6. Relationale Entwurfstheorie

1. Funktionale Abhängigkeiten
2. Armstrong-Kalkül
3. Zerlegung von Relationen
4. Normalformen und Normalisierungen

Einführung

Bis jetzt:

- Nutzen- und Anforderungsanalys (Pflichtenheft)
- Entity-Relationship-Entwurf
- relationales Schema

Zu tun:

- Feintuning des erstellten Schemas auf der Basis von intrarelationalen Abhängigkeiten
  - Funktionale Abhängigkeiten
  - Kriterien für gute Schemata, schlechte Schemata
  - Normalformen
  - Algorithmen zur Normalisierung

Ein „schlechtes“ Schema…(Warum?)

ProfVorl						
PersNr	Name	Rang	Raum	VorlNr	Titel	SWS
2125	Sokrates	W3	226	5041	Ethik	4
2125	Sokrates	W3	226	5049	Mäeutik	2
2125	Sokrates	W3	226	4052	Logik	4
2132	Popper	W2	52	5259	Der Wiener Kreis	2
2137	Kant	W3	7	4630	Die 3 Kritiken	4

6. Relationale Entwurfstheorie

1. **Funktionale Abhängigkeiten**
2. Armstrong-Kalkül
3. Zerlegung von Relationen
4. Normalformen und Normalisierungen

Funktionale Abhängigkeiten

- Das zentrale Konzept der relationalen Entwurfstheorie

- Sei die Attributmenge eines Relationenschemas . Die funktionalen Abhängigkeiten über bilden eine zweistellige Relation „“ auf den Attributmengen aus :

(gesprochen: von Alpha nach Beta)

- In Worten: ist funktional abhängig von
  oder die -Werte bestimmen die -Werte funktional (d.h. eindeutig)

- Für zwei Attributmengen und eine Relation sagen wir *erfüllt* die funktionale Abhängigkeit , wenn gilt:

Funktionale Abhängigkeiten

- Verallgemeinerung der Schlüsseleigenschaft
  - Eindeutigkeitseigenschaft der Schlüssel als funktionale Abhängigkeit:

- Eine funktionale Abhängigkeit lässt sich ebenfalls als intrarelationale Abhängigkeit auffassen:

- Ist , so heißt eine *triviale Abhängigkeit*

- Funktionale Abhängigkeiten werden auch als FDs (functional dependencies) abgekürzt

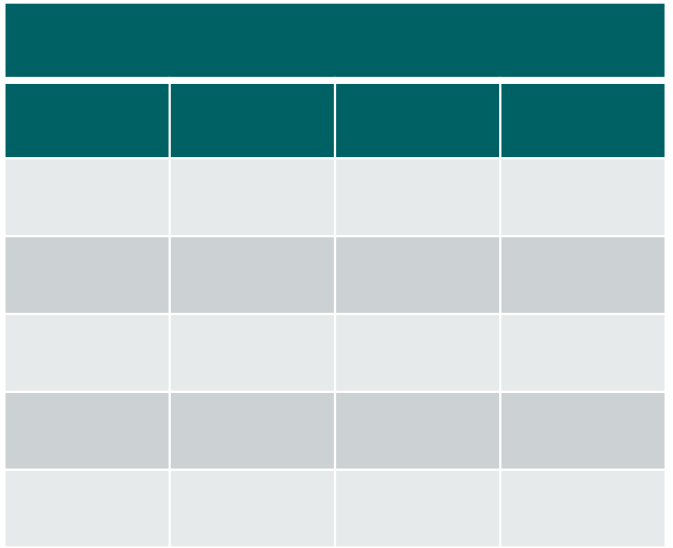
Funktionale Abhängigkeiten – Beispiel

- Betrachte Schema mit Attributmenge und FD .

  Die Ausprägung erfüllt diese FD:

  nur für die Tupel gilt und für diese gilt ebenfalls

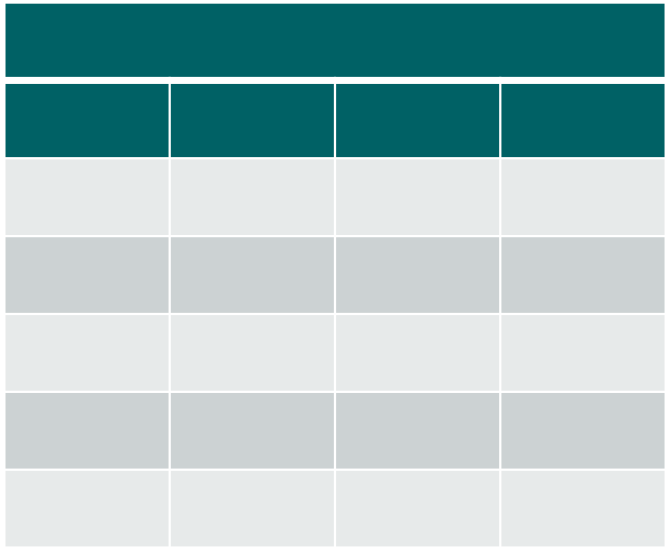
  – diese Ausprägung erfüllt auch die FDs



nicht aber die FDs

Funktionale Abhängigkeiten – Beispiel

- Wichtig:
  - funktionale Abhängigkeiten beschreiben die Menge aller gültigen Relationen (wenn nichts anderes gesagt wird)
  - üblicherweise wird gefragt: welche zusätzlichen FDs lassen sich aus den gegebenen FDs ableiten
  - es wird nicht gefragt: welche zusätzlichen FDs erfüllt diese konkrete Ausprägung



- Übrigens: die Notation

durch wird häufig auch abgekürzt, z.B.

Funktionale Abhängigkeiten – Beispiel

1234	Michael	6	eingeschrieben
5678	Andrea	4	eingeschrieben
4711	Sabine	8	beurlaubt
815	Franz	12	exmatrikuliert

	Wert in der Ausprägung	Wert in allen Ausprägungen

Überprüfen funktionaler Abhängigkeiten

- Ein einfacher Algorithmus zur Überprüfung einer FD:
  - Eingabe: eine Relation und eine FD
  - Ausgabe: *ja* genau dann, wenn in erfüllt ist
  - Algorithmus:
    - sortiere nach den -Werten
    - falls alle Gruppen, bestehend aus Tupeln mit gleichen -Werten, auch gleiche -Werte aufweisen: *ja*; sonst: *nein*

- Die Laufzeit dieses Algorithmus wird durch die Sortierung dominiert
  - Komplexität

Schlüssel

- Präzisierung des Schlüsselbegriffs
  - Dazu sei ein Relationenschema mit Attributmenge und funktionalen Abhängigkeiten
- Superschlüssel (Eindeutigkeit) :
  - heißt *Superschlüssel*, falls gilt
  - bestimmt also alle anderen Attributwerte
  - selbst ist stets auch ein Superschlüssel, da trivialerweise gilt
- voll funktional abhängig (Minimalität):
  - heißt *voll funktional abhängig* von , falls
    - gilt
    - für alle gilt, d.h. kann nicht „verkleinert“ werden

Schlüssel

- Präzisierung des Schlüsselbegriffs
  - Dazu sei ein Relationenschema mit Attributmenge und funktionalen Abhängigkeiten
- Schlüsselkandidat:
  - Eine Attributmenge heißt *Schlüsselkandidat*, falls voll funktional abhängig von ist
- Primärschlüssel:
  - In einem Relationenschema wird einer der Schlüsselkandidaten als *Primärschlüssel* ausgewählt
  - Fremdschlüssel sollten z.B. immer nur auf den Primärschlüssel verweisen

Schlüssel – Beispiel

Orte			
Name	BLand	Vorwahl	EW
Frankfurt	Hessen	069	690.000
Frankfurt	Brandenburg	0335	60.000
München	Bayern	089	1.378.000.000
Passau	Bayern	0851	50.000

- Annahme: Ortsnamen sind innerhalb eines Bundeslandes eindeutig
- Schlüsselkandidaten:

  beide sind minimal:
  - Städte in unterschiedlichen Bundesländern können denselben Namen besitzen
  - kleine Dörfer können sich dieselbe Vorwahl teilen

6. Relationale Entwurfstheorie

1. Funktionale Abhängigkeiten
2. **Armstrong-Kalkül**
3. Zerlegung von Relationen
4. Normalformen und Normalisierungen

Bestimmung aller funktionalen Abhängigkeiten

- Frage: Ausgehend von einer Menge funktionaler Abhängigkeiten (beim Datenbank-Entwurf erstellt), welche zusätzlichen funktionalen Abhängigkeiten sind implizit immer erfüllt?

- *Beispiel*:
  - Erweiterung von Universitäts-Beispiel um Adressen
  - erster Entwurf für Professoren und Adressen:

Bestimmung aller funktionalen Abhängigkeiten – Beispiel

- Funktionale Abhängigkeiten :

- implizierte funktionale Abhängigkeiten:

Implizierte funktionale Abhängigkeiten (Formalisierung)

- Seien eine Attributmenge und eine Menge von funktionalen Abhängigkeiten über .
- Semantische Grundlagen:
  - Eine Relationsausprägung *erfüllt* , falls jede funktionale Abhängigkeit erfüllt
    (man sagt auch: ist ein Modell von)
  - Schreibweise:
  - Die Menge aller *gültigen Ausprägungen* des Schemas ist
  - Zwei Mengen von funktionalen Abhängigkeiten über heißen *äquivalent*, falls sie die gleichen gültigen Relationen definieren:

Implizierte funktionale Abhängigkeiten (Definition)

- Seien eine Attributmenge und eine Menge von funktionalen Abhängigkeiten über .
- *Implikation*:
  - Wir sagen, dass die Menge funktionaler Abhängigkeiten die funktionale Abhängigkeit impliziert, falls alle erfüllenden Relationenausprägungen auch erfüllen.
  - Schreibweise:

ist eine semantische Beziehung

- nicht praktikabel: Überprüfe jede gültige Ausprägung , ob gilt
- stattdessen: Armstrong-Kalkül

Armstrong-Kalkül

- Algorithmische Bestimmung aller implizierten funktionalen Abhängigkeiten

- Hilfsmittel: Armstrong-Axiome
  Seien Attributmengen.

  - Reflexivität : Ist eine Teilmenge von , so gilt auch .
  - Verstärkung : Falls gilt, so gilt auch . Wobei hier .
  - Transitivität : Falls und gilt, so gilt auch .

Armstrong-Kalkül

- Algorithmische Bestimmung aller implizierten funktionalen Abhängigkeiten

- *Ableitbar* :
  - Wir sagen, dass die funktionale Abhängigkeit aus der Menge der funktionalen Abhängigkeiten *ableitbar* ist, falls:

    Es gibt eine endliche Folge , sodass für jedes gilt:

    erhält man aus durch Anwendung der Axiome oder

  - Schreibweise

ist eine syntaktische Beziehung

Armstrong-Kalkül – Beispiel

- *Ableitbar* :
- Funktionale Abhängigkeiten :
- Mithilfe der Transitivitätsregel : Falls und gilt, so gilt auch erhält man aus

  die funktionale Abhängigkeit .

Armstrong-Kalkül – Korrektheit und Vollständigkeit

- Liefert das Armstrong-Kalkül alle „gültigen“ bzw. von implizierten FDs?

  Sei eine Attributmenge und eine Menge von FDs über

  - Zu nennen wir die *(geschlossene) Hülle* von
  - Gilt also

- Ja, der Armstrong-Kalkül ist *korrekt* und *vollständig*.
  - *korrekt*: Für jede funktionale Abhängigkeit mit gilt auch

    (es lassen sich nur „gültige“ funktionale Abhängigkeiten ableiten, „“)
  - *vollständig*: Jede von implizierte funktionale Abhängigkeit (also) lässt sich mithilfe des Armstrong-Kalküls ableiten, d.h. („“)

  (Beweis der Korrektheit jetzt, Beweis der Vollständigkeit später)

Armstrong-Kalkül – Korrektheit

- Korrektheit der Reflexivitätsregel : Für gilt .
  - Seien eine gültige Relationsausprägung, und . Außerdem seien zwei beliebige Tupel mit . Dann gilt auch .

    Insgesamt: gilt auch in .

Armstrong-Kalkül – Korrektheit

- Korrektheit der Verstärkungsregel : Für gilt auch .
  - Seien eine gültige Relationsausprägung, und . Außerdem seien zwei beliebige Tupel mit . Dann gilt auch und . Daraus folgt .

    Insgesamt: gilt auch in .

Armstrong-Kalkül – Korrektheit

- Korrektheit der Transitivitätsregel : Für gilt auch
  - Seien eine gültige Relationenausprägung, und . Außerdem seien zwei beliebige Tupel mit . Wegen , gilt also auch .

    Wegen , gilt dann auch .

    Insgesamt: gilt auch in .

Armstrong-Kalkül – Erweiterung

Es ist für den Herleitungsprozess komfortabel, weitere Regeln hinzuzunehmen

- Erweiterung der Armstrong-Axiome um drei Regeln:

  Seien Attributmengen.

  - Vereinigung : Gelten und , so gilt auch .
  - Dekomposition : Falls gilt, so gelten auch und .
  - Pseudotransitivität : Falls und gilt, so gilt auch .

Armstrong-Kalkül – Erweiterung

- Ableitung der Vereinigungsregel : Gelten $\alpha \rightarrow \beta$ und $\alpha \rightarrow \gamma$, so gilt auch $\alpha \rightarrow \beta\gamma$.
  - Seien und . Über die Grundregeln erhalten wir

Armstrong-Kalkül – Erweiterung

- Ableitung der Dekompositionsregel ():

  Falls $\alpha \rightarrow \beta\gamma$ gilt, so gelten auch $\alpha \rightarrow \beta$ und $\alpha \rightarrow \gamma$.

  - Sei $\alpha \rightarrow \beta\gamma$. Über die Grundregeln erhalten wir

Armstrong-Kalkül – Erweiterung

- Ableitung der Pseudotransitivitätsregel :

  Falls und gilt, so gilt auch

  Sei und . Über die Grundregeln erhalten wir

Armstrong-Kalkül – Vereinigung

- Funktionale Abhängigkeiten :

- Beispiel: ist ebenfalls aus ableitbar
  –
  –
  –

Attributhülle

- Oft ist man nicht an der gesamten Hülle interessiert:
  - Welche Attribute sind unter einer gegebenen Menge von FDs von einer bestimmten Attributmenge funktional bestimmt?
  - Man nennt die Attributhülle von unter
  - Algorithmus zur Bestimmung von

ist genau dann ein Superschlüssel, falls gilt:

Attributhülle – Beispiel

- Attributhülle

- Beispiel:

Durchlaufe :

: x, : , :

Durchlaufe nochmal:

Attributhülle

- Lemma **L1**: Die Attributhülle besitzt folgende Eigenschaft.

  Für jede Teilmenge gilt auch .

- Beweis (Skizze):

  Wir beschränken uns auf den Fall . Da ist, gibt es und
  mit . Mit der Reflexivitätsregel sind auch
  . Mit der Transitivitätsregel und

  sind auch . Mit der Vereinigungsregel erhalten wir

Armstrong-Kalkül – Vollständigkeit

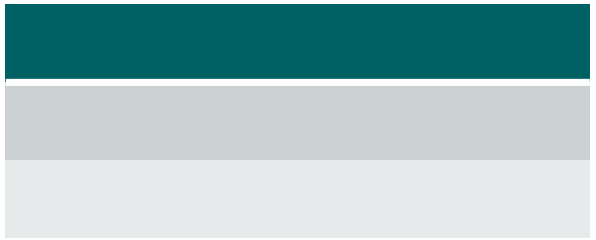
- Das Armstrong-Kalkül ist vollständig, d.h. jede von funktionalen Abhängigkeiten implizierte FD lässt sich mithilfe des Armstrong-Kalküls ableiten. Es gilt also

- Beweis (durch Kontraposition):
  - Wir zeigen die Aussage . D.h. ist eine funktionale Abhängigkeit nicht ableitbar, so wird sie auch nicht von impliziert. Sei dazu eine FD mit und die Menge aller Attribute aus und .
  - Um zu zeigen, dass gilt, konstruieren wir eine Relation , in der gilt, aber nicht.

    <u>Konstruktion:</u>

    Seien und .

    Konstruiere wie rechts.



Armstrong-Kalkül – Vollständigkeit

- Zeige für Vollständigkeit:
- Hilfslemma 1:

  Es gilt , d.h. alle FDs in werden von erfüllt.
- Beweis (durch Widerspruch):

  Angenommen es existiert mit . Dann muss für die beiden Tupel gelten, dass und ist. Nach Konstruktion von kann nur gelten, wenn ist. Ebenso impliziert , dass ist.

  Nach dem vorigen Lemma **L1** folgt aus , dass ist. Mit der Transivititätsregel erhalten wir aus , dass auch ist.

  Dies liefert einen Widerspruch zu .

Armstrong-Kalkül – Vollständigkeit

- Zeige für Vollständigkeit:
- Hilfslemma 2:

  Es gilt , d.h. wird von nicht erfüllt.
- Beweis:

  Da nicht aus ableitbar ist gilt insbesondere . Also existiert ein

  mit . Direkt aus der Konstruktion von folgt dann, dass ist, aber

  Insgesamt erhalten wir, dass von nicht erfüllt wird.
- Es gilt also , aber . Darauf folgt .

Kanonische Überdeckung – Motivation

- Um für zwei Mengen und zu entscheiden, ob sie äquivalent sind (), reicht es zu überprüfen.

Warum?
Freiwillige Übung

- Im Allgemeinen ist die Hülle einer Menge von FDs sehr groß
- Vor allem bei Datenbankmodifikationen:
  - Überprüfen der Konsistenz anhand von sehr aufwändig (auch viele triviale Abhängigkeiten)
  - minimale Menge von „erzeugenden“ funktionalen Abhängigkeiten wünschenswert

- Statt der Hülle : kanonische Überdeckung

Kanonische Überdeckung (Definition)

- Sei eine Menge von funktionalen Abhängigkeiten über einer Attributmenge . Dann heißt eine *kanonische Überdeckung* von , falls gilt:
  1. , d.h. (äquivalent)
  2. In existieren keine FDs bei denen oder überflüssige Attribute enthalten. D.h.
     - Für alle : (nicht äquivalent)
     - Für alle :
  3. Jede linke Seite einer FD ist einzigartig in (sonst ersetze durch Vereinigungen)

- Beispiel:

  Eine kanonische Überdeckung ist .
  (Algorithmus: nächste Folie)

Kanonische Überdeckung – Algorithmus

- Eingabe: Menge von FDs , Ausgabe: Kanonische Überdeckung
  1. Setze
  2. Führe für jede FD eine Linksreduktion durch:
     - Überprüfe für alle , ob überflüssig ist. D.h. ob gilt:

       Falls ja: ersetze durch .

  3. Führe für jede FD eine Rechtsreduktion durch:
     - Überprüfe für alle , ob überflüssig ist. D.h. ob gilt:

       Falls ja: ersetze durch .
  4. Entferne die im 3. Schritt entstandenen FDs der Form
  5. Fasse über die Vereinigungsregel FDs der Form

Kanonische Überdeckung – Beispiel

- Beispiel:

  1. Linksreduktion:
     - : ist nicht in . Ok
     - : Ok
     - : Überprüfe . Ist ? Ja, denn

       Ersetze durch .

       muss nun in nicht mehr überprüft werden.

- Zwischenergebnis:

Kanonische Überdeckung – Beispiel

- Beispiel:

  1. Linksreduktion:
     - : ist nicht in . Ok
     - : Ok
     - : Überprüfe . Ist ? Ja, denn

       Ersetze durch .

       muss nun in nicht mehr überprüft werden.

- Zwischenergebnis:

Kanonische Überdeckung – Beispiel

- Zwischenergebnis:

2. Rechtsreduktion:
   - : Überprüfe .

     Also ist rechts nicht überflüssig.
   - : Analog: ist rechts nicht überflüssig.
   - : Überprüfe .

     Also ist auf der rechten Seite überflüssig. Ersetze durch .

- Neues Zwischenergebnis:

Kanonische Überdeckung – Beispiel

- Zwischenergebnis:

2. Rechtsreduktion:
   - : Überprüfe .

     Also ist rechts nicht überflüssig.
   - : Analog: ist rechts nicht überflüssig.
   - : Überprüfe .

     Also ist auf der rechten Seite überflüssig. Ersetze durch .

- Neues Zwischenergebnis:

Kanonische Überdeckung – Beispiel

- Zwischenergebnis:

2. Rechtsreduktion:
   - : Überprüfe .

     Also ist rechts nicht überflüssig.
   - : Analog: ist rechts nicht überflüssig.
   - : Überprüfe .

     Also ist auf der rechten Seite überflüssig. Ersetze durch .

- Neues Zwischenergebnis:

Kanonische Überdeckung – Beispiel

- Zwischenergebnis:

  3. Entferne :
     - Entferne die Abhängigkeit

  4. Vereinigen:
     - Hier ist nichts zu tun.

- Ergebnis:

6. Relationale Entwurfstheorie

1. Funktionale Abhängigkeiten
2. Armstrong-Kalkül
3. **Zerlegung von Relationen**
4. Normalformen und Normalisierungen

Zerlegung

- funktionale Abhängigkeiten: mächtiges Werkzeug um Konsistenzbedingungen zu modellieren
- In der Sprache der FDs kann nun ausgedrückt werden, welche Schemata „gut“ und welche „schlecht“ sind (siehe „Normalformen und Normalisierungen“)
- „schlechte“ Schemata können durch Zerlegung/Dekomposition in die Normalformen überführt werden.
- Welche Anomalien können bei einem schlechtem Relationenschema auftreten?

Anomalien

- ein offensichtlich „schlechtes“ Schema

ProfVorl						
PersNr	Name	Rang	Raum	VorlNr	Titel	SWS
2125	Sokrates	W3	226	5041	Ethik	4
2125	Sokrates	W3	226	5049	Mäeutik	2
2125	Sokrates	W3	226	4052	Logik	4
2132	Popper	W2	52	5259	Der Wiener Kreis	2
2137	Kant	W3	7	4630	Die 3 Kritiken	4

Änderungs-Anomalie

ProfVorl						
PersNr	Name	Rang	Raum	VorlNr	Titel	SWS
2125	Sokrates	W3	226	5041	Ethik	4
2125	Sokrates	W3	226	5049	Mäeutik	2
2125	Sokrates	W3	226	4052	Logik	4
2132	Popper	W2	52	5259	Der Wiener Kreis	2
2137	Kant	W3	7	4630	Die 3 Kritiken	4

- Angenommen Sokrates soll von Raum 226 nach Raum 233 umziehen
- Die Information ‘Raum’ existiert in diesem Fall mehrfach
- Lösung: Änderung aller Einträge gleichzeitig
  - hoher Speicherbedarf durch Redundanz
  - erhöhter Zeitbedarf bei Änderungen

Einfüge-Anomalie

ProfVorl						
PersNr	Name	Rang	Raum	VorlNr	Titel	SWS
2125	Sokrates	W3	226	5041	Ethik	4
2125	Sokrates	W3	226	5049	Mäeutik	2
2125	Sokrates	W3	226	4052	Logik	4
2132	Popper	W2	52	5259	Der Wiener Kreis	2
2137	Kant	W3	7	4630	Die 3 Kritiken	4

- Schema kombiniert Informationen verschiedener unpassender Entitytypen
  - Hinzufügen eines Professors ohne Vorlesung

    NULL-Werte in VorlNr, Titel und SWS
  - Analog: Hinzufügen einer Vorlesung zu der noch kein Dozent festgelegt wurde

    NULL-Werte in PersNr, Name, Rang und Raum

Lösch-Anomalie

ProfVorl						
PersNr	Name	Rang	Raum	VorlNr	Titel	SWS
2125	Sokrates	W3	226	5041	Ethik	4
2125	Sokrates	W3	226	5049	Mäeutik	2
2125	Sokrates	W3	226	4052	Logik	4
2132	Popper	W2	52	5259	Der Wiener Kreis	2
2137	Kant	W3	7	4630	Die 3 Kritiken	4

- Schema kombiniert Informationen verschiedener unpassender Entitytypen
  - Löschen von Elementen eines Entitytyps kann Verlust eines anderen Entitytyps bewirken
    - Löschen des Eintrags zu „Der Wiener Kreis“ (die einzige Vorlesung von Popper) würde auch Informationen zu Popper löschen
    - Alternative: Prüfen der *gesamten* Datenbank, ob dieser Eintrag die einzige Vorlesung von Popper ist. In diesem Fall durch NULL-Werte ersetzen

Zerlegung

- nicht zusammenpassende Informationen
- Basis jeder Normalisierung:

  Zerlege das Relationenschema in Schemata

- Kriterien für eine korrekte Zerlegung?
  1. Verlustlosigkeit:
     - Die Informationen einer Relationsausprägung von müssen aus den resultierenden Ausprägungen wieder komplett rekonstruiert werden können.
  2. Abhängigkeitserhaltung:
     - Die für geltenden funktionalen Abhängigkeiten müssen sich auf übertragen lassen

  (Formalisierung auf den nächsten Folien)

Gültig, Verlustlos

Betrachte die Zerlegung von mit FDs in
Seien und die entsprechenden Attributmengen.

- *Gültigkeit*:

  Die Zerlegung von heißt *gültig*, falls gilt

- *Verlustlosigkeit*:

  Sei eine beliebige gültige Ausprägung von . Definiere

  Die Zerlegung von in heißt *verlustlos*, falls für alle gilt

Verlustlos – Beispiel

- Verlust von Informationen:

Relationsschema mit FD

Biertrinker		
Kneipe	Gast	Bier
Domkeller	Kemper	Pils
Domkeller	Eickler	Hefeweizen
Die Kiste	Eickler	Pils

(Kneipe, Gast und Bier werden eindeutig durch ihren Namen bestimmt)

- Welches Getränk ein Gast trinkt hängt von der besuchten Kneipe ab
- Zerlegung in

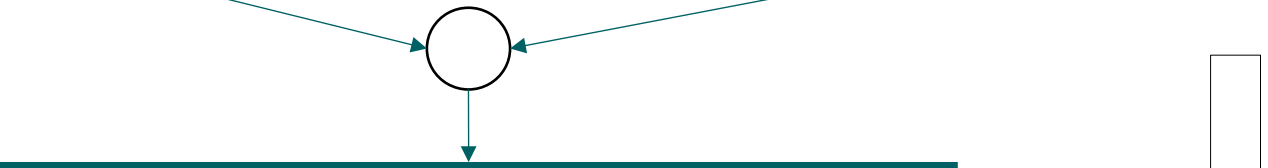
Besucht	
Kneipe	Gast
Domkeller	Kemper
Domkeller	Eickler
Die Kiste	Eickler

Trinkt	
Gast	Bier
Kemper	Pils
Eickler	Hefeweizen
Eickler	Pils

Nicht verlustlose Zerlegung – ein Beispiel

Besucht	
Kneipe	Gast
Domkeller	Kemper
Domkeller	Eickler
Die Kiste	Eickler

Trinkt	
Gast	Bier
Kemper	Pils
Eickler	Hefeweizen
Eickler	Pils



Besucht Trinkt		
Kneipe	Gast	Bier
Domkeller	Kemper	Pils
Domkeller	*Eickler*	*Hefeweizen*
Domkeller	Eickler	Pils
Die Kiste	*Eickler*	*Hefeweizen*
Die Kiste	Eickler	Pils

Ein Mehr an Tupeln bedeutet einen Verlust an Informationen!

Kriterien für Verlustlosigkeit

- Finde Bedingungen unter denen eine verlustlose Zerlegung garantiert ist
  Seien Relationenschemata mit den Attributmengen und wie vorher definiert.

- Hinreichend:

  Die Zerlegung von ist verlustlos, falls gilt

- In anderen Worten:

  Die Zerlegung von ist verlustlos, falls gilt

- Intutition: Joinattribut bestimmt eines der beiden Teilschemata

- Im Biertrinker-Beispiel: einzige nicht-triviale Abhängigkeit war

Verlustlos – Beispiel

- Verlustlose Zerlegung:

Relationsschema mit FDs

(biol.) Eltern		
Vater	Mutter	Kind
Johann	Martha	Else
Johann	Maria	Theo
Heinz	Martha	Cleo

(Personen sind durch ihren Vornamen eindeutig bestimmt)

Verlustlose Zerlegung

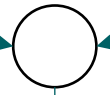
Väter	
Vater	Kind
Johann	Else
Johann	Theo
Heinz	Cleo

Mütter	
Mutter	Kind
Martha	Else
Maria	Theo
Martha	Cleo

Verlustlos – Beispiel

Väter	
Vater	Kind
Johann	Else
Johann	Theo
Heinz	Cleo

Mütter	
Mutter	Kind
Martha	Else
Maria	Theo
Martha	Cleo



Väter Mütter		
Vater	Mutter	Kind
Johann	Martha	Else
Johann	Maria	Theo
Heinz	Martha	Cleo

Abhängigkeitserhaltung

- Gegeben Zerlegung von mit FDs in die Schemata mit entsprechenden Attributmengen

  Die Überprüfung der Konsistenzbedinungen bei Einfügen, Löschen und Updaten auf

  benötigt Joins, falls funktionale Abhängigkeiten sich nach der Zerlegung auf Attribute verschiedenen Relationenschemata beziehen (Überprüfung bei jeder Transaktion). Ineffzient!
- Wunsch: alle FDs, die für das Schema gelten, sollen *lokal* (ohne Joins) auf den einzelnen Schemata überprüfbar sein.
- Formal*: Abhängigkeitserhaltend*:

  Sei die Menge der FDs mit , (also deren Attribute aus sind) .

  Die Zerlegung von in heißt *abhängigkeitserhaltend*, falls

  Diese Eigenschaft wird auch *Hüllentreue* der Zerlegung genannt.

Abhängigkeitserhaltung – Beispiel

- Beispiel: Verlustlose aber nicht abhängigkeitserhaltende Zerlegung

  Betrachte das Relationsschema

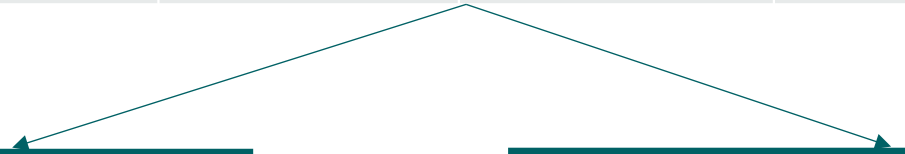
  PLZVerzeichnis(BLand, Ort, Straße, PLZ)

  - Annahme:
    - Ortsnamen sind innerhalb eines Bundeslandes eindeutig
    - PLZ'n ändern sich nicht innerhalb einer Straße
  - Funktionale Abhängigkeiten:

  - Betrachte die Zerlegung in

Abhängigkeitserhaltung – Beispiel

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



Straßen	
PLZ	Straße
60313	Goethestraße
60437	Galgenstraße
15234	Goethestraße

Orte		
Ort	Bland	PLZ
Frankfurt	Hessen	60313
Frankfurt	Hessen	60437
Frankfurt	Brandenburg	15234

Abhängigkeitserhaltung – Beispiel

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

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

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

Siehe funktionale Abhängigkeit: BLand, Ort, Straße → PLZ

Ungültiger Eintrag kann nur durch überprüft werden

Abhängigkeitserhaltung – Beispiel

- Zerlegung von PLZVerzeichnis(BLand, Ort, Straße, PLZ) in

- Also:
  - Die Zerlegung ist verlustlos, da ist und

    die funktionale Abhängigkeit

    gilt.

  - Die Zerlegung ist nicht abhängigkeitserhaltend, da die funktionale Abhängigkeit

    keiner der neuen Relationen Straßen oder Orte zugeordnet werden kann.

6. Relationale Entwurfstheorie

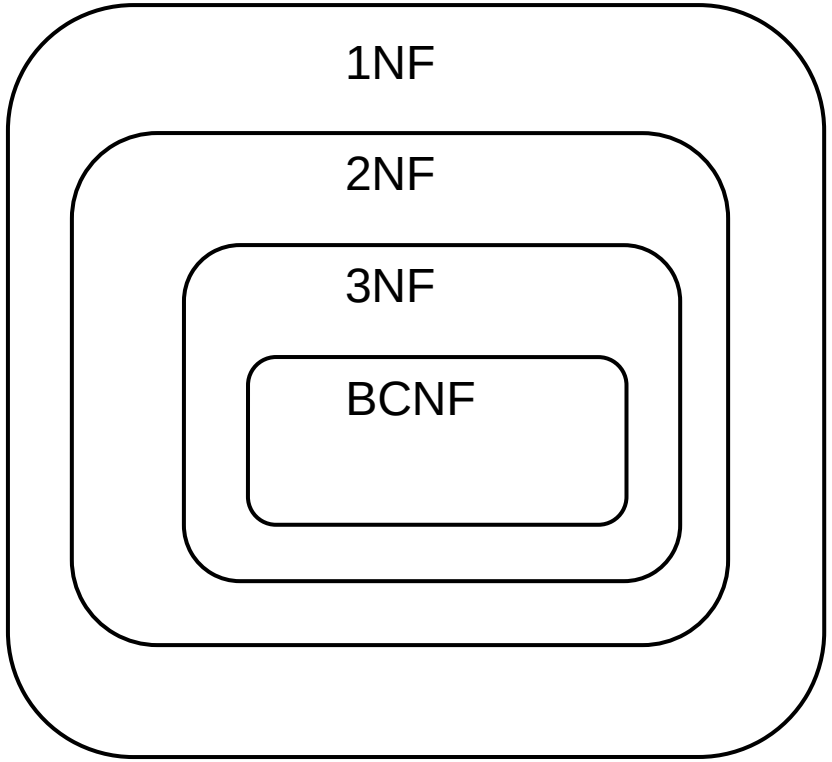
1. Funktionale Abhängigkeiten
2. Armstrong-Kalkül
3. Zerlegung von Relationen
4. **Normalformen und Normalisierungen**

Normalformen und Normalisierungen

•Vermeidung von Redundanzen und damit zusammenhängender Anomalien

- Normalformen unterscheiden sich in der Strenge der Anforderungen an die funktionalen Abhängigkeiten

- Normalisierungsalgorithmen berechnen zu einem Schema eine verlustlose und, *wenn möglich*, auch abhängigkeitserhaltende Zerlegung



Erste Normalform

- Alle Attribute haben atomare Wertebereiche (zB String, Integer, …)
  - zusammengesetzte, mengenwertige oder relationenwertige Attribute sind nicht erlaubt

- Beispiel: nicht in 1NF

Eltern-1		
Vater	Mutter	Kind
Johann	Martha	
Johann	Maria	
Heinz	Martha	

Eltern-2		
Vater	Mutter	Kind
Johann	Martha	Else
Johann	Martha	Lucia
Johann	Maria	Theo
Johann	Maria	Jose
Heinz	Martha	Cleo

NF²-Modelle:
non-first-normal-form Modelle

- Im Folgenden gehen wir stets von Relationen in 1NF aus

Zweite Normalform

- Betrachte wieder das Beispiel

ProfVorl						
PersNr	Name	Rang	Raum	VorlNr	Titel	SWS
2125	Sokrates	W3	226	5041	Ethik	4
2125	Sokrates	W3	226	5049	Mäeutik	2
2125	Sokrates	W3	226	4052	Logik	4

- Anschaulich ist die 2NF verletzt, wenn eine Relation Informationen aus mehreren Konzepten enthält (hier: ProfVorl entspricht)
- Attribute dürfen nicht von Teilmengen von Schlüsselkandidaten abhängen
- FDs:

Zweite Normalform (Definition)

- *Nichtschlüssel-Attribut (NSA)*:

  Seien die Schlüsselkandidaten einer gegebenen Relation und die zugehörige Attributmenge. Dann heißt die Menge aller *Nichtschlüssel-Attribute*.

- *Zweite Normalform*:

  Ein Relationenschema mit FDs ist in *zweiter Normalform*, falls für jedes Nichtschlüssel-Attribut und jeden Schlüsselkandidaten gilt:

  - ist voll funktional abhängig von , d.h. für kein Attribut gilt

- Beispiel:
  - Einziger Schlüsselkandidat von ProfVorl ist . Es gilt aber . Also ist Name nicht voll funktional abhängig von .

Zweite Normalform – Algorithmus (Skizze)

- Ein Relationenschema kann wie folgt in zweite Normalform überführt werden
  a) Fasse alle Nichtschlüsselattribute, die nur von einem Teilschlüssel abhängen, mit diesem Teilschlüssel als Primärschlüssel in einer eigenen Relation zusammen. Alle Attribute, die von demselben Teilschlüssel abhängen, müssen in derselben Relation zusammengefasst werden

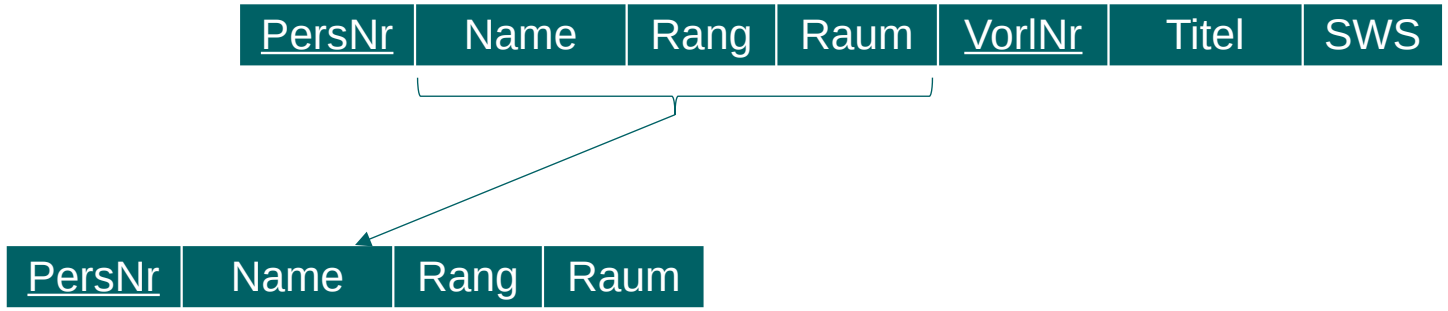
  b) Entferne die ausgelagerten Nichtschlüsselattribute aus der Ursprungsrelation und fasse die übriggebliebenen Attribute zu einer neuen Relation zusammen

PersNr	Name	Rang	Raum	VorlNr	Titel	SWS

Zweite Normalform – Algorithmus (Skizze)

- Ein Relationenschema kann wie folgt in zweite Normalform überführt werden
  a) Fasse alle Nichtschlüsselattribute, die nur von einem Teilschlüssel abhängen, mit diesem Teilschlüssel als Primärschlüssel in einer eigenen Relation zusammen. Alle Attribute, die von demselben Teilschlüssel abhängen, müssen in derselben Relation zusammengefasst werden

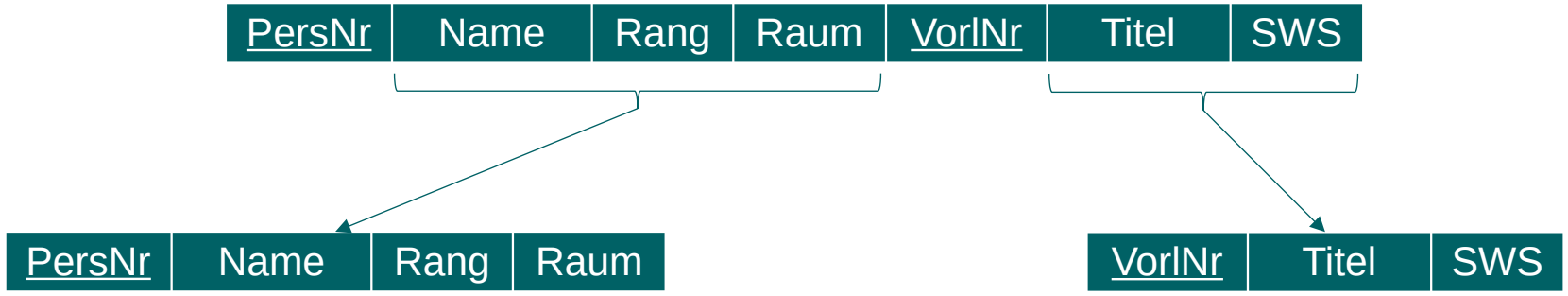
  b) Entferne die ausgelagerten Nichtschlüsselattribute aus der Ursprungsrelation und fasse die übriggebliebenen Attribute zu einer neuen Relation zusammen



Zweite Normalform – Algorithmus (Skizze)

- Ein Relationenschema kann wie folgt in zweite Normalform überführt werden
  a) Fasse alle Nichtschlüsselattribute, die nur von einem Teilschlüssel abhängen, mit diesem Teilschlüssel als Primärschlüssel in einer eigenen Relation zusammen. Alle Attribute, die von demselben Teilschlüssel abhängen, müssen in derselben Relation zusammengefasst werden

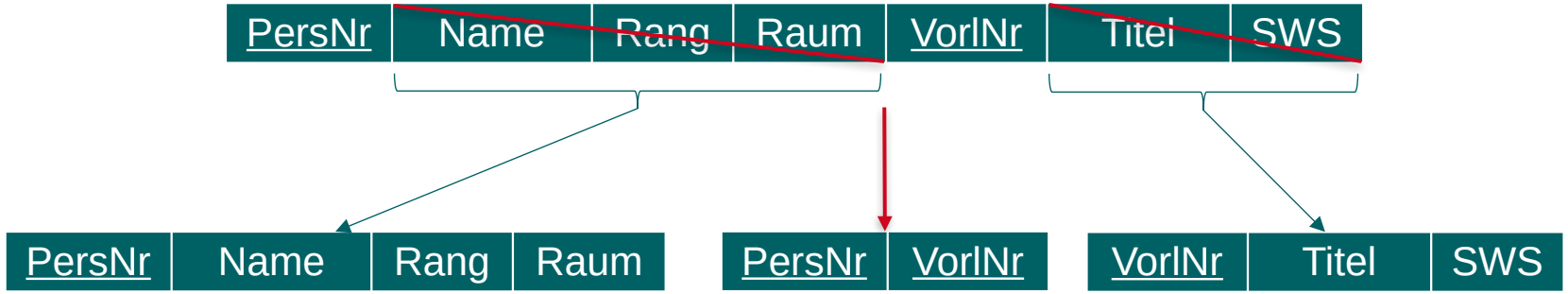
  b) Entferne die ausgelagerten Nichtschlüsselattribute aus der Ursprungsrelation und fasse die übriggebliebenen Attribute zu einer neuen Relation zusammen



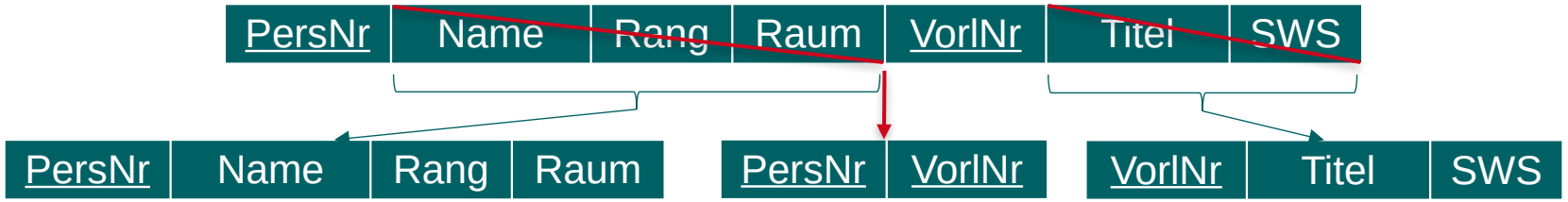
Zweite Normalform – Algorithmus (Skizze)

- Ein Relationenschema kann wie folgt in zweite Normalform überführt werden
  a) Fasse alle Nichtschlüsselattribute, die nur von einem Teilschlüssel abhängen, mit diesem Teilschlüssel als Primärschlüssel in einer eigenen Relation zusammen. Alle Attribute, die von demselben Teilschlüssel abhängen, müssen in derselben Relation zusammengefasst werden

  b) Entferne die ausgelagerten Nichtschlüsselattribute aus der Ursprungsrelation und fasse die übriggebliebenen Attribute zu einer neuen Relation zusammen



Zweite Normalform – Beispiel



PersNr	Name	Rang	Raum	VorlNr	Titel	SWS
2125	Sokrates	W3	226	5041	Ethik	4
2125	Sokrates	W3	226	5049	Mäeutik	2
2125	Sokrates	W3	226	4052	Logik	4

PersNr	Name	Rang	Raum
2125	Sokrates	W3	226

PersNr	VorlNr
2125	5041
2125	5049
2125	4052

VorlNr	Titel	SWS
5041	Ethik	4
5049	Mäeutik	2
4052	Logik	4

Dritte Normalform – Motivation

- Beispiel: Erweiterung des Universitäts-Beispiels um Adressen

ProfessorenAdressen						
PersNr	Name		Ort		PLZ	
2125	Sokrates		Aachen		52052	
2132	Popper		Aachen		52052	

- Diese Relation ist in zweiter Normalform
- Die Information, dass die PLZ 52052 zu Aachen gehört ist hier dennoch redundant
- Ursache: funktionale Abhängigkeit zwischen NSAs

Dritte Normalform (Definition)

- *Dritte Normalform*:

  Ein Relationenschema mit FDs ist in *dritter Normalform*, falls für jedes Nichtschlüssel-Attribut und jede nicht-triviale Abhängigkeit gilt:

  - ist ein Superschlüssel (d.h. nicht notwendig minimal)

- Die zweite Normalform lässt auch Abhängigkeiten zwischen NSAs zu
  - In dritter Normalform dürfen NSAs im Endeffekt nur von Schlüsselkandidaten abhängen

- Beispiel:

  - Sei die Attributmenge von . Es gelten die FDs:

nicht erlaubt!

Dritte Normalform – Synthesealgorithmus

- Synthesealgorithmus:

  Eingabe: Relationenschema mit FDs und Attributmenge

  Ausgabe: verlustlose, abhängigkeitserhaltende 3NF-Zerlegung

  1. Bestimme die kanonische Überdeckung zu

     Zur Wiederholung:

     a) Linksreduktion der FDs
     b) Rechtsreduktion der FDs
     c) Entfernung der FDs der Form
     d) Zusammenfassung von FDs mit gleichen linken Seiten

Dritte Normalform – Synthesealgorithmus

- Synthesealgorithmus:

  Eingabe: Relationenschema mit FDs und Attributmenge

  Ausgabe: verlustlose, abhängigkeitserhaltende 3NF-Zerlegung

  1. Bestimme die kanonische Überdeckung zu
  2. Für jede funktionale Abhängigkeit :
     - Erstelle Relationenschema mit Attributmenge
     - Ordne die FDs zu
  3. Falls ein in Schritt 2 erzeugtes Schema einen Schlüsselkandidaten von enthält:
     fertig mit Schritt 3

     sonst: wähle einen Schlüsselkandidaten von aus und erstelle das zusätzliche Schema mit funktionaler Abhängigkeit
  4. Eliminiere jedes Schema dessen Attributmenge in der eines größeren Schemas enthalten ist

Dritte Normalform – Synthesealgorithmus (Beispiel)

- Beispiel:
- Funktionale Abhängigkeiten:
  –
  –

Dritte Normalform – Synthesealgorithmus (Beispiel)

- Beispiel:

1. die kanonische Überdeckung enthält die FDs

,

,

2. erzeugte Relationenschemata:

aus : mit

aus : mit

aus : mit

aus : mit

aus : mit

Dritte Normalform – Synthesealgorithmus (Beispiel)

- Beispiel:

2. erzeugte Relationenschemata:

aus : mit

aus : mit

aus : mit

aus : mit

aus : mit

3. ist ein Schlüsselkandidat des gesamten Schemas

Es muss keine neue Relation erzeugt werden

4. und werden entfernt, da sie in bzw. enthalten sind

Boyce-Codd Normalform

- Ziel: Jede Information wird genau einmal gespeichert
- *BCNF*:

  Ein Relationsschema mit FDs ist genau dann in *Boyce-Codd Normalform*, wenn für alle nicht-trivialen FDs gilt:

  - ist ein Superschlüssel von

- Beispiel: In dritter Normalform aber nicht in BCNF

  - zwei Schlüsselkandidaten

  - ist in 3NF: einziges NSA ist . kommt nur in vor
  - nicht in BCNF: , die linke Seite ist kein Schlüssel

BCNF – Dekompositionsalgorithmus

- Achtung: Die resultierende Zerlegung ist i.A. nicht abhängigkeitserhaltend!

  Starte mit

  Solange es noch ein Schema gibt, das nicht in BCNF ist:

  - finde nicht-triviale, in geltende FD mit
    - und (ist Attributmenge von)
  - Zerlege in),) *entlang* der FD

    mit Attributmengen:

    und

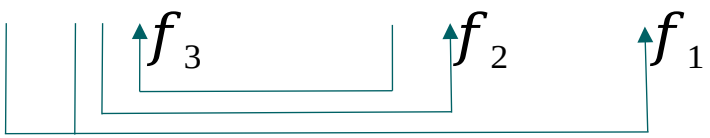
    und FDs:

    und
  - Entferne aus und füge hinzu.

Definition:

$$\Pi_X(F) := \{\alpha \rightarrow \beta \in F \mid (\alpha \cup \beta) \subseteq X\}$$

Dekompositionsalgorithmus Beispiel 1



- Zerlegung *entlang* von
  - mit FDs:

    -

  - mit FDs:

wegen in **(wichtig!)**

in diesem Beispiel ist die Zerlegung abhängigkeitserhaltend, da sich rekonstruieren lässt.

- **Bitte beachten:** BLand wurde durch MinisterpräsidentIn in ersetzt!
- Die Relationen Städte1 und Regierungen sind nun in BNCF.

Dekompositionsalgorithmus Beispiel 2

- Beispiel für nicht abhängigkeitserhaltende Zerlegung

  mit funktionalen Abhängigkeiten

  Die Zerlegung von entlang von ergibt

  - mit FDs:
  - mit FDs:

  Wie schon diskutiert, geht in diesem Fall die Abhängigkeit verloren

- Wäre eine Zerlegung nicht abhängigkeitserhaltend, gibt man sich normalerweise mit der dritten Normalform zufrieden

Zusammenfassung

- Ziel der relationalen Entwurfstheorie
  - Was ist ein redundanzfreies relationales Datenbank-Schema?
- Zerlegung von Relationenschemata
  - Verlustlos
  - Abhängigkeitserhaltend
- Normalformen
  - Algorithmen
    - Synthesealgorithmus (3NF)
    - Dekompositionsalgorithmus (BCNF)
  - Abhängigkeitserhaltung ist nur bis zur dritten Normalform garantiert