R}_3 weglassen (d FD_3 in FD_1)

Dekomp :

$ACD \rightarrow E$ verletzt BCNF (ACD ist Superfl.)



$$\mathcal{R}_1 = \underline{ACD}\bar{E}$$

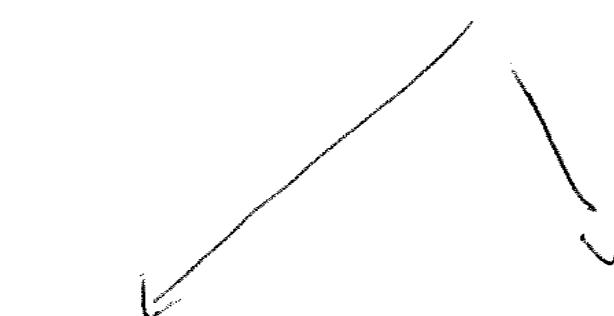
$$\text{FD}_1 = \{ACD \rightarrow E, AC \rightarrow DE\}$$

$$\mathcal{R}_2 = \mathcal{R} - E = \underline{ABCD}$$

$$\text{FD}_2 = \{AC \rightarrow D\}$$

(neue knd. SL. folgt aus
neuer Syllese)

hier: $AC \rightarrow D$ verletzt
BCNF
da AC ist
Superfl.



$$\mathcal{R}_{21} = \underline{ACD}$$

$$\text{FD}_{21} = \{AC \rightarrow D\}$$

$$\mathcal{R}_{22} = ABCD - D = \underline{ABC}$$

$$\text{FD}_{22} = \{\}$$

Dekompositionsalgorithmus

Beispiel 5

Bsp.

$$FD = \{ A \rightarrow B, B \rightarrow DE, C \rightarrow B, E \rightarrow F, AC \rightarrow BD \}$$

bcd. s.t. AC

$A \rightarrow B$ verletzt BCNF

$$R_1 = \underline{AB}$$

$$FD_1 = \{ A \rightarrow B \}$$

$$R_2 = \underline{ACDEF}$$

$$FD_2 = \{ \underline{E} \rightarrow F, AC \rightarrow D, \\ A \rightarrow DEF, \\ C \rightarrow DEF, \\ AC \rightarrow EF \}$$

sub (e FD)

hau'kr

$E \rightarrow F$ verletzt BCNF

$$R_{21} = \underline{EF}$$

$$FD_{21} = \{ E \rightarrow F \}$$

$$R_{22} = \underline{ACDE}$$

$$FD_{22} = \{ AC \rightarrow D, \\ A \rightarrow DE, \\ C \rightarrow DE, \\ AC \rightarrow E \}$$

$$R_{21} = \underline{EF}$$

$$FD_{21} = \{ E \rightarrow F \}$$

$$R_{22} = \underline{ACDE}$$

$$FD_{22} = \{ AC \rightarrow D, \\ A \rightarrow DE, \\ C \rightarrow DE, \\ AC \rightarrow E \}$$

$A \rightarrow DE$ verletzt BCNF

$$R_{221} = \underline{ADE}$$

$$FD_{221} = \{ A \rightarrow DE \}$$

$$R_{222} = \underline{AC}$$

$$FD_{222} = \{ \}$$

Finale

Aufgabe

- Entwicklung einer 'guten' Datenbank [...] u.a. minimal, redundanzfrei [...]

Vorgehen

- Konzeptueller Entwurf [...] ER-Modells ✓
- Überführung in eine 'gute' Datenbank [...] dazu benötigen wir:
 - Implementationsentwurf [...] → Relationales Modell ✓
 - Mathematisches Modell [...] → Relationale Algebra ✓
 - Gütemaß → Normalformen (Anomalien, Verlustlosigkeit, Abhängigkeitserhaltung)
 - ♦ Kanonische Überdeckung ✓, funktionale Abhängigkeiten ✓, Attributhülle ✓,
 - ♦ Synthesealgorithmus ✓,
 - ♦ Dekompositionsalgorithmus ✓.

