184.686 VU Datenbanksysteme

Relationale Entwurfstheorie

Katja Hose

Institut für Logic and Computation

Sommersemester 2024

TU WIEN Informatics

Lernziele

Lernziele

- Funktionale Abhängigkeiten und Normalformen verstehen
- Die Güte eines Entwurfs mit Hilfe von Normalformen beschreiben
- Einen Datenbankentwurf mit Hilfe von Zerlegung verbessern

Motivation

- Ein guter Entwurf verhindert Probleme, die während der späteren Verwendung auftreten.
- Datenbankschemata lassen sich viel leichter verändern, bevor sie in Verwendung sind.
- Redundanz ist die Quelle vieler Probleme und sollte durch bestimmte Entwurfskriterien vermieden werden.

1. Einführung
   - Anomalien
   - Gute und schlechte Zerlegungen
2. Funktionale Abhängigkeiten
   - Definition
   - Schlüssel
   - Ableitung funktionaler Abhängigkeiten
   - Berechnung der Attributhülle
   - Kanonische Überdeckung (minimale Überdeckung)
3. Normalisieren durch Zerlegung der Relationen
   - Verlustlose Zerlegung
   - Abhängigkeitsbewahrung
4. Normalformen
   - Erste Normalform
   - Zweite Normalform
   - Dritte Normalform

- Boyce-Codd-Normalform

Anomalien

courseOffers						
empID	teacher	rank	office	courseID	title	ects
2125	Socrates	C4	226	5041	Robotics	4
2125	Socrates	C4	226	5049	Ethics	2
2125	Socrates	C4	226	4052	Logic	4
2133	Curie	C3	52	5259	Chemistry	2
2137	Curie	C4	7	4630	Physics	4
...	...	...	...	...	...	...

Wieso ist das ein schlechtes Schema?

Anomalien

courseOffers						
empID	teacher	rank	office	courseID	title	ects
2125	Socrates	C4	226	5041	Robotics	4
2125	Socrates	C4	226	5049	Ethics	2
2125	Socrates	C4	226	4052	Logic	4
2133	Curie	C3	52	5259	Chemistry	2
2137	Curie	C4	7	4630	Physics	4
$\perp$	$\perp$	$\perp$	$\perp$	5432	Economy	4
...	...	...	...	...	...	...

Wieso ist das ein schlechtes Schema?

Anomalien

courseOffers						
empID	teacher	rank	office	courseID	title	ects
2125	Socrates	C4	226	5041	Robotics	4
2125	Socrates	C4	226	5049	Ethics	2
2125	Socrates	C4	226	4052	Logic	4
2133	Curie	C3	52	5259	Chemistry	2
2137	Curie	C4	7	4630	Physics	4
$\perp$	$\perp$	$\perp$	$\perp$	5432	Economy	4
...	...	...	...	...	...	...

Hauptquellen für Probleme

- Information mehrfach (Redundanz)
- Speicherplatzverschwendung
- Professor:innen ohne LVAs sowie LVAs ohne Professor:innen

Anomalien

courseOffers						
empID	teacher	rank	office	courseID	title	ects
2125	Socrates	C4	226	5041	Robotics	4
2125	Socrates	C4	226	5049	Ethics	2
2125	Socrates	C4	226	4052	Logic	4
2133	Curie	C3	52	5259	Chemistry	2
2137	Curie	C4	7	4630	Physics	4
$\perp$	$\perp$	$\perp$	$\perp$	5432	Economy	4
...	...	...	...	...	...	...

- **Änderungsanomalien:**
- **Einfügeanomalien:**
- **Löschanomalien:**

Anomalien

courseOffers						
empID	teacher	rank	office	courseID	title	ects
2125	Socrates	C4	226	5041	Robotics	4
2125	Socrates	C4	226	5049	Ethics	2
2125	Socrates	C4	226	4052	Logic	4
2133	Curie	C3	52	5259	Chemistry	2
2137	Curie	C4	7	4630	Physics	4
$\perp$	$\perp$	$\perp$	$\perp$	5432	Economy	4
...	...	...	...	...	...	...

- **Änderungsanomalien:** Sokrates zieht um – von Raum 226 in Raum 338. Anomalie, wenn mache Tupeln verändert werden, aber nicht alle (Murphy's law).
- **Einfügeanomalien:**
- **Löschanomalien:**

Anomalien

courseOffers						
empID	teacher	rank	office	courseID	title	ects
2125	Socrates	C4	226	5041	Robotics	4
2125	Socrates	C4	226	5049	Ethics	2
2125	Socrates	C4	226	4052	Logic	4
2133	Curie	C3	52	5259	Chemistry	2
2137	Curie	C4	7	4630	Physics	4
$\perp$	$\perp$	$\perp$	$\perp$	5432	Economy	4
...	...	...	...	...	...	...

- **Änderungsanomalien:** Sokrates zieht um – von Raum 226 in Raum 338. Anomalie, wenn mache Tupeln verändert werden, aber nicht alle (Murphy's law).
- **Einfügeanomalien:** Neue:r Professor:in ohne LVA nicht möglich ohne Null-Werten.
- **Löschanomalien:**

Anomalien

courseOffers						
empID	teacher	rank	office	courseID	title	ects
2125	Socrates	C4	226	5041	Robotics	4
2125	Socrates	C4	226	5049	Ethics	2
2125	Socrates	C4	226	4052	Logic	4
2133	Curie	C3	52	5259	Chemistry	2
2137	Curie	C4	7	4630	Physics	4
$\perp$	$\perp$	$\perp$	$\perp$	5432	Economy	4
...	...	...	...	...	...	...

- **Änderungsanomalien:** Sokrates zieht um – von Raum 226 in Raum 338. Anomalie, wenn mache Tupeln verändert werden, aber nicht alle (Murphy's law).
- **Einfügeanomalien:** Neue:r Professor:in ohne LVA nicht möglich ohne Null-Werten.
- **Löschanomalien:** Letzte LVA von einem:einer Professor:in wird gelöscht. Gesamte Information über eine:n Professor:in geht möglicherweise verloren.

Anomalien

courseOffers						
empID	teacher	rank	office	courseID	title	ects
2125	Socrates	C4	226	5041	Robotics	4
2125	Socrates	C4	226	5049	Ethics	2
2125	Socrates	C4	226	4052	Logic	4
2133	Curie	C3	52	5259	Chemistry	2
2137	Curie	C4	7	4630	Physics	4
$\perp$	$\perp$	$\perp$	$\perp$	5432	Economy	4
...	...	...	...	...	...	...

Ziel eines Datenbankentwurfs

- Vermeidung von Redundanzen, Null-Werten und Anomalien
- Alle Beziehungen zwischen Attributen abbilden

Strategie

- Alle gegebenen funktionalen Abhängigkeiten in **Schlüsselabhängigkeiten** transformieren, ohne dabei semantische Informationen zu verlieren.

