Datenbanken

Relationale Entwurfstheorie

Dr. Özgür Özçep

Prof. Dr. Ralf Möller

Universität zu Lübeck

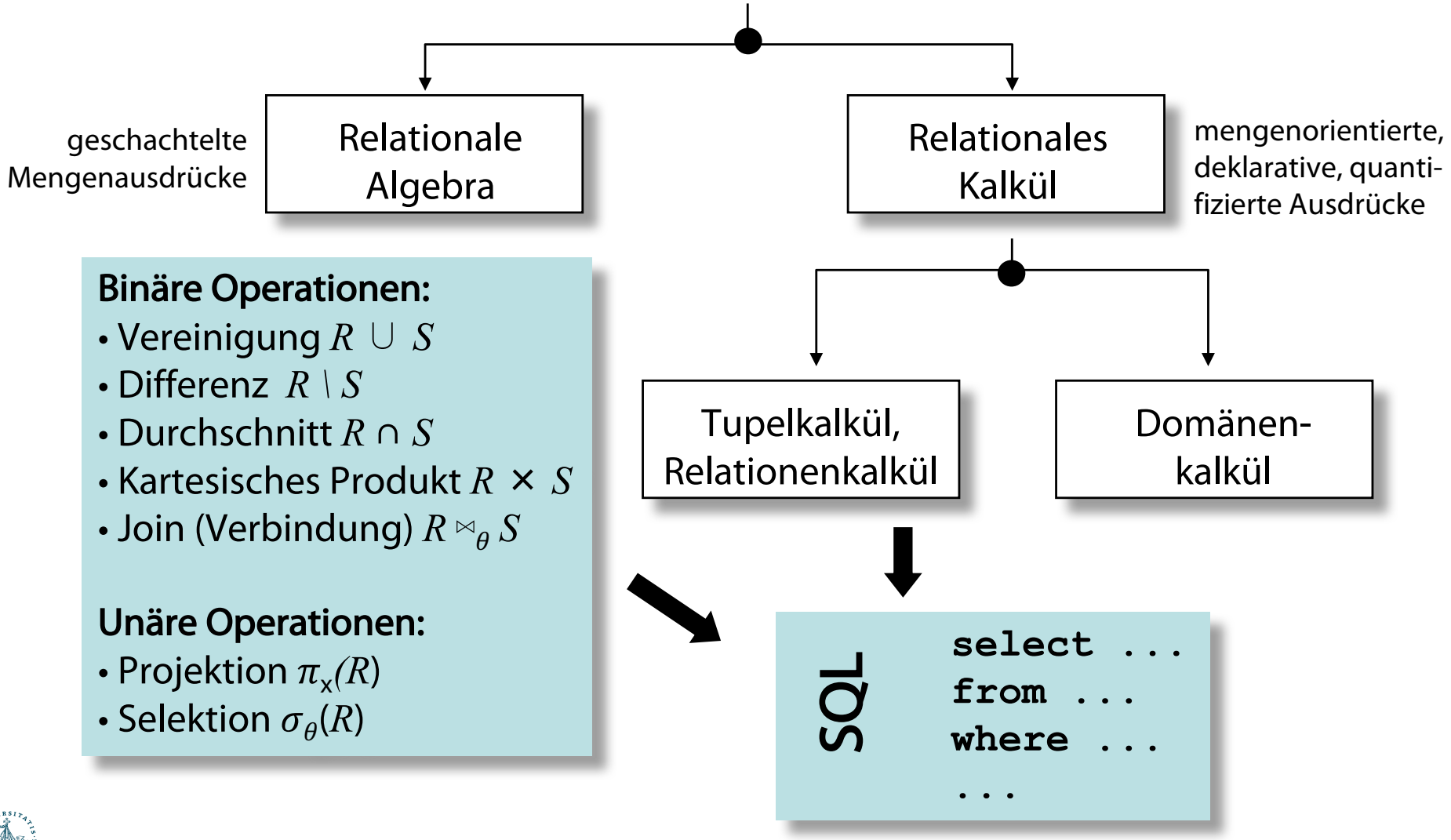
Institut für Informationssysteme

Felix Kuhr (Übungen)

und studentische Tutoren

RDM: Anfragen

Relationale Anfragesprachen im Überblick:



Binäre Operationen:

- Vereinigung $R \cup S$
- Differenz $R \setminus S$
- Durchschnitt $R \cap S$
- Kartesisches Produkt $R \times S$
- Join (Verbindung) $R \bowtie_\theta S$

Unäre Operationen:

- Projektion $\pi_x(R)$
- Selektion $\sigma_\theta(R)$

Relationale Algebra in SQL

- Projektion
  - SELECT col_1, col_2, …
    // oder * für alle Attribute
    FROM *tab*
- Selektion
  - SELECT *
    FROM ...
    WHERE *cond*
- Verbund/Join
  - SELELCT *
    FROM tab_1 JOIN tab_2
    ON *col1* = *col2*
- Umbenennung/Renaming
  - SELECT *col* AS *new_col* , …
    FROM …

- Sonstige
  - tab_1 UNION tab_2
  - tab_1 EXCEPT tab_2
  - tab_1 INTERSECT tab_2

- Kreuzprodukt
  - SELECT *
    FROM tab_1, tab_2, ..., tab_n

Kandidatenschlüssel

- Sei die Menge der Kandidatenschlüssel für $\mathcal{R}$ gegeben
  - $\mathcal{R} = \{ \{A, B\}, \{B, C\} \}$
  - Wähle Primärschlüssel, z.B. {A, B}
  - Keine Nullwerte als Primärschlüsselwerte
- Deklariere Tabelle

```
create table R
 (A ...,
  B ...,
  C ...,
  ...
  primary key (A, B),
  unique (B, C) )
```

Schlüssel {A,B}

Eindeutigkeit für {B, C}

Schlüsselbestimmung

NB: Funktionale Unabhängigkeiten verstanden als eigenständige Entitäten– unabhängig von einer konkreten Relation (von einem Relationsschema)

Städte			
Name	BLand	Vorwahl	EW
Frankfurt	Hessen	069	650000
Frankfurt	Brandenburg	0335	84000
München	Bayern	089	1200000
Passau	Bayern	0851	50000
...	...	...	...

Kandidatenschlüssel von *Städte*:

- {Name,BLand}
- {Name,Vorwahl}

Kandidatenschlüssel lassen sich nicht aus Beispielen bestimmen!

Funktionale Abhängigkeiten zählen.

Schlüssel sollen FDs umsetzen!

Beachte, dass 2 kleinere Städte dieselbe Vorwahl haben können

Hülle Funktionaler Abhängigkeiten

Sei F eine Menge von Funktionalen Abhängigkeiten (FDs)

F^+ bezeichnet die Menge aller aus F ableitbaren FDs und wird Hülle genannt.

Im allgemeinen gibt es unterschiedliche Mengen von FDs, deren Hülle gleich sind.

In diesem Fall schreiben wir: $F_1 \equiv F_2$
(F_1 und F_2 sind äquivalent)

Herleitung von F^+: Armstrong-Axiome

Reflexivität

- Falls β eine Teilmenge von α ist ($\beta \subseteq \alpha$), dann gilt immer $\alpha \longrightarrow \beta$.
  Insbesondere gilt immer $\alpha \longrightarrow \alpha$.

Verstärkung

- Falls $\alpha \longrightarrow \beta$ gilt, dann gilt auch $\alpha\gamma \longrightarrow \beta\gamma$. Hierbei stehe z.B. $\alpha\gamma$ für $\alpha \cup \gamma$.

Transitivität

- Falls $\alpha \longrightarrow \beta$ und $\beta \longrightarrow \gamma$ gilt, dann gilt auch $\alpha \longrightarrow \gamma$.

Diese drei Axiome sind ausreichend zur Herleitung sämtlicher FDs

Zusätzliche Axiome erleichtern die Herleitung von FDs:

- Vereinigung:
  - Wenn $\alpha \longrightarrow \beta$ und $\alpha \longrightarrow \gamma$ gelten, dann gilt auch $\alpha \longrightarrow \beta\gamma$
- Dekomposition:
  - Wenn $\alpha \longrightarrow \beta\gamma$ gilt, dann gelten auch $\alpha \longrightarrow \beta$ und $\alpha \longrightarrow \gamma$
- Pseudotransitivität:
  - Wenn $\alpha \longrightarrow \beta$ und $\gamma\beta \longrightarrow \delta$, dann gilt auch $\alpha\gamma \longrightarrow \delta$

William W. Armstrong, Dependency Structures of Data Base Relationships, IFIP Congress, 1974

Armstrong Axiome

- Der Kalkül, der sich durch sättigende Anwendung der Armstrong-Axiome ergibt, ist korrekt und vollständig
  - Korrektheit
    - Bedeutet hier: Höchstens echt geltenden FDs werden abgeleitet.
    - Beweis (meist) einfach
  - Vollständigkeit
    - Bedeutet hier: Mindestens alle geltenden FDs werden abgeleitet
    - Beweis (meist) nicht einfach

William W. Armstrong, Dependencse Relationships, IFIP Congress, 1974

C. Beeri, M. Dowd, R. Fagin, R. Statman. On the Structure of Armstrong Relations for Functional Dependencies, Journal of the ACM 31, pp. 30–46, 1984

Schlüsselbestimmung

Manuelle Bestimmung der Kandidatenschlüssel

- ist aufwendig und
- bei vielen FDs fehleranfällig.

Automatisierbar?

Bestimmung der Hülle einer **Attribut**menge

Bei der Schlüsselbestimmung ist man nicht an der gesamten Hülle einer Menge F von FDs interessiert, sondern nur an der Menge von Attributen, die von α gemäß F funktional bestimmt werden (sog. Attributhülle).

Eingabe: eine Menge F von FDs und eine Menge von Attributen α.

Ausgabe: die vollständige Menge von Attributen α^+, für die gilt $\alpha \longrightarrow \alpha^+$.

```
AttrHülle(F,α)
    erg := α ; erg' := ∅
    while  erg' ≠ erg do
        erg' := erg
        foreach β ⟶ γ ∈ F do
            if  β ⊆ erg then
                erg := erg ∪  γ
    return erg                          // Attributhülle α+
```

Nutzen der Attributhülle

- Bestimmung, ob Menge von Attributen κ einen Superschlüssel für $\mathcal{R}$ darstellt:
  - Bestimme κ^+ und prüfe ob $\kappa^+ = \mathcal{R}$

- Kandidatenschlüssel für ein Relationenschema bestimmen:
  - Bestimme alle bzgl. Mengeninklusion minimalen Mengen κ, so dass $\kappa^+ = \mathcal{R}$, und damit $\kappa \longrightarrow \mathcal{R}$

Herleitung von Relationenschemata aus FDs

- {PersNr} ⟶ {PersNr, Name, Rang, Raum, Ort, Straße, PLZ, Vorwahl, Bland, EW, Landesregierung}
- {Ort,BLand} ⟶ {EW, Vorwahl}
- {PLZ} ⟶ {Bland, Ort, EW}
- {Bland, Ort, Straße} ⟶ {PLZ}
- {Bland} ⟶ {Landesregierung}
- {Raum} ⟶ {PersNr}

Welche Relationenschemata sollen verwendet werden,
so dass FDs durch Schlüsselbedingung geprüft werden können?

- Professoren: {[PersNr, Name, Rang, Raum, Ort, Straße, PLZ, Vorwahl, Bland, EW, Landesregierung]} ???
- Für jede FD ein Schema???

Vermeidung von Redundanz

- In der Modellierung:
  - Sind einige der aufgeschriebenen FDs überflüssig?
- In den Daten:
  - Können wir doppelt repräsentierte Daten vermeiden?

Redundanzfreie Darstellung von FDs

F_c heißt kanonische Überdeckung von F, wenn die folgenden Kriterien erfüllt sind:

1. $F_c \equiv F$, d.h. $F_c^+ = F^+$

2. In F_c existieren keine FDs, die überflüssige Attribute enthalten.
   D.h. für $\alpha \longrightarrow \beta \in F_c$ muss gelten:

   - $\forall A \in \alpha: (F_c \setminus \{\alpha \longrightarrow \beta\}) \cup \{(\alpha \setminus \{A\}) \longrightarrow \beta\} \not\equiv F_c$
   - $\forall B \in \beta: (F_c \setminus \{\alpha \longrightarrow \beta\}) \cup \{\alpha \longrightarrow (\beta \setminus \{B\})\} \not\equiv F_c$

3. Jede linke Seite einer funktionalen Abhängigkeit in F_c ist einzigartig.
   (Erreichbar durch sukzessive Anwendung der Vereinigungsregel auf FDs der Art $\alpha \longrightarrow \beta$ und $\alpha \longrightarrow \gamma$, so dass beide FDs durch $\alpha \longrightarrow \beta\gamma$ ersetzt werden.)

Berechnung der kanonischen Überdeckung

- Führe für jede FD $\alpha \longrightarrow \beta \in F$ die Linksreduktion durch, also:

  Überprüfe für alle $A \in \alpha$, ob A überflüssig ist, d.h., ob

  $\beta \subseteq$ AttrHülle(F, $\alpha \setminus \{A\}$) gilt

  Falls dies der Fall ist, ersetze $\alpha \longrightarrow \beta$ durch $(\alpha \setminus \{A\}) \longrightarrow \beta$
- Führe für jede (verbliebene) FD die Rechtsreduktion durch, also:

  Überprüfe für alle $B \in \beta$, ob

  $B \in$ AttrHülle($(F \setminus \{\alpha \longrightarrow \beta\}) \cup \{\alpha \longrightarrow (\beta \setminus \{B\})\}, \alpha$) gilt

  Falls dies der Fall ist, ist B auf der rechten Seite überflüssig und

  Kann eliminiert werden, d.h. ersetze $\alpha \longrightarrow \beta$ durch $\alpha \longrightarrow (\beta \setminus \{B\})$
- Entferne FDs der Form $\alpha \longrightarrow \varnothing$, die im 2. Schritt möglicherweise entstanden sind
- Fasse FDs der Form $\alpha \longrightarrow \beta_1, \ldots, \alpha \longrightarrow \beta_n$ zusammen, so dass $\alpha \longrightarrow (\beta_1 \cup \ldots \cup \beta_n)$ verbleibt

Nutzung der kanonischen Überdeckung

Naiver Ansatz:

- Bilde relationales Schema für jede FD der kanonischen Überdeckung
    - Eventuell immer noch zu viele Relationen
    - Beispiel:
        - FDs = {A ⟶ BCD, D ⟶ ABC}
        - Zwei Relationen?
- Mehrere FDs einem relationalen Schema zuordnen?

Vereinbarungen

- FDs, die von jeder Relationenausprägung automatisch immer erfüllt werden, nennen wir **trivial**.
  Nur FDs der Art $\alpha \longrightarrow \beta$ mit $\beta \subseteq \alpha$ sind trivial.

- Attribute eines Kandidatenschlüssels heißen **prim**.
  Alle anderen Attribute nennen wir nicht prim.

- Sei $\mathcal{R}$ ein Relationenschema, dann ist
  $F_{\mathcal{R}}$ die zugeordnete Menge von FDs.
  Wenn $\mathcal{R}$ klar ist, dann wird meist nur F geschrieben.

Vermeidung von Redundanz in den Daten

Beispiel:

$\mathcal{R}$ = {[A, B, C ,D]}, F = {A $\longrightarrow$ B, D $\longrightarrow$ ABCD}, Schlüsselkandidat: {D}

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

„Do not represent the same fact twice“

Allgemeiner Fall: $\alpha \longrightarrow \beta \in F$, dann: α Superschlüssel oder FD ist trivial

ggf. Dekomposition notwendig (verlustfrei und abhängigkeitsbewahrend)

Zerlegung (Dekomposition) von Relationen

Korrektheitskriterien für die Zerlegung von Relationenschemata:

- Verlustlosigkeit

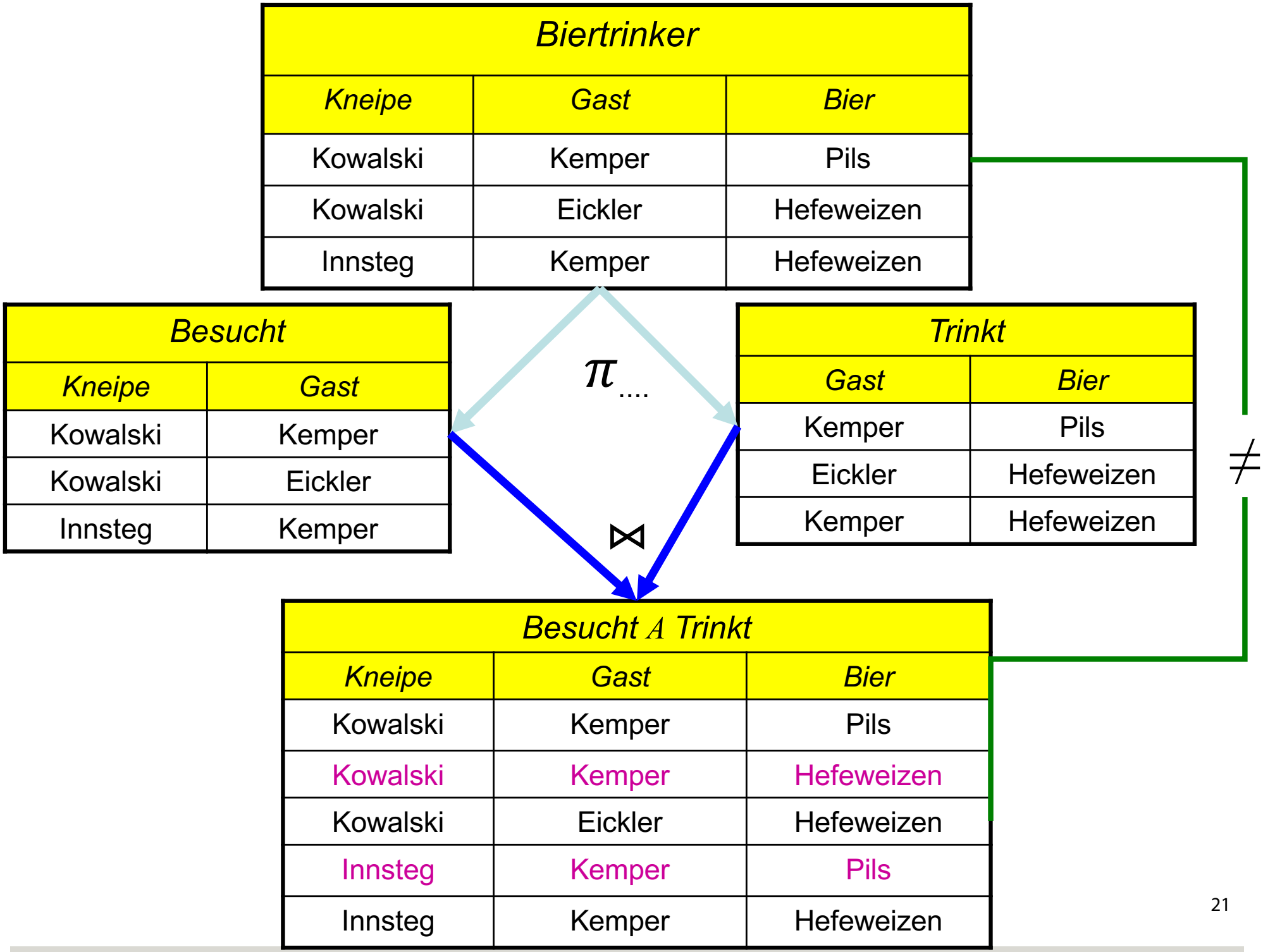
  Die in der ursprünglichen Relationenausprägung R des Schemas $\mathcal{R}$ enthaltenen Informationen müssen aus den Ausprägungen $R_1, ..., R_n$ der neuen Relationenschemata $\mathcal{R}_1, .., \mathcal{R}_n$ rekonstruierbar sein.

- Abhängigkeitserhaltung

  Die für $\mathcal{R}$ geltenden funktionalen Anhängigkeiten $F_{\mathcal{R}}$ müssen auf die Schemata $\mathcal{R}_1, ..., \mathcal{R}_n$ übertragbar sein.

Biertrinker-Beispiel

Biertrinker		
Kneipe	*Gast*	*Bier*
Kowalski	Kemper	Pils
Kowalski	Eickler	Hefeweizen
Innsteg	Kemper	Hefeweizen


Biertrinker
Kneipe
Gast
Bier
Kowalski
Kemper
Pils
Kowalski
Eickler
Hefeweizen
Innsteg
Kemper
Hefeweizen
Besucht
Kneipe
Gast
Kowalski
Kemper
Kowalski
Eickler
Innsteg
Kemper
π....
⋈
Trinkt
Gast
Bier
Kemper
Pils
Eickler
Hefeweizen
Kemper
Hefeweizen
≠
Besucht A Trinkt
Kneipe
Gast
Bier
Kowalski
Kemper
Pils
Kowalski
Kemper
Hefeweizen
Kowalski
Eickler
Hefeweizen
Innsteg
Kemper
Pils
Innsteg
Kemper
Hefeweizen

Erläuterung des Biertrinker-Beispiels

Unser Biertrinker-Beispiel war eine „verlustige“ Zerlegung und dementsprechend war die hinreichende Bedingung verletzt. Es gilt nämlich nur die eine nicht-triviale funktionale Abhängigkeit

- {Kneipe, Gast} ⟶ {Bier}

Wohingegen keine der zwei möglichen, die Verlustlosigkeit garantierenden FDs gelten

- {Gast} ⟶ {Bier}
- {Gast} ⟶ {Kneipe}

Das liegt daran, dass die Leute (insbes. Kemper) in unterschiedlichen Kneipen unterschiedliches Bier trinken, in derselben Kneipe aber immer das gleiche Bier (damit sich die KellnerInnen darauf einstellen können?)

Verlustfreie Zerlegung

Eltern		
Vater	*Mutter*	*Kind*
Johann	Martha	Else
Johann	Maria	Theo
Heinz	Martha	Cleo

$\pi_{\text{Vater, Kind}}$

$\pi_{\text{Mutter, Kind}}$

Väter	
Vater	*Kind*
Johann	Else
Johann	Theo
Heinz	Cleo

Mütter	
Mutter	*Kind*
Martha	Else
Maria	Theo
Martha	Cleo

Erläuterung der verlustfreien Zerlegung der Eltern-Relation

Eltern: {[Vater, Mutter, Kind]}
Väter: {[Vater, Kind]}
Mütter: {[Mutter, Kind]}

Verlustlosigkeit ist garantiert
Es gilt nicht nur eine der hinreichenden FDs, sondern gleich beide

- {Kind}⟶{Mutter}
- {Kind}⟶{Vater}

Also ist {Kind} natürlich auch der Schlüssel der Relation Eltern

Die Zerlegung von Eltern ist zwar verlustlos, aber auch ziemlich unnötig, da die Relation in sehr gutem Zustand ist (s. Normalform)

Kriterien für die Verlustlosigkeit einer Zerlegung

$\mathcal{R} = \mathcal{R}1 \cup \mathcal{R}2$

- $R1 := \Pi_{\mathcal{R}1}(R)$
- $R2 := \Pi_{\mathcal{R}2}(R)$

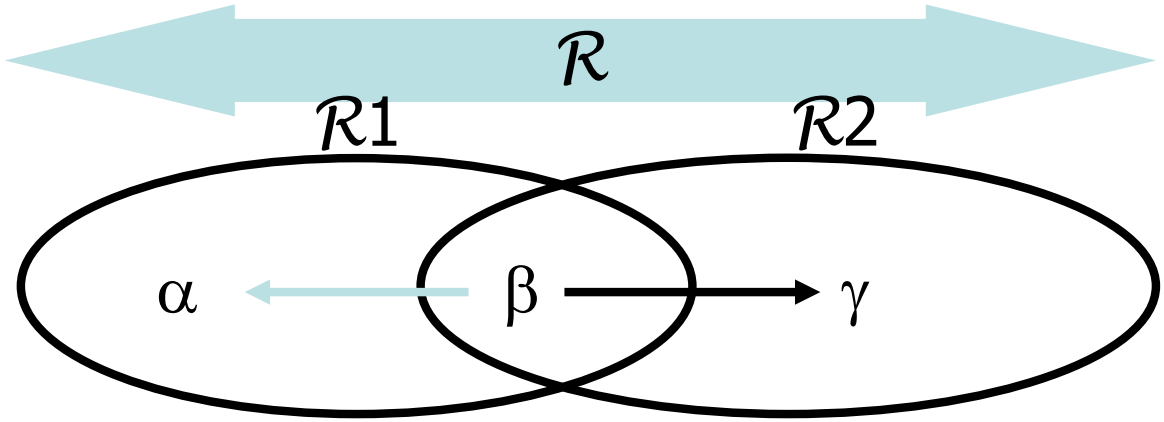
Eine Zerlegung von $\mathcal{R}$ in $\mathcal{R}1$ und $\mathcal{R}2$ ist verlustlos,
falls für jede mögliche (gültige) Ausprägung R von $\mathcal{R}$ gilt:

- $R = R1 \bowtie R2$

Hinreichende Bedingung für die Verlustlosigkeit einer Zerlegung:
Es muss eine FD der Form

- $(\mathcal{R}1 \cap \mathcal{R}2) \longrightarrow \mathcal{R}1$ oder
- $(\mathcal{R}1 \cap \mathcal{R}2) \longrightarrow \mathcal{R}2$

gelten



Abhängigkeitserhaltung

$\mathcal{R}$ ist zerlegt in $\mathcal{R}_1$, ..., $\mathcal{R}_n$

$F_{\mathcal{R}} \equiv (F_{\mathcal{R}1} \cup \ldots \cup F_{\mathcal{R}n})$ bzw. $F_{\mathcal{R}}^{+} = (F_{\mathcal{R}1} \cup \ldots \cup F_{\mathcal{R}n})^{+}$

Beispiel für Abhängigkeitsverlust

- Geg. Schema PLZverzeichnis: {[Straße, Ort, Bland, PLZ]}

Zugeordnete FDs

1. {PLZ} ⟶ {Ort, BLand}
2. {Straße, Ort, BLand} ⟶ {PLZ}

Betrachte die Zerlegung

- Straßen: {[PLZ, Straße]}
- Orte: {[PLZ, Ort, BLand]}

FD 2 kann weder direkt über Straßen noch über Orte geprüft werden

Zerlegung der Relation PLZverzeichnis

PLZverzeichnis			
Ort	***BLand***	***Straße***	*PLZ*
Frankfurt	Hessen	Goethestraße	60313
Frankfurt	Hessen	Galgenstraße	60437
Frankfurt	Brandenburg	Goethestraße	15234

$\pi_{PLZ,Straße}$

$\pi_{Ort,Bland,PLZ}$

Straßen	
PLZ	***Straße***
15234	Goethestraße
60313	Goethestraße
60437	Galgenstraße

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

Die FD {Straße, Ort, BLand} ⟶ {PLZ} ist im zerlegten Schema nicht mehr enthalten ⟶ Einfügen inkonsistenter Tupel möglich

Einfügen zweier Tupel, die die FD Ort,Bland,Straße→PLZ verletzen

PLZverzeichnis			
Ort	***BLand***	***Straße***	*PLZ*
Frankfurt	Hessen	Goethestraße	60313
Frankfurt	Hessen	Galgenstraße	60437
Frankfurt	Brandenburg	Goethestraße	15234

$\pi_{\text{PLZ,Straße}}$

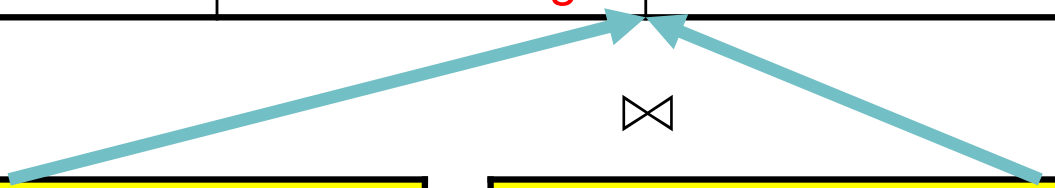
$\pi_{\text{Stadt,Bland,PLZ}}$

Straßen	
PLZ	***Straße***
15234	Goethestraße
60313	Goethestraße
60437	Galgenstraße
15235	Goethestraße

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

Einfügen zweier Tupel, die die FD Ort,Bland,Straße → PLZ verletzen

PLZverzeichnis			
Ort	***BLand***	***Straße***	*PLZ*
Frankfurt	Hessen	Goethestraße	60313
Frankfurt	Hessen	Galgenstraße	60437
Frankfurt	Brandenburg	Goethestraße	15234
Frankfurt	Brandenburg	Goethestraße	15235



⋈

Straßen	
PLZ	***Straße***
15234	Goethestraße
60313	Goethestraße
60437	Galgenstraße
15235	Goethestraße

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

Gütekriterien für Relationenschemata

- Redundanzfreiheit in den Daten
- Prüfung der einem Relationenschema zugeordneten FDs möglichst nur durch Schlüsselbedingung (und nicht durch aufwendige Berechnung von Joins)

→ Normalformen

Erste Normalform: nur „einfache“ Domänen

Beispiel:

Eltern		
Vater	*Mutter*	*Kinder*
Johann	Martha	{Else, Lucie}
Johann	Maria	{Theo, Josef}
Heinz	Martha	{Cleo}

1 NF

Eltern		
Vater	*Mutter*	*Kind*
Johann	Martha	Else
Johann	Martha	Lucie
Johann	Maria	Theo
Johann	Maria	Josef
Heinz	Martha	Cleo

Exkurs: NF^2-Relationen

Non-First Normal-Form-Relationen

Geschachtelte Relationen

Nachteil: Anfragesprache erheblich komplizierter

(zusätzlich Schachtelungs- und Entschachtelungsoperator nötig)

Eltern			
Vater	*Mutter*	*Kinder*	
		KName	*KAlter*
Johann	Martha	Else	5
		Lucie	3
Johann	Maria	Theo	3
		Josef	1
Heinz	Martha	Cleo	9

Zweite Normalform

Eine Relation $\mathcal{R}$ mit zugehörigen FDs $F_{\mathcal{R}}$ ist in zweiter Normalform, falls jedes Nichtschlüssel-Attribut $A \in \mathcal{R}$ voll funktional abhängig ist von jedem Kandidatenschlüssel der Relation.

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
...	...	...	...

Studentenbelegung mit Schlüssel {MatrNr, VorlNr} ist nicht in zweiter NF

- {MatrNr} ⟶ {Name}
- {MatrNr} ⟶ {Semester}

Vermeidung von Redundanz in den Daten

Beispiel:

$\mathcal{R}$ = {[A, B, C ,D]}, F = {A ⟶ B, D ⟶ ABCD}, Schlüsselkandidat: {D}

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

„Do not represent the same fact twice!“

Allgemeiner Fall: $\alpha \longrightarrow \beta \in F$, dann: α Superschlüssel oder FD ist trivial

ggf. Dekomposition notwendig (verlustfrei und abhängigkeitsbewahrend)

Boyce-Codd-Normalform

Die Boyce-Codd-Normalform (BCNF) stellt nochmals eine Verschärfung der zweiten Normalform dar.

Ein Relationenschema $\mathcal{R}$ mit FDs F ist in BCNF, wenn

für jede für $\mathcal{R}$ geltende funktionale Abhängigkeit der Form $\alpha \longrightarrow \beta \in F$ mindestens eine der folgenden zwei Bedingungen gilt:

- $\beta \subseteq \alpha$, d.h., die Abhängigkeit ist trivial oder
- α ist Superschlüssel von $\mathcal{R}$

NB: Aus BCNF folgt 2NF

Beispiel:

- Gegeben ein relationales Schema mit zugeordneten FDs:

Städte: {[Ort, BLand, Ministerpräsident/in, EW]}

Geltende FDs:

{BLand} ⟶ {Ministerpräsident/in}
{Ort, BLand} ⟶ {EW}
{Ministerpräsident/in} ⟶ {BLand}

- 2. NF?

Beispiel:

- Gegeben ein relationales Schema mit zugeordneten FDs:

Städte: {[Ort, BLand, Ministerpräsident/in, EW]}

Geltende FDs:

{BLand} ⟶ {Ministerpräsident/in}
{Ort, BLand} ⟶ {EW}
{Ministerpräsident/in} ⟶ {BLand}

Kandidatenschlüssel: {{Ministerpräsident/in, Ort}
{BLand, Ort}}

- 2. NF? Ja
- BCNF?

Beispiel:

- Gegeben ein relationales Schema mit zugeordneten FDs:

Städte: {[Ort, BLand, Ministerpräsident/in, EW]}

Geltende FDs:

{BLand} ⟶ {Ministerpräsident/in}
{Ort, BLand} ⟶ {EW}
{Ministerpräsident/in} ⟶ {BLand}

Kandidatenschlüssel: {{Ministerpräsident/in, Ort}
{BLand, Ort}}

- 2. NF? Ja
- BCNF? Nein, daher Zerlegung nötig!

Beispiel: Dekomposition der Relation Städte

Städte: {[Ort, BLand, Ministerpräsident/in, EW]}

Geltende FDs:

- {BLand} ⟶ {Ministerpräsident/in}
- {Ort, BLand} ⟶ {EW}
- {Ministerpräsident/in} ⟶ {BLand}

$\mathcal{R}$i1:

- **Regierungen:** {[BLand, Ministerpräsident/in]}
- **FDs:** { {BLand} ⟶ {Ministerpräsident/in}, {Ministerpräsident/in} ⟶ {BLand} }
- **Kandidatenschlüssel:** κ = { {BLand}, {Ministerpräsident/in} }

$\mathcal{R}$i2:

- **Städte:** {[Ort, BLand, EW]}
- **FDs:** { {Ort, BLand} ⟶ {EW} }
- **Kandidatenschlüssel:** κ = { {Ort, BLand} }

Dekompositions-Algorithmus

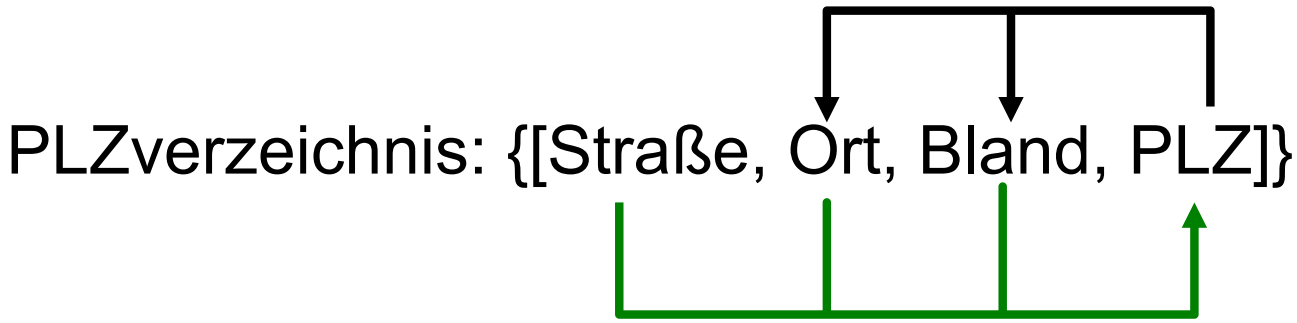
Starte mit $Z = \{\mathcal{R}\}$

Solange es noch ein Relationenschema $\mathcal{R}_i$ in Z gibt, das nicht in BCNF ist, mache Folgendes:

- Es gibt also eine für $\mathcal{R}_i$ **geltende** nicht-triviale funktionale Abhängigkeit $\alpha \longrightarrow \beta$ mit
  - $\alpha \cap \beta = \varnothing$ und
  - $\alpha \longrightarrow \mathcal{R}_i$ gilt nicht
- **Finde** eine solche FD
  - Man sollte sie 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 $\mathcal{R}_{i2} := \mathcal{R}_i \setminus \beta$
- **Entferne** $\mathcal{R}_i$ aus Z und füge $\mathcal{R}_{i1}$ und $\mathcal{R}_{i2}$ ein, also

  $Z := (Z \setminus \{\mathcal{R}_i\}) \cup \{\mathcal{R}_{i1}\} \cup \{\mathcal{R}_{i2}\}$

Beispiel



Funktionale Abhängigkeiten:

- {PLZ} ⟶ {Ort, BLand}
- {Straße, Ort, BLand} ⟶ {PLZ}

Betrachte die Zerlegung

- Straßen: {[PLZ, Straße]}
- Orte: {[PLZ, Ort, BLand]}

Diese Zerlegung

- ist verlustlos, aber
- nicht abhängigkeitserhaltend (die zweite FD kann keiner Subrelation zugeordnet werden)

Prüfung der zweiten FD wäre
bei Zerlegung nur über Join möglich

Boyce-Codd-Normalform

Man kann jede Relation verlustlos in BCNF-Relationen zerlegen

Manchmal lässt sich dabei die Abhängigkeitserhaltung aber nicht erzielen

Warum ist das ein Problem?

- Prüfung von {Straße, Ort, BLand} ⟶ {PLZ} muss explizit erfolgen
- Hierzu müsste die Relation Straßen ⋈ Orte berechnet werden
- Prüfung muss bei jeder Änderung von Straßen oder Orte erfolgen
- Extrem aufwendig

Was, wenn Boyce-Codd-Normalform nicht möglich?

Beispiel:

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

$F_{\mathcal{R}}$ = {A ⟶ D, CD ⟶ AB}

Schlüsselkandidaten: {{A,C}, {C,D}}

- Codd 71: Schema „ganz gut“, wenn es keine „transitiven Abhängigkeiten“ gibt
- FD A ⟶ D zugeordnet zu $\mathcal{R}$ wäre vielleicht tolerierbar (keine „Transitivität“)
  - Zwar Redundanz vorhanden aber unvermeidbar
  - Zusätzliche Prüfung A ⟶ D (neben Eindeutigkeitstest von AC und CD) nötig, aber lokal mögl. (kein Join erforderlich)

Dritte Normalform (formuliert nach Zaniolo 82)

Ein Relationenschema $\mathcal{R}$ ist in dritter Normalform, wenn für **jede für $\mathcal{R}$ geltende** funktionale Abhängigkeit der Form
$\alpha \longrightarrow B$ mit $\alpha \subseteq \mathcal{R}$ und $B \in \mathcal{R}$
mindestens eine von drei Bedingungen gilt:

- $B \in \alpha$, d.h., die FD ist trivial
- α ist Superschlüssel von $\mathcal{R}$
- Das Attribut B ist in einem Kandidatenschlüssel von $\mathcal{R}$ enthalten (B ist prim)

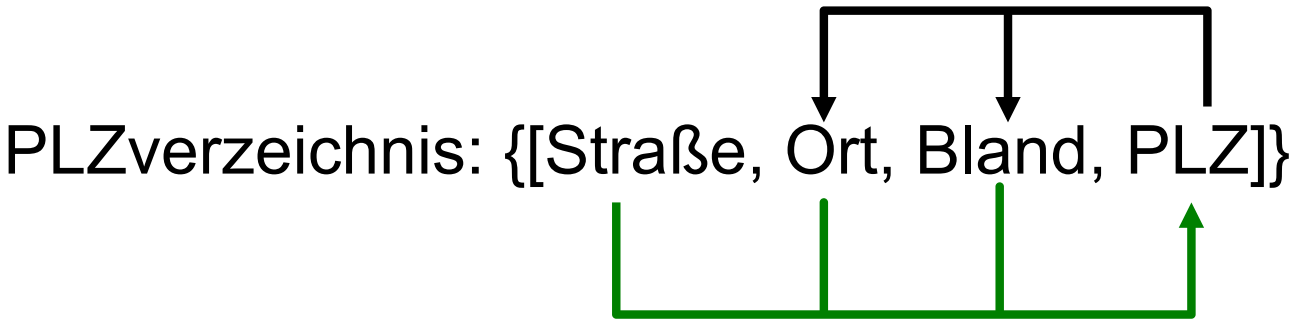
Man beachte: Es wird **jede für $\mathcal{R}$ geltende** FD betrachtet → FD-Hülle!
Warum ist Zaniolos Formulierung interessant?

Frage

- Können wir relationale Schemata finden, so dass alle FDs einem Schema zugeordnet werden können (Abhängigkeitserhaltung) und wenigstens die dritte Normalform gegeben und eine **lokale Prüfung** der FDs möglich ist

- Ja

Redundanz unvermeidbar, Zusätzliche Prüfung



Funktionale Abhängigkeiten:

- {PLZ} ⟶ {Ort, BLand}
- {Straße, Ort, BLand} ⟶ {PLZ}

Kandidatenschlüssel

- { {Straße, Ort, BLand}, {Straße, PLZ} }

3. Normalform gegeben:

- Ort, Bland und PLZ sind in einem Kandidatenschlüssel enthalten

```
create table PLZverzeichnis
 (Straße ...,
  Ort ...,
  BLand ...,
  PLZ ...,
  primary key (Straße, PLZ),
  unique (Straße, Ort, BLand),
  check(all x, all y:
          x.PLZ <> y.PLZ or
            (x.Ort = y.Ort and
             x.BLand = y.BLand)))
```

Zusätzlicher Test nötig

Synthese von Relationenschemata

ProfessorenAdr: {[PersNr, Name, Rang, Raum, Ort, Straße, PLZ, Vorwahl, BLand, EW, Landesregierung]}

F_c:

- F1 {PersNr} ⟶ {Name, Rang, Raum, Ort, Straße, BLand}
- F2 {Raum} ⟶ {PersNr}
- F3 {Straße, BLand, Ort} ⟶ {PLZ}
- F4 {Ort,BLand} ⟶ {EW, Vorwahl}
- F5 {BLand} ⟶ {Landesregierung}
- F6 {PLZ} ⟶ {BLand, Ort}

Professoren: {[PersNr, Name, Rang, Raum, Ort, Straße, BLand]}
zugeordnete **FDs** = {F1, F2}, κ = {{PersNr}, {Raum}}

PLZverzeichnis: {[Straße, BLand, Ort, PLZ]}
zug. **FDs** = {F3, F6}, κ = {{Straße, BLand, Ort}, {Straße, PLZ}}

OrteVerzeichnis: {[Ort, BLand, EW, Vorwahl]}
zugeordnete **FDs** = {F4}, κ = {{Ort,BLand}}

Regierungen: {[Bland, Landesregierung]}
zugeordnete **FDs** = {F5}, κ = {{BLand}}

Synthesealgorithmus

Wir geben jetzt einen sogenannten Synthesealgorithmus an, mit dem zu einem gegebenen Relationenschema $\mathcal{R}$ mit funktionalen Anhängigkeiten F eine Darstellung in $\mathcal{R}_1$, ..., $\mathcal{R}_n$ ermittelt wird, die alle drei folgenden Kriterien erfüllt.

- $\mathcal{R}_1$, ..., $\mathcal{R}_n$ ist eine verlustlose Relationendarstellung von $\mathcal{R}$.
- Die Relationendarstellung $\mathcal{R}_1$, ..., $\mathcal{R}_n$ ist abhängigkeitserhaltend.
- Alle $\mathcal{R}_1$, ..., $\mathcal{R}_n$ sind in dritter Normalform.

Synthesealgorithmus

1. Bestimme die **kanonische Überdeckung** F_c zu F. Wiederholung:
   - Linksreduktion
   - Rechtsreduktion
   - Entfernung von FDs der Form $\alpha \rightarrow \varnothing$
   - 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 $\mathcal{R}\alpha$ die FDs $F\alpha := \{\alpha` \rightarrow \beta` \in F_c \mid \alpha` \cup \beta` \subseteq \mathcal{R}\alpha\}$ zu.
3. Falls eines der in Schritt 2. erzeugten Schemata einen Kandidatenschlüssel von $\mathcal{R}$ bzgl. F_c enthält, sind wir fertig. Sonst wähle einen Kandidatenschlüssel $\kappa \subseteq \mathcal{R}$ aus und definiere folgendes Schema:
   - $\mathcal{R}\kappa := \kappa$
   - $F\kappa := \varnothing$
4. Eliminiere diejenigen Schemata $\mathcal{R}\alpha$, die in einem anderen Relationenschema $\mathcal{R}\alpha`$ enthalten sind, d.h.,
   - $\mathcal{R}\alpha \subseteq \mathcal{R}\alpha`$

Synthesealgorithmus erzeugt Rel. in 3. NF

Siehe David Maier, The Theory of Relational Databases, Computer Science Press, 1983

http://web.cecs.pdx.edu/~maier/TheoryBook/TRD.html

Argumente in Kurzform:

- Jedes erzeugte $\mathcal{R}_\alpha$ wird aus FD $\alpha \longrightarrow \beta \in F_c$ erzeugt.
- Nehmen wir an, es gäbe $\gamma \longrightarrow B \in F_c^+$ von $\mathcal{R}_\alpha$, so dass B nicht prim und γ kein Schlüssel ist.
- Es muss gelten $B \in \beta$. Da aber $\gamma \longrightarrow B$ und $\gamma \subseteq \alpha \cup \beta\backslash\{B\}$ gilt, wäre B überflüssig in β und damit $\alpha \longrightarrow \beta \notin F_c$

Weitere Einschränkungen im Datenmodell

- Spezialisierung / Generalisierung von Entitätstypen (ISA)
  - Fremdschlüssel (in SQL)
- Enthaltensein-Einschränkungen von Relationen (inclusion dependencies)
  - Beispiel: hat-kind vs. hat-nachfahre
  - Multidimensionale Fremdschlüssel (in SQL)
- Mehrwertige Abhängigkeiten (multiple value dependencies)
  - (Weitere) Zerlegung
- Tupel-generierende Abhängigkeiten (tuple generating dependencies)
  - Wichtig für Datenaustausch und –integration
  - Kommt später

Mehrwertige Abhängigkeiten: ein Beispiel

Notation mehrwertige Abhängigkeiten dieser Relation:

- {PersNr} $\twoheadrightarrow$ {Sprache} und
- {PersNr} $\twoheadrightarrow$ {ProgSprache}

MVDs führen zu Redundanz und Anomalien

Fähigkeiten		
PersNr	Sprache	ProgSprache
3002	griechisch	C
3002	lateinisch	Pascal
3002	griechisch	Pascal
3002	lateinisch	C
3005	deutsch	Ada

$\alpha \twoheadrightarrow \beta$ gilt genau dann wenn

- es zu zwei Tupel t1 und t2 mit gleichen α–Werten
- auch zwei Tupel t3 und t4 gibt mit
  - t3.α = t4.α = t1.α = t2.α
  - t3.β = t2.β , t4.β = t1.β
  - t4.γ = t2.γ , t3.γ = t1.γ
    wobei $\gamma = \mathcal{R} \setminus (\alpha \cup \beta)$

"Zu zwei Tupeln mit gleichem α -Wert kann man die β -Werte vertauschen, und die Tupel müssen bei gleichem γ auch in der Relation sein"

Tuple-generating dependencies

- Man kann eine Relation MVD-konform machen, indem man zusätzliche Tupel einfügt
- Bei FDs geht das nicht!!

Mehrwertige Abhängigkeiten: ein Beispiel

Fähigkeiten		
PersNr	Sprache	ProgSprache
3002	griechisch	C
3002	lateinisch	Pascal
3002	griechisch	Pascal
3002	lateinisch	C
3005	deutsch	Ada

$\pi_{\text{PersNr, Sprache}}$

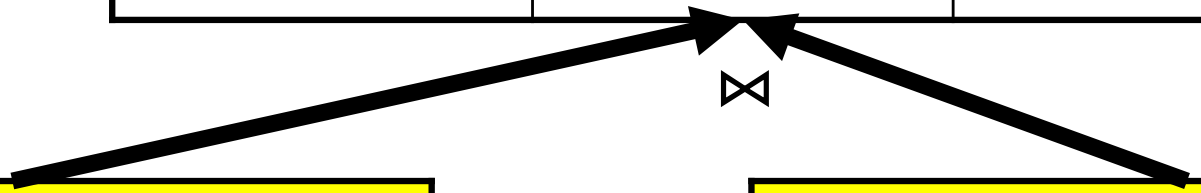
$\pi_{\text{PersNr, ProgSprache}}$

Sprachen	
PersNr	Sprache
3002	griechsich
3002	lateinisch
3005	deutsch

ProgSprachen	
PersNr	ProgSprache
3002	C
3002	Pascal
3005	Ada

Mehrwertige Abhängigkeiten: ein Beispiel

Fähigkeiten		
PersNr	Sprache	ProgSprache
3002	griechisch	C
3002	lateinisch	Pascal
3002	griechisch	Pascal
3002	lateinisch	C
3005	deutsch	Ada



Sprachen	
PersNr	Sprache
3002	griechisch
3002	lateinisch
3005	deutsch

ProgSprachen	
PersNr	ProgSprache
3002	C
3002	Pascal
3005	Ada

Vierte Normalform

Eine MVD $\alpha \twoheadrightarrow \beta$ ist trivial genau dann wenn

- $\beta \subseteq \alpha$ oder
- $\beta = R \setminus \{\alpha\}$

Eine Relation R ist in 4 NF wenn für jede MVD $\alpha \twoheadrightarrow \beta$
eine der folgenden Bedingungen gilt:

- Die MVD ist trivial **oder**
- α ist Superschlüssel von R

Zerlegung einer Relation in 3. Normalform, so dass
nur triviale MVDs vorkommen verlustlos möglich

Zusammenfassung

Die Verlustlosigkeit ist für alle Zerlegungsalgorithmen in alle Normalformen garantiert

Die Abhängigkeitserhaltung kann nur bis zur dritten Normalform garantiert werden