Finale

Schlussbemerkungen

- Wie am Beispiel der Artikeldatenbank gezeigt, kann man *mit Verständnis für die Problematik von Anomalien und Redundanzen* sowohl aus einer großen Tabelle, funktionalen Abhängigkeiten und Normalisierung (Universal Relation Assumption), als auch über gut modellierte ER-Diagramme und der Umsetzung in die Relationen, einen guten Datenbankentwurf, d.h. die konkreten Tabellen, ableiten!
- Nicht immer ist eine Normalisierung bis ins Kleinste anzustreben:
 - Durch die Aufteilung der Daten entsteht Overhead im Datenhandling (Joins), der Durchsatz kann leiden, etc. Hier können bewusst eingesetzte Redundanzen zur Performancesteigerung beitragen.
 - SQL-Integritätsbedingungen können ebenfalls helfen, Anomalien zu vermeiden.

Wie immer Sie modellieren - beide Ansätze zu Kennen ist wichtig!



Typische Prüfungsaufgabe

Aufgabe

- Ermitteln Sie für das vorliegende Schema alle Schlüsselkandidaten.
- Prüfen Sie, in welcher Normalform das Schema vorliegt.
- Entweder: Überführen Sie es mit dem Synthesealgorithmus in 3NF, falls notwendig.
- Oder: Überführen Sie es mit dem Dekompositionsalgorithmus in BCNF, falls notwendig.

Anmerkung

Man findet im Internet viele "Algorithmen" zur "schnellen und unkomplizierten" Berechnung aller Schlüsselkandidaten... Folgt man aber der Literatur, etwa Lucchesi, Osborn (1978). *Candidate keys for relations. Journal of Computer and System Sciences*, 17, so stellt sich heraus, dass das Ermitteln aller Schlüsselkandidaten im worst case exponentielle Komplexität besitzt. Also Vorsicht vor falschen Rezepten.

Beispiel

$$R = \{A, B, C, D, E, F\}$$

①

$$FD_S = \{BCE \rightarrow D, B \rightarrow CEF, DE \rightarrow BC\bar{F}\}$$

immer benötigt: Schlüsselkandidaten \rightsquigarrow A fehlt auf rechter Seite,
d.h. kann nicht gefolgt werden,
muss also in Schlüssel \triangleright

- Alt. Menge (FD, AB) = $ABC\bar{D}EF \vee$
(kurz: $AB \rightarrow ABC\bar{E}F \rightarrow ABCDEF$) also AB Schlüsselkandidat \triangleright
- AC, AD, \dots braucht nicht präsent zu werden, da aus C, D, etc. nichts folgt.
- ADE ist die nächste sinnvolle Kombination (siehe FD)
also $ADE \rightarrow ABCDEF \vee$ also ADE Schlüsselkand. \triangleright
- $ABC\bar{E}$ Superschlüssel
- wg Struktur von FD keine weiteren Schlüsselkandidaten
- also AB, ADE Schlüsselkand., A, B, D, E prim
 C, F nicht-prim

Beispiel

Welche NF liegt vor?

NF1 ✓

NF2: wg. $B \rightarrow CEF$ und
 AB Schlüsselkandidat,
 insbes. $B \rightarrow C$, mit
 C nicht-prim, liegt
 NF2 nicht vor.

NF3, BCNF wg. fehlede NF2
 auch nicht ?

Kanonische Überdeckung für Synthesealg.:

Linksnormalisierung:

$BCE \rightarrow D$: formal naheinander C, E^-
 streichen, aber wg. $B \rightarrow CEF$
 sieht man darunter
 beide streichen kann ✓
 $B \rightarrow BCEF \rightarrow BCD^+EF$
 also durch $B \rightarrow D$ ersetzen

- | $B \rightarrow CEF$ und nicht
unterstellt werden (2)
- | $DE \rightarrow BCF$: wg. $D \rightarrow D$,
 $E \rightarrow E^-$
leider nichts geschieht werden
- | also $FD' = \{B \rightarrow D, B \rightarrow CEF,$
 $DE \rightarrow BCF\}$
- | Reduktionsreduzibilität:
 - | $B \rightarrow D$: formal
Alt. Nullte ($FD' \setminus \{B \rightarrow D\}$, B)
 $= BCEF \not\ni D$
 - | oder: ohne $B \rightarrow D$ leidet D nicht
unter, also $B \rightarrow D$ bleibt
- | $B \rightarrow CEF$:
 - | Alt. Nullte ($FD' \setminus \{B \rightarrow CEF\} \cup \{B \rightarrow CE\}$, B)
 $= BCDEF \ni CEF$
 - | also F überflüssig, $FD'' = \{B \rightarrow D,$
 $B \rightarrow CE,$
 $DE \rightarrow BCF\}$
- | $B \rightarrow CE$: ...

Beispiel

$B \rightarrow CE$:

$$\text{Attr. Hülle } (FD)^H \setminus \{B \rightarrow CE\} \cup \{B \rightarrow C\}, B \\ = BCD \not\Rightarrow CE$$

also E notwendig

$$\text{Attr. Hülle } (FD)^H \setminus \{B \rightarrow CE\} \cup \{B \rightarrow E\}, B \\ = BCDEF \not\Rightarrow CE$$

also C überflüssig

$$FD^H = \{B \rightarrow D, B \rightarrow E, DE \rightarrow BCF\}$$

$DE \rightarrow BCF$:

$$\text{Formal Attr. Hülle } (FD)^H \setminus \{DE \rightarrow BCF\} \\ \cup \{DE \rightarrow \dots\}, DE \\ = \dots$$

Oder: Weder B , noch C , noch F

folgt aus $\overline{FD}^H \setminus \{DE \rightarrow BCF\}$

also alle notwendig

$$\Rightarrow \overline{FD}^C = \{B \rightarrow DE, DE \rightarrow BCF\}$$

| Synthesearc.

$$|\bullet R_1 = \{BDE\}, FD_1 = \{B \rightarrow DE, DE \rightarrow B\}$$

$$|\bullet R_2 = \{BCDEF\}, FD_2 = \{B \rightarrow DE, DE \rightarrow BCF\}$$

$$|\bullet \text{ Weder } AD \text{ noch } ADE \text{ in Schenata,} \\ |\bullet \text{ also } R_3 = \{AB\}, FD_3 = \{\}$$

| $n_1 \subset n_2$, also final:

$$R_1 = \{BCDEF\}$$

$$FD_1 = \{B \rightarrow DE, DE \rightarrow BCF\}$$

$$n_2 = \{AB\}$$

$$FD_2 = \{\}$$

(3)

Beispiel

Dekompositionsalg.

$$FD = \{ BCE \rightarrow D, B \rightarrow CEF, DE \rightarrow BCF \}$$

- Teste fktl. Abh. auf BCNF Kriterie:

$BCE \rightarrow D$: BCE kein Superkey,
also $BCNF$ verletzt

$$R_1 = \{ BCDE \}, FD_1 = \{ BCE \rightarrow D, B \rightarrow CE, DE \rightarrow BC \}$$

$$R_2 = \{ ABCEF \}, FD_2 = \{ B \rightarrow CEF \}$$

- Anmerkung: weitere fktl. Abh. aus da DB-Normalisierung unnötig

- unterscheide nur R_1, R_2

- R_1 : Schlüsselknd.: B und DE
 \Rightarrow nur Superkey auf linken Seiten, also BCNF liegt vor

R_2 : Schlüsselknd. AB
 \uparrow nicht superkey
 $\Rightarrow B \rightarrow CEF$ verletzt $BCNF$
 da B kein Superkey

- Auflösung von R_2 :

$$R_{21} = \{ BCEF \}, FD_{21} = \{ B \rightarrow CEF \}$$

Schlüssel B , und somit $BCNF$

$$R_{22} = \{ AB \}, FD_{22} = \{ \}$$

Final: R_1, R_{21}, R_{22}
 mit FD_1, FD_{21}, FD_{22}