Eine gute Zerlegung

course			
courseID	title	ects	empID
5041	Robotics	4	2125
5049	Ethics	2	2125
4052	Logic	4	2125
5259	Chemistry	2	2133
4630	Physics	4	2137

instructor			
empID	teacher	rank	office
2125	Socrates	C4	226
2133	Curie	C4	36
2137	Curie	C4	7

$\pi_{courseID,title,ects,empID}(courseOffers)$

$\pi_{empID,teacher,rank,office}(courseOffers)$

courseOffers						
empID	teacher	rank	office	courseID	title	ects
2125	Socrates	C4	226	5041	Robotics	4
2125	Socrates	C4	226	5049	Ethics	2
2125	Socrates	C4	226	4052	Logic	4
2133	Curie	C3	52	5259	Chemistry	2
2137	Curie	C4	7	4630	Physics	4

$$courseOffers = instructor \bowtie course$$

Eine gute Zerlegung

Grundidee zur Vermeidung von Redundanz, Anomalien etc.

- Zerlege courseOffers in:
  - course (courseID, title, ects, empID)
  - instructor (empID, teacher, rank, office)

Anforderungen

- Alle Attribute des originalen Schemas müssen in der Zerlegung vorkommen: $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$
- Zerlegung mit verlustfreiem Join: für Relationen R mit Schema $\mathcal{R}$ gilt: $R = \pi_{\mathcal{R}_1}(R) \bowtie \pi_{\mathcal{R}_2}(R)$

Eine schlechte Zerlegung

course			
courseID	title	ects	teacher
5041	Robotics	4	Socrates
5049	Ethics	2	Socrates
4052	Logic	4	Socrates
5259	Chemistry	2	Curie
4630	Physics	4	Curie

instructor			
empID	teacher	rank	office
2125	Socrates	C4	226
2133	Curie	C4	36
2137	Curie	C4	7

$\pi_{courseID,title,ects,teacher}(courseOffers)$ $\pi_{empID,teacher,rank,office}(courseOffers)$

Eine schlechte Zerlegung

course			
courseID	title	ects	teacher
5041	Robotics	4	Socrates
5049	Ethics	2	Socrates
4052	Logic	4	Socrates
5259	Chemistry	2	Curie
4630	Physics	4	Curie

instructor			
empID	teacher	rank	office
2125	Socrates	C4	226
2133	Curie	C4	36
2137	Curie	C4	7

$\pi_{courseID,title,ects,teacher}(courseOffers)$ $\pi_{empID,teacher,rank,office}(courseOffers)$

courseOffers						
empID	teacher	rank	office	courseID	title	ects
2125	Socrates	C4	226	5041	Robotics	4
2125	Socrates	C4	226	5049	Ethics	2
2125	Socrates	C4	226	4052	Logic	4
2133	Curie	C3	36	5259	Chemistry	2
2133	Curie	C3	36	4630	Physics	4
2137	Curie	C4	7	5259	Chemistry	2
2137	Curie	C4	7	4630	Physics	4
...	...	...	...	...	...	...

$$courseOffers = instructor \bowtie course$$

funktionale Abhängigkeiten

Symbole

- Schema $\mathcal{R} = \{A, B, C, D\}$
- Ausprägung/Instanz R
- Attributmengen $\alpha \subseteq \mathcal{R}$ und $\beta \subseteq \mathcal{R}$

Eine **funktionale Abhängigkeit** $\alpha \rightarrow \beta$ in $\mathcal{R}$ liegt genau dann vor, wenn für alle legalen Ausprägungen R von $\mathcal{R}$ gilt:

$$\forall r, s \in R : r.\alpha = s.\alpha \Rightarrow r.\beta = s.\beta$$

Die α-Werte identifizieren die β-Werte eindeutig.
Bzw.: Die α-Werte bestimmen die β-Werte funktional.

Eine funktionale Abhängigkeit $\alpha \rightarrow \beta$ wird **trivial** genannt, wenn $\beta \subseteq \alpha$.

Beispiele

Wenn ich die Postleitzahl kenne, dann kenne ich die Stadt

- Die Postleitzahl 1040 gibt mir genau eine Stadt: Wien
- Die Postleitzahl 3400 gibt mir nicht { Wien, Klosterneuburg }

Notation: { Postleitzahl } $\rightarrow$ { Stadt }

Beispiele

Wenn ich die Postleitzahl kenne, dann kenne ich die Stadt

- Die Postleitzahl 1040 gibt mir genau eine Stadt: Wien
- Die Postleitzahl 3400 gibt mir nicht { Wien, Klosterneuburg }

Notation: { Postleitzahl } $\rightarrow$ { Stadt }

Die Sozialversicherungsnummer (VSNR) gibt mir einen Namen

- VSNR 8476120478 gibt mir Sean Connery

Notation: { VSNR } $\rightarrow$ { Vorname, Nachname }

Funktionale Abhängigkeiten

R			
A	B	C	D
a4	b2	c4	d3
a1	b1	c1	d1
a1	b1	c1	d2
a2	b2	c3	d2
a3	b2	c4	d3

Welche funktionalen Abhängigkeiten (FDs, Functional Dependencies) sind erfüllt?

$\{A\} \rightarrow \{A\}$
$\{A\} \rightarrow \{B\}$
$\{A\} \rightarrow \{C\}$
$\{A\} \rightarrow \{D\}$
$\{A\} \rightarrow \{C, D\}$

$\{B\} \rightarrow \{D\}$
$\{B\} \rightarrow \{A\}$
$\{C, D\} \rightarrow \{A\}$
$\{C, D\} \rightarrow \{B\}$

Funktionale Abhängigkeiten

R			
A	B	C	D
a4	b2	c4	d3
a1	b1	c1	d1
a1	b1	c1	d2
a2	b2	c3	d2
a3	b2	c4	d3

Welche funktionalen Abhängigkeiten (FDs, Functional Dependencies) sind erfüllt?

$\{A\} \rightarrow \{A\}$ ja
$\{A\} \rightarrow \{B\}$ ja
$\{A\} \rightarrow \{C\}$ ja
$\{A\} \rightarrow \{D\}$ nein
$\{A\} \rightarrow \{C, D\}$ nein

$\{B\} \rightarrow \{D\}$ nein
$\{B\} \rightarrow \{A\}$ nein
$\{C, D\} \rightarrow \{A\}$ nein
$\{C, D\} \rightarrow \{B\}$ ja

Funktionale Abhängigkeiten

R			
A	B	C	D
a4	b2	c4	d3
a1	b1	c1	d1
a1	b1	c1	d2
a2	b2	c3	d2
a3	b2	c4	d3

Welche funktionalen Abhängigkeiten (FDs, Functional Dependencies) sind erfüllt?

$\{A\} \rightarrow \{A\}$ ja
$\{A\} \rightarrow \{B\}$ ja
$\{A\} \rightarrow \{C\}$ ja
$\{A\} \rightarrow \{D\}$ nein
$\{A\} \rightarrow \{C, D\}$ nein

$\{B\} \rightarrow \{D\}$ nein
$\{B\} \rightarrow \{A\}$ nein
$\{C, D\} \rightarrow \{A\}$ nein
$\{C, D\} \rightarrow \{B\}$ ja

Vereinfachte Notation: $CD \rightarrow B$

Funktionale Abhängigkeiten sind **semantische Konsistenzbedingungen**, die sich aus der jeweiligen Anwendungssemantik und **nicht** aus der aktuellen Ausprägung einer Relation ergeben. Sie müssen zu jedem (gültigen) Datenbankzustand eingehalten werden.

Schlüssel

Superschlüssel

- $\alpha \subseteq \mathcal{R}$ ist ein **Superschlüssel** wenn $\alpha \rightarrow \mathcal{R}$,
  d.h., α bestimmt alle anderen Attributwerte
- Die Menge aller Attribute bildet einen Superschlüssel $\mathcal{R} \rightarrow \mathcal{R}$
- Superschlüssel sind nicht notwendigerweise minimal!

Schlüssel

Superschlüssel

- $\alpha \subseteq \mathcal{R}$ ist ein **Superschlüssel** wenn $\alpha \rightarrow \mathcal{R}$,
  d.h., α bestimmt alle anderen Attributwerte
- Die Menge aller Attribute bildet einen Superschlüssel $\mathcal{R} \rightarrow \mathcal{R}$
- Superschlüssel sind nicht notwendigerweise minimal!

Volle funktionale Abhängigkeit

β ist **voll funktional abhängig** von α wenn:

- $\alpha \rightarrow \beta$ und
- α kann nicht mehr verkleinert (=linksreduziert) werden, d.h.

$$\forall A \in \alpha : (\alpha - \{A\}) \not\rightarrow \beta)$$

Schlüssel

Schlüsselkandidat

$\alpha \subseteq \mathcal{R}$ ist ein **Schlüsselkandidat**, wenn $\mathcal{R}$ **voll funktional abhängig** von α ist.

Ein Schlüsselkandidatwird als **Primärschlüssel** ausgewählt!

Bestimmung funktionaler Abhängigkeiten

Relation professor:
{[empID, name, rank, office, city, street, zip, dialCode, region, inhabitants, government]}

Funktionale Abhängigkeiten

- $\{\text{empID}\} \rightarrow$
- $\{\text{city, region}\} \rightarrow$
- $\{\text{zip}\} \rightarrow$
- $\{\text{region, city, street}\} \rightarrow$
- $\{\text{region}\} \rightarrow$
- $\{\text{office}\} \rightarrow$

Bestimmung funktionaler Abhängigkeiten

Relation professor:
{[empID, name, rank, office, city, street, zip, dialCode, region, inhabitants, government]}

Funktionale Abhängigkeiten

- {empID} → {empID, name, rank, office, city, street, zip, dialCode, region, inhabitants, government} **Kandidatenschlüssel**
- {city, region} →
- {zip} →
- {region, city, street} →
- {region} →
- {office} →

Bestimmung funktionaler Abhängigkeiten

Relation professor:
{[empID, name, rank, office, city, street, zip, dialCode, region, inhabitants, government]}

Funktionale Abhängigkeiten

- $\{\text{empID}\} \rightarrow \{\text{empID, name, rank, office, city, street, zip, dialCode, region, inhabitants, government}\}$ **Kandidatenschlüssel**
- $\{\text{city, region}\} \rightarrow \{\text{inhabitants, dialCode}\}$
- $\{\text{zip}\} \rightarrow \{\text{region, city, inhabitants}\}$
- $\{\text{region, city, street}\} \rightarrow \{\text{zip}\}$
- $\{\text{region}\} \rightarrow \{\text{government}\}$
- $\{\text{office}\} \rightarrow \{\text{empID}\}$

Bestimmung funktionaler Abhängigkeiten

Funktionale Abhängigkeiten

- $\{\text{empID}\} \rightarrow \{\text{empID, name, rank, office, city, street, zip, dialCode, region, inhabitants, government}\}$ **Kandidatenschlüssel**
- $\{\text{city, region}\} \rightarrow \{\text{inhabitants, dialCode}\}$
- $\{\text{zip}\} \rightarrow \{\text{region, city, inhabitants}\}$
- $\{\text{region, city, street}\} \rightarrow \{\text{zip}\}$
- $\{\text{region}\} \rightarrow \{\text{government}\}$
- $\{\text{office}\} \rightarrow \{\text{empID}\}$

Davon können weitere abgeleitet werden

- $\{\text{office}\} \rightarrow$

Bestimmung funktionaler Abhängigkeiten

Funktionale Abhängigkeiten

- $\{\text{empID}\} \rightarrow \{\text{empID, name, rank, office, city, street, zip, dialCode, region, inhabitants, government}\}$ **Kandidatenschlüssel**
- $\{\text{city, region}\} \rightarrow \{\text{inhabitants, dialCode}\}$
- $\{\text{zip}\} \rightarrow \{\text{region, city, inhabitants}\}$
- $\{\text{region, city, street}\} \rightarrow \{\text{zip}\}$
- $\{\text{region}\} \rightarrow \{\text{government}\}$
- $\{\text{office}\} \rightarrow \{\text{empID}\}$

Davon können weitere abgeleitet werden

- $\{\text{office}\} \rightarrow \{\text{empID, name, rank, office, city, street, zip, dialCode, city, inhabitants, government}\}$

Bestimmung funktionaler Abhängigkeiten

Funktionale Abhängigkeiten

- $\{\text{empID}\} \rightarrow \{\text{empID, name, rank, office, city, street, zip, dialCode, region, inhabitants, government}\}$ **Kandidatenschlüssel**
- $\{\text{city, region}\} \rightarrow \{\text{inhabitants, dialCode}\}$
- $\{\text{zip}\} \rightarrow \{\text{region, city, inhabitants}\}$
- $\{\text{region, city, street}\} \rightarrow \{\text{zip}\}$
- $\{\text{region}\} \rightarrow \{\text{government}\}$
- $\{\text{office}\} \rightarrow \{\text{empID}\}$

Davon können weitere abgeleitet werden

- $\{\text{office}\} \rightarrow \{\text{empID, name, rank, office, city, street, zip, dialCode, city, inhabitants, government}\}$
- $\{\text{zip}\} \rightarrow$

Bestimmung funktionaler Abhängigkeiten

Funktionale Abhängigkeiten

- $\{\text{empID}\} \rightarrow \{\text{empID, name, rank, office, city, street, zip, dialCode, region, inhabitants, government}\}$ **Kandidatenschlüssel**
- $\{\text{city, region}\} \rightarrow \{\text{inhabitants, dialCode}\}$
- $\{\text{zip}\} \rightarrow \{\text{region, city, inhabitants}\}$
- $\{\text{region, city, street}\} \rightarrow \{\text{zip}\}$
- $\{\text{region}\} \rightarrow \{\text{government}\}$
- $\{\text{office}\} \rightarrow \{\text{empID}\}$

Davon können weitere abgeleitet werden

- $\{\text{office}\} \rightarrow \{\text{empID, name, rank, office, city, street, zip, dialCode, city, inhabitants, government}\}$
- $\{\text{zip}\} \rightarrow \{\text{government}\}$

Herleitung weiterer FDs

Aus einer Menge von FDs F sind weitere FDs herleitbar

- F^+ beinhaltet alle FDs, die aus F abgeleitet werden können, d.h., alle FDs, die von FDs in F **logisch impliziert** werden.
- F^+ wird **Hülle** (closure) von F genannt
- Inferenzregeln (Armstrong-Axiome) beschreiben die Herleitung

Die Armstrong-Axiome

α, β, γ, δ sind Teilmengen der Attribute aus $\mathcal{R}$

Reflexivität:
Falls $\beta \subseteq \alpha$, dann $\alpha \rightarrow \beta$ Insbesondere: $\alpha \rightarrow \alpha$

Erweiterung/Verstärkung:
Falls $\alpha \rightarrow \beta$, dann $\alpha\gamma \rightarrow \beta\gamma$

Transitivität:
Falls $\alpha \rightarrow \beta$ und $\beta \rightarrow \gamma$, dann $\alpha \rightarrow \gamma$

Die Armstrong-Axiome

α, β, γ, δ sind Teilmengen der Attribute aus $\mathcal{R}$

Reflexivität:
Falls $\beta \subseteq \alpha$, dann $\alpha \rightarrow \beta$ Insbesondere: $\alpha \rightarrow \alpha$

Erweiterung/Verstärkung:
Falls $\alpha \rightarrow \beta$, dann $\alpha\gamma \rightarrow \beta\gamma$

Transitivität:
Falls $\alpha \rightarrow \beta$ und $\beta \rightarrow \gamma$, dann $\alpha \rightarrow \gamma$

Die Armstrong-Axiome sind **korrekt** und **vollständig**.

- Sie sind korrekt in dem Sinne, dass sich mit ihnen nur korrekte funktionale Abhängigkeiten herleiten lassen.
- Sie sind vollständig in dem Sinne, dass sich mit ihnen alle möglichen FDs (F^+) von F herleiten lassen.

Weitere Ableitungsregeln

Nicht zwingend notwendig, aber oftmals komfortabel für Herleitungen

Vereinigung:
Wenn $\alpha \to \beta$ und $\alpha \to \gamma$, dann auch $\alpha \to \beta\gamma$

Dekomposition:
Wenn $\alpha \to \beta\gamma$, dann auch $\alpha \to \beta$ und $\alpha \to \gamma$

Pseudotransitivität:
Wenn $\alpha \to \beta$ und $\gamma\beta \to \delta$, dann auch $\alpha\gamma \to \delta$

Beispiel

Gegeben seien die folgenden FDs F, leite mithilfe der Armstrong-Axiome weitere her.

- A $\rightarrow$ BC
- CD $\rightarrow$ E
- B $\rightarrow$ D
- E $\rightarrow$ A

Hergeleitete FDs

- E $\rightarrow$ A und A $\rightarrow$ BC, dann
- B $\rightarrow$ D, dann
- CB $\rightarrow$ CD und CD $\rightarrow$ E, dann

Beispiel

Gegeben seien die folgenden FDs F, leite mithilfe der Armstrong-Axiome weitere her.

- A $\rightarrow$ BC
- CD $\rightarrow$ E
- B $\rightarrow$ D
- E $\rightarrow$ A

Hergeleitete FDs

- E $\rightarrow$ A und A $\rightarrow$ BC, dann E $\rightarrow$ BC (Transitivität)
- B $\rightarrow$ D, dann CB $\rightarrow$ CD (Erweiterung/Verstärkung)
- CB $\rightarrow$ CD und CD $\rightarrow$ E, dann CB $\rightarrow$ E (Transitivität)

Attributhülle

Die **Attributhülle** (α^+) bezüglich einer Menge von FDs F und einer Menge von Attributen α ist

$$\alpha^+ = \{A \mid \alpha \rightarrow A \in F^+\}$$

Beobachtung:
Wenn $\alpha \rightarrow \beta$ in F^+ ist, dann ist β in α^+

Algorithmus zur Bestimmung der Attributhülle

Algorithmus attrClosure(F, α):
Input

- eine Menge von FDs F
- eine Menge von Attributen $\alpha \subseteq \mathcal{R}$

result $= \alpha$
repeat
  for each $\beta \rightarrow \gamma$ in F **do**
    if $\beta \subseteq$ result **then**
      result $=$ result $\cup\, \gamma$
    end if
  end for
until result changes no more

Algorithmus zur Bestimmung der Attributhülle

Algorithmus attrClosure(F, α):

result $= \alpha$
repeat
  for each $\beta \to \gamma$ in F **do**
    if $\beta \subseteq$ result **then**
      result $=$ result $\cup \gamma$
    end if
  end for
until result changes no more

Der Algorithmus berechnet die Attributhülle von α korrekt

- Bei Termination: alle Teile der Attributhülle von α sind in result.
- Der Algorithmus terminiert.

Algorithmus zur Bestimmung der Attributhülle

Algorithmus attrClosure(F, α):

result $= \alpha$
repeat
  for each $\beta \to \gamma$ in F **do**
    if $\beta \subseteq$ result **then**
      result $=$ result $\cup\, \gamma$
    end if
  end for
until result changes no more

Anwendungen:
Testen, ob eine funktionale Abhängigkeit $\alpha \to \beta$ gilt.
Testen, on eine Menge von Attributen $\kappa \subseteq \mathcal{R}$ ein Superschlüssel ist.

Algorithmus zur Bestimmung der Attributhülle

Algorithmus attrClosure(F, α):

result $= \alpha$
repeat
  for each $\beta \to \gamma$ in F **do**
    if $\beta \subseteq$ result **then**
      result $=$ result $\cup \gamma$
    end if
  end for
until result changes no more

Anwendungen:
Testen, ob eine funktionale Abhängigkeit $\alpha \to \beta$ gilt.
Testen, on eine Menge von Attributen $\kappa \subseteq \mathcal{R}$ ein Superschlüssel ist.

Wie und warum können wir diesen Algorithmus verwenden, um auf Superschlüssel zu testen?

Algorithmus zur Bestimmung der Attributhülle

Algorithmus attrClosure(F, α):

result $= \alpha$
repeat
  for each $\beta \to \gamma$ in F **do**
    if $\beta \subseteq$ result **then**
      result $=$ result $\cup\, \gamma$
    end if
  end for
until result changes no more

Anwendungen:
Testen, ob eine funktionale Abhängigkeit $\alpha \to \beta$ gilt.
Testen, on eine Menge von Attributen $\kappa \subseteq \mathcal{R}$ ein Superschlüssel ist.

Durch Aufruf attrClosure(F,κ) erhalten wir κ^+.
Falls $\kappa^+ = \mathcal{R}$, dann ist κ ein Superschlüssel von $\mathcal{R}$.

Beispiel: Ausführung von attrClosure

result $= \alpha$
repeat
  for each $\beta \rightarrow \gamma$ in F **do**
    if $\beta \subseteq$ result **then**
      result $=$ result $\cup\, \gamma$
    end if
  end for
until result changes no more

Relation R: {[A, B, C, D]} $\alpha = \{A, D\}$

Menge an FDs: $F = \{A \rightarrow B, A \rightarrow C, CD \rightarrow A\}$

$\alpha^+ = result = \{$

$\}$

Beispiel: Ausführung von attrClosure

result $= \alpha$
repeat
  for each $\beta \rightarrow \gamma$ in F **do**
    if $\beta \subseteq$ result **then**
      result $=$ result $\cup\, \gamma$
    end if
  end for
until result changes no more

Relation R: {[A, B, C, D]} $\alpha = \{A, D\}$

Menge an FDs: $F = \{A \rightarrow B, A \rightarrow C, CD \rightarrow A\}$

$\alpha^+ = result = \{$

A, D

$\}$

Beispiel: Ausführung von attrClosure

result $= \alpha$
repeat
  for each $\beta \rightarrow \gamma$ in F **do**
    if $\beta \subseteq$ result **then**
      result $=$ result $\cup\, \gamma$
    end if
  end for
until result changes no more

Relation R: {[A, B, C, D]} $\alpha = \{A, D\}$

Menge an FDs: $F = \{A \rightarrow B, A \rightarrow C, CD \rightarrow A\}$

$\alpha^+ = result = \{$

A, D, B, da A $\rightarrow B$

$\}$

Beispiel: Ausführung von attrClosure

result $= \alpha$
repeat
  for each $\beta \rightarrow \gamma$ in F **do**
    if $\beta \subseteq$ result **then**
      result $=$ result $\cup\, \gamma$
    end if
  end for
until result changes no more

Relation R: {[A, B, C, D]} $\alpha = \{A, D\}$

Menge an FDs: $F = \{A \rightarrow B, A \rightarrow C, CD \rightarrow A\}$

$\alpha^+ = result = \{$

A, D, B, C, da A $\rightarrow C$

$\}$

Beispiel: Ausführung von attrClosure

Relation R: {[A, B, C, D]} $\alpha = \{A, D\}$

Menge an FDs: $F = \{A \rightarrow B, A \rightarrow C, CD \rightarrow A\}$

$\alpha^+ = result = \{$

A, D, B, C, da A $\rightarrow C$

$\}$

Is { A, D } a super key? ja

Is { A } a super key? nein

Is { D } a super key? nein

Äquivalente FD-Mengen

- Zwei Mengen von FDs F und G werden als äquivalent angesehen ($F \equiv G$), wenn deren Hüllen gleich sind, d.h., $F^+ = G^+$.
- Beide Mengen erlauben die gleiche Menge von FDs.

Äquivalente FD-Mengen

- Zwei Mengen von FDs F und G werden als äquivalent angesehen ($F \equiv G$), wenn deren Hüllen gleich sind, d.h., $F^+ = G^+$.
- Beide Mengen erlauben die gleiche Menge von FDs.

Beobachtung

- F^+ kann riesig sein
- Viele redundante Abhängigkeiten
- In der Praxis unübersichtlich

Ziel

Finde die kleinstmögliche Menge F_c für F, so dass $F_c^+ = F^+$.
Es gibt möglicherweise **mehrere minimale Mengen**!

Kanonische Überdeckung F_c (minimale Überdeckung)

Eine **minimale Überdeckung** F_c ist eine kanonische Darstellung einer Menge F von funktionale Abhängigkeiten.

Eigenschaften

Kanonische Überdeckung F_c (minimale Überdeckung)

Eine **minimale Überdeckung** F_c ist eine kanonische Darstellung einer Menge F von funktionale Abhängigkeiten.

Eigenschaften

1. $F_c \equiv F$, also $F_c^+ = F^+$ (Äquivalenz, falls die Hüllen gleich sind)

Kanonische Überdeckung F_c (minimale Überdeckung)

Eine **minimale Überdeckung** F_c ist eine kanonische Darstellung einer Menge F von funktionale Abhängigkeiten.

Eigenschaften

1. $F_c \equiv F$, also $F_c^+ = F^+$ (Äquivalenz, falls die Hüllen gleich sind)
2. In F_c existieren keine FDs $\alpha \rightarrow \beta$, bei denen α oder β **überflüssige Attribute** enthalten. D.h. es muss gelten:
   (a) $\forall A \in \alpha : (F_c - (\alpha \rightarrow \beta) \cup ((\alpha - A) \rightarrow \beta)) \not\equiv F_c$
   (b) $\forall B \in \beta : (F_c - (\alpha \rightarrow \beta) \cup (\alpha \rightarrow (\beta - B))) \not\equiv F_c$

Kanonische Überdeckung F_c (minimale Überdeckung)

Eine **minimale Überdeckung** F_c ist eine kanonische Darstellung einer Menge F von funktionale Abhängigkeiten.

Eigenschaften

1. $F_c \equiv F$, also $F_c^+ = F^+$ (Äquivalenz, falls die Hüllen gleich sind)
2. In F_c existieren keine FDs $\alpha \rightarrow \beta$, bei denen α oder β **überflüssige Attribute** enthalten. D.h. es muss gelten:
   (a) $\forall A \in \alpha : (F_c - (\alpha \rightarrow \beta) \cup ((\alpha - A) \rightarrow \beta)) \not\equiv F_c$
   (b) $\forall B \in \beta : (F_c - (\alpha \rightarrow \beta) \cup (\alpha \rightarrow (\beta - B))) \not\equiv F_c$
3. Jede linke Seite einer funktionalen Abhängigkeit in F_c ist einzigartig. Durch Anwendung der Vereinigungsregel wird ($\alpha \rightarrow \beta$ und $\alpha \rightarrow \gamma$) durch $\alpha \rightarrow \beta\gamma$ ersetzt.

Kanonische Überdeckung F_c (minimale Überdeckung)

Eine **minimale Überdeckung** F_c ist eine kanonische Darstellung einer Menge F von funktionale Abhängigkeiten.

Eigenschaften

1. $F_c \equiv F$, also $F_c^+ = F^+$ (Äquivalenz, falls die Hüllen gleich sind)
2. In F_c existieren keine FDs $\alpha \rightarrow \beta$, bei denen α oder β **überflüssige Attribute** enthalten. D.h. es muss gelten:
   (a) $\forall A \in \alpha : (F_c - (\alpha \rightarrow \beta) \cup ((\alpha - A) \rightarrow \beta)) \not\equiv F_c$
   (b) $\forall B \in \beta : (F_c - (\alpha \rightarrow \beta) \cup (\alpha \rightarrow (\beta - B))) \not\equiv F_c$
3. Jede linke Seite einer funktionalen Abhängigkeit in F_c ist einzigartig. Durch Anwendung der Vereinigungsregel wird ($\alpha \rightarrow \beta$ und $\alpha \rightarrow \gamma$) durch $\alpha \rightarrow \beta\gamma$ ersetzt.

Wie können wir effizient überprüfen, ob ein Attribut überflüssig ist?

Kanonische Überdeckung F_c (minimale Überdeckung)

Eine **minimale Überdeckung** F_c ist eine kanonische Darstellung einer Menge F von funktionale Abhängigkeiten.

Eigenschaften

1. $F_c \equiv F$, also $F_c^+ = F^+$ (Äquivalenz, falls die Hüllen gleich sind)
2. In F_c existieren keine FDs $\alpha \rightarrow \beta$, bei denen α oder β **überflüssige Attribute** enthalten. D.h. es muss gelten:
   (a) $\forall A \in \alpha : (F_c - (\alpha \rightarrow \beta) \cup ((\alpha - A) \rightarrow \beta)) \not\equiv F_c$
   (b) $\forall B \in \beta : (F_c - (\alpha \rightarrow \beta) \cup (\alpha \rightarrow (\beta - B))) \not\equiv F_c$
3. Jede linke Seite einer funktionalen Abhängigkeit in F_c ist einzigartig. Durch Anwendung der Vereinigungsregel wird ($\alpha \rightarrow \beta$ und $\alpha \rightarrow \gamma$) durch $\alpha \rightarrow \beta\gamma$ ersetzt.

Überprüfe mithilfe der Attributhülle, ob $A \in \alpha$ in $\alpha \rightarrow \beta$ überflüssig ist:
(a) $A \in \alpha$ ist überflüssig, wenn $\beta \subseteq \text{attrClosure}(F, \alpha - A)$

Kanonische Überdeckung F_c (minimale Überdeckung)

Eine **minimale Überdeckung** F_c ist eine kanonische Darstellung einer Menge F von funktionale Abhängigkeiten.

Eigenschaften

1. $F_c \equiv F$, also $F_c^+ = F^+$ (Äquivalenz, falls die Hüllen gleich sind)
2. In F_c existieren keine FDs $\alpha \rightarrow \beta$, bei denen α oder β **überflüssige Attribute** enthalten. D.h. es muss gelten:
   (a) $\forall A \in \alpha : (F_c - (\alpha \rightarrow \beta) \cup ((\alpha - A) \rightarrow \beta)) \not\equiv F_c$
   (b) $\forall B \in \beta : (F_c - (\alpha \rightarrow \beta) \cup (\alpha \rightarrow (\beta - B))) \not\equiv F_c$
3. Jede linke Seite einer funktionalen Abhängigkeit in F_c ist einzigartig. Durch Anwendung der Vereinigungsregel wird ($\alpha \rightarrow \beta$ und $\alpha \rightarrow \gamma$) durch $\alpha \rightarrow \beta\gamma$ ersetzt.

Überprüfe mithilfe der Attributhülle, ob $B \in \beta$ in $\alpha \rightarrow \beta$ überflüssig ist:
(b) $B \in \beta$ ist überflüssig, wenn
$B \in \text{attrClosure}((F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - B)\}, \alpha)$

Algorithmus minimale Überdeckung

Hauptschritte:

1. Führe FDs mit der gleichen linken Seite zusammen:
   $\alpha \rightarrow \beta_1, ..., \alpha \rightarrow \beta_n$ zu $\alpha \rightarrow (\beta_1 \cup ...\beta_n)$

Algorithmus minimale Überdeckung

Hauptschritte:

1. Führe FDs mit der gleichen linken Seite zusammen:
   $\alpha \rightarrow \beta_1, ..., \alpha \rightarrow \beta_n$ zu $\alpha \rightarrow (\beta_1 \cup ...\beta_n)$
2. Für jede FD $(\alpha \rightarrow \beta) \in F$ führe **Linksreduktion** durch:
   Überprüfe für jedes $A \in \alpha$, ob A überflüssig ist, d.h., ob
   $\beta \subseteq \text{attrClosure}(F, \alpha - A)$
   Wenn ja, entferne A, indem $\alpha \rightarrow \beta$ durch $(\alpha - A) \rightarrow \beta$ ersetzt wird.

Algorithmus minimale Überdeckung

Hauptschritte:

1. Führe FDs mit der gleichen linken Seite zusammen:
   $\alpha \to \beta_1, ..., \alpha \to \beta_n$ zu $\alpha \to (\beta_1 \cup ...\beta_n)$
2. Für jede FD $(\alpha \to \beta) \in F$ führe **Linksreduktion** durch:
   Überprüfe für jedes $A \in \alpha$, ob A überflüssig ist, d.h., ob
   $\beta \subseteq \text{attrClosure}(F, \alpha - A)$
   Wenn ja, entferne A, indem $\alpha \to \beta$ durch $(\alpha - A) \to \beta$ ersetzt wird.
3. Für jede FD $(\alpha \to \beta) \in F$ führe **Rechtsreduktion** durch:
   Überprüfe für jedes $B \in \beta$, ob B überflüssig ist, d.h., ob
   $B \in \text{attrClosure}((F - \{\alpha \to \beta\}) \cup \{\alpha \to (\beta - B)\}, \alpha)$
   Wenn ja, entferne B, indem $\alpha \to \beta$ durch $\alpha \to (\beta - B)$ ersetzt wird.

Algorithmus minimale Überdeckung

Hauptschritte:

1. Führe FDs mit der gleichen linken Seite zusammen:
   $\alpha \to \beta_1, ..., \alpha \to \beta_n$ zu $\alpha \to (\beta_1 \cup ...\beta_n)$
2. Für jede FD $(\alpha \to \beta) \in F$ führe **Linksreduktion** durch:
   Überprüfe für jedes $A \in \alpha$, ob A überflüssig ist, d.h., ob
   $\beta \subseteq \text{attrClosure}(F, \alpha - A)$
   Wenn ja, entferne A, indem $\alpha \to \beta$ durch $(\alpha - A) \to \beta$ ersetzt wird.
3. Für jede FD $(\alpha \to \beta) \in F$ führe **Rechtsreduktion** durch:
   Überprüfe für jedes $B \in \beta$, ob B überflüssig ist, d.h., ob
   $B \in \text{attrClosure}((F - \{\alpha \to \beta\}) \cup \{\alpha \to (\beta - B)\}, \alpha)$
   Wenn ja, entferne B, indem $\alpha \to \beta$ durch $\alpha \to (\beta - B)$ ersetzt wird.
4. Entferne FDs der Form $\alpha \to \emptyset$ (sind möglicherweise in Schritt 3 entstanden).

Algorithmus minimale Überdeckung

Hauptschritte:

1. Führe FDs mit der gleichen linken Seite zusammen:
   $\alpha \to \beta_1, ..., \alpha \to \beta_n$ zu $\alpha \to (\beta_1 \cup ...\beta_n)$
2. Für jede FD $(\alpha \to \beta) \in F$ führe **Linksreduktion** durch:
   Überprüfe für jedes $A \in \alpha$, ob A überflüssig ist, d.h., ob
   $\beta \subseteq \text{attrClosure}(F, \alpha - A)$
   Wenn ja, entferne A, indem $\alpha \to \beta$ durch $(\alpha - A) \to \beta$ ersetzt wird.
3. Für jede FD $(\alpha \to \beta) \in F$ führe **Rechtsreduktion** durch:
   Überprüfe für jedes $B \in \beta$, ob B überflüssig ist, d.h., ob
   $B \in \text{attrClosure}((F - \{\alpha \to \beta\}) \cup \{\alpha \to (\beta - B)\}, \alpha)$
   Wenn ja, entferne B, indem $\alpha \to \beta$ durch $\alpha \to (\beta - B)$ ersetzt wird.
4. Entferne FDs der Form $\alpha \to \emptyset$ (sind möglicherweise in Schritt 3 entstanden).
5. Gehe zu Schritt 1
   Im Allgemeinen ist eine Runde plus ein weiteres mal Schritt 1 genug.

Beispiel

Initiale Menge von Abhängigkeiten:
$F = \{A \rightarrow B, B \rightarrow C, AB \rightarrow C\}$

Momentane Menge von Abhängigkeiten:
$F_c = \{A \rightarrow B, B \rightarrow C, AB \rightarrow C\}$

Beispiel

Initiale Menge von Abhängigkeiten:
$F = \{A \rightarrow B, B \rightarrow C, AB \rightarrow C\}$

Momentane Menge von Abhängigkeiten:
$F_c = \{A \rightarrow B, B \rightarrow C, AB \rightarrow C\}$

- Schritt 1: Einzigartige linke Seiten
  Keine Zusammenführung in diesem Beispiel notwendig

Beispiel

Initiale Menge von Abhängigkeiten:
$F = \{A \to B, B \to C, AB \to C\}$

Momentane Menge von Abhängigkeiten:
$F_c = \{A \to B, B \to C, A \to C\}$

- Schritt 1: Einzigartige linke Seiten
  Keine Zusammenführung in diesem Beispiel notwendig
- Schritt 2: Linksreduktion
  Ersetzte $AB \to C$ durch $A \to C$

Beispiel

Initiale Menge von Abhängigkeiten:
$F = \{A \to B, B \to C, AB \to C\}$

Momentane Menge von Abhängigkeiten:
$F_c = \{A \to B, B \to C, A \to \emptyset\}$

- Schritt 1: Einzigartige linke Seiten
  Keine Zusammenführung in diesem Beispiel notwendig
- Schritt 2: Linksreduktion
  Ersetzte $AB \to C$ durch $A \to C$
- Schritt 3: Rechtsreduktion
  Ersetzte $A \to C$ durch $A \to \emptyset$

Beispiel

Initiale Menge von Abhängigkeiten:
$F = \{A \to B, B \to C, AB \to C\}$

Momentane Menge von Abhängigkeiten:
$F_c = \{A \to B, B \to C\}$

- Schritt 1: Einzigartige linke Seiten
  Keine Zusammenführung in diesem Beispiel notwendig
- Schritt 2: Linksreduktion
  Ersetzte $AB \to C$ durch $A \to C$
- Schritt 3: Rechtsreduktion
  Ersetzte $A \to C$ durch $A \to \emptyset$
- Schritt 4: Entferne Regeln der Form $\alpha \to \emptyset$
  Entferne $A \to \emptyset$

Relationen normalisieren

Normalisierung durch Zerlegung

Zerlege ein Relationenschema $\mathcal{R}$ in mehrere Relationenschemata $\mathcal{R}_1, \ldots, \mathcal{R}_n$, um Probleme des originalen Entwurfs zu beheben.

Normalformen

- **Normalformen** beschreiben die Güte des Entwurfs
- 1NF, 2NF, 3NF, BCNF, 4NF, ...
- Sie verbieten bestimmte funktionale Abhängigkeiten in einer Relation um **Redundanz, Null-Werte und Anomalien** zu verhindern.

Relationen normalisieren

Normalisierung durch Zerlegung

Zerlege ein Relationenschema $\mathcal{R}$ in mehrere Relationenschemata $\mathcal{R}_1, \ldots, \mathcal{R}_n$, um Probleme des originalen Entwurfs zu beheben.

Normalformen

- **Normalformen** beschreiben die Güte des Entwurfs
- 1NF, 2NF, 3NF, BCNF, 4NF, ...
- Sie verbieten bestimmte funktionale Abhängigkeiten in einer Relation um **Redundanz, Null-Werte und Anomalien** zu verhindern.

- Gute ER-Modelle führen typischerweise direkt zur 3NF (oder höhere NF)
- Normalisierung verhindert Probleme, welche durch funktionalen Abhängigkeiten zwischen den Attributen eines Entitytyps ausgelöst werden.

Gültige und verlustlose Zerlegungen

Eine Zerlegung ist **gültig**, falls $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$, d.h., wenn alle Attribute aus $\mathcal{R}$ in der Zerlegung erhalten bleiben.

- $R_1 := \pi_{\mathcal{R}_1}(R)$
- $R_2 := \pi_{\mathcal{R}_2}(R)$

Eine Zerlegung von $\mathcal{R}$ in $\mathcal{R}_1$ und $\mathcal{R}_2$ ist **verlustlos**, wenn Folgendes für alle möglichen Ausprägungen R von $\mathcal{R}$ gilt:

$$R = R_1 \bowtie R_2$$

Die in der ursprünglichen Ausprägung R des Schemas $\mathcal{R}$ enthaltenen Informationen müssen aus den Ausprägungen $R_1, \ldots, R_n$ der neuen Schemata $\mathcal{R}_1, \ldots, \mathcal{R}_n$ durch einen natürlichen Join rekonstruierbar sein.

Formale Charakterisierung verlustloser Zerlegungen

Gegeben

- Eine Zerlegung von $\mathcal{R}$ in $\mathcal{R}_1$ und $\mathcal{R}_2$
- $F_{\mathcal{R}}$ ist die Menge der FDs in $\mathcal{R}$

Eine Zerlegung ist **verlustlos**, wenn mindestens **eine** der folgenden FDs herleitbar ist:

- $(\mathcal{R}_1 \cap \mathcal{R}_2) \rightarrow \mathcal{R}_1 \in F_{\mathcal{R}}^+$ die gemeinsamen Attribute bestimmen $\mathcal{R}_1$
  oder
- $(\mathcal{R}_1 \cap \mathcal{R}_2) \rightarrow \mathcal{R}_2 \in F_{\mathcal{R}}^+$ die gemeinsamen Attribute bestimmen $\mathcal{R}_2$

Beispielzerlegung

beerDrinkers		
pub	guest	beer
Kowalski	Kemper	pils
Kowalski	Eickler	wheat beer
Innsteg	Kemper	wheat beer

zerlegt in

$\pi_{pub,guest}$

customer	
pub	guest
Kowalski	Kemper
Kowalski	Eickler
Innsteg	Kemper

$\pi_{guest,beer}$

drinks	
guest	beer
Kemper	pils
Eickler	wheat beer
Kemper	wheat beer

Verlustlos oder nicht?

Beispielzerlegung

$\pi_{pub,guest}$

customer	
pub	guest
Kowalski	Kemper
Kowalski	Eickler
Innsteg	Kemper

$\pi_{guest,beer}$

drinks	
guest	beer
Kemper	pils
Eickler	wheat beer
Kemper	wheat beer

Rekonstruktion: $R_1 \bowtie R_2$

beerDrinkers		
pub	guest	beer
Kowalski	Kemper	pils
Kowalski	Kemper	wheat beer
Kowalski	Eickler	wheat beer
Innsteg	Kemper	pils
Innsteg	Kemper	wheat beer

Beispielzerlegung

$\pi_{pub,guest}$

customer	
pub	guest
Kowalski	Kemper
Kowalski	Eickler
Innsteg	Kemper

$\pi_{guest,beer}$

drinks	
guest	beer
Kemper	pils
Eickler	wheat beer
Kemper	wheat beer

Rekonstruktion: $R_1 \bowtie R_2$

beerDrinkers		
pub	guest	beer
Kowalski	Kemper	pils
Kowalski	**Kemper**	**wheat beer**
Kowalski	Eickler	wheat beer
Innsteg	**Kemper**	**pils**
Innsteg	Kemper	wheat beer

Beispielzerlegung

Die Beziehung zwischen guest, pub und beer ging verloren.

Die Verletzung der Verlustfreiheit kann manchmal auch bedeuten, dass bei der Wiederherstellung zusätzliche Tupel entstehen.

Das vorherige Beispiel war keine verlustlose Zerlegung.

Abhängigkeitsbewahrung

Zweites Kriterium für eine gute Zerlegung

Die für $\mathcal{R}$ geltenden Abhängigkeiten müssen in den neuen Schemata $\mathcal{R}_1, \ldots, \mathcal{R}_n$ verifizierbar sein.

- Wir können alle Abhängigkeiten lokal in $\mathcal{R}_1, ..., \mathcal{R}_n$ überprüfen.
- Wir vermeiden, dass wir den Join $R_1 \bowtie \cdots \bowtie R_n$ berechnen müssen, um überprüfen zu können, ob eine FD verletzt wird.

Eine Zerlegung bewahrt Abhängigkeiten, wenn

$F_{\mathcal{R}} \equiv (F_{\mathcal{R}_1} \cup \cdots \cup F_{\mathcal{R}_n})$ d.h., $F_{\mathcal{R}}^+ = (F_{\mathcal{R}_1} \cup \cdots \cup F_{\mathcal{R}_n})^+$

$F_{\mathcal{R}_i}$ sind die funktionalen Abhängigkeiten, die sich über R_i effizient überprüfen lassen.

Abhängigkeitsbewahrung

Überprüfen, ob eine Zerlegung Abhängigkeiten bewahrt, ohne F^+ berechnen zu müssen:

1. Können FDs effizient über dem zerlegtem Schema getestet werden?
   - Weise FDs den Relationen zu, welche alle involvierten Attribute enthalten.

Abhängigkeitsbewahrung

Überprüfen, ob eine Zerlegung Abhängigkeiten bewahrt, ohne F^+ berechnen zu müssen:

1. Können FDs effizient über dem zerlegtem Schema getestet werden?
   - Weise FDs den Relationen zu, welche alle involvierten Attribute enthalten.

   Falls ja: die Zerlegung bewahrt Abhängigkeiten

Abhängigkeitsbewahrung

Überprüfen, ob eine Zerlegung Abhängigkeiten bewahrt, ohne F^+ berechnen zu müssen:

1. Können FDs effizient über dem zerlegtem Schema getestet werden?
   - Weise FDs den Relationen zu, welche alle involvierten Attribute enthalten.

   Falls ja: die Zerlegung bewahrt Abhängigkeiten
2. Falls nicht: verwende modifizierte Version von attrClosure um jede $\alpha \rightarrow \beta$ in F zu testen

$result = \alpha$
repeat
  for each R_i in the decomposition **do**
    $t = (result \cap R_i)^+ \cap R_i$
    $result = result \cup t$
  end for
until result changes no more

Abhängigkeitsbewahrung

Überprüfen, ob eine Zerlegung Abhängigkeiten bewahrt, ohne F^+ berechnen zu müssen:

1. Können FDs effizient über dem zerlegtem Schema getestet werden?
   - Weise FDs den Relationen zu, welche alle involvierten Attribute enthalten.

   Falls ja: die Zerlegung bewahrt Abhängigkeiten
2. Falls nicht: verwende modifizierte Version von attrClosure um jede $\alpha \rightarrow \beta$ in F zu testen

   Falls $\beta \subseteq result$, dann ist die FD bewahrt

$result = \alpha$
repeat
  for each R_i in the decomposition **do**
    $t = (result \cap R_i)^+ \cap R_i$
    $result = result \cup t$
  end for
until result changes no more

Abhängigkeitsbewahrung

Überprüfen, ob eine Zerlegung Abhängigkeiten bewahrt, ohne F^+ berechnen zu müssen:

1. Können FDs effizient über dem zerlegtem Schema getestet werden?
   - Weise FDs den Relationen zu, welche alle involvierten Attribute enthalten.

   Falls ja: die Zerlegung bewahrt Abhängigkeiten
2. Falls nicht: verwende modifizierte Version von attrClosure um jede $\alpha \rightarrow \beta$ in F zu testen

   Falls $\beta \subseteq result$, dann ist die FD bewahrt

   Wenn alle FDs bewahrt werden, dann bewahrt die Zerlegung Abhängigkeiten

$result = \alpha$
repeat
  for each R_i in the decomposition **do**
    $t = (result \cap R_i)^+ \cap R_i$
    $result = result \cup t$
  end for
until result changes no more

Beispiel

zipCatalog: {[street, city, region, zip]}

FDs

- {zip} → {city, region}
- {street, city, region} → {zip}

Zerlegung

- streets: {[zip, street]}
- cities: {[zip, city, region]}

Ist diese Zerlegung verlustfrei?
Bewahrt diese Zerlegung Abhängigkeiten?

Beispiel

zipCatalog			
city	region	street	zip
Frankfurt	Hesse	Goethestraße	60313
Frankfurt	Hesse	Galgenstraße	60437
Frankfurt	Brandenburg	Goethestraße	15234

$\pi_{zip,street}$

street	
zip	street
15234	Goethestraße
60313	Goethestraße
60437	Galgenstraße

$\pi_{city,region,zip}$

cities		
city	region	zip
Frankfurt	Hesse	60313
Frankfurt	Hesse	60437
Frankfurt	Brandenburg	15234

Verlustfrei, da $\{\text{zip}\} \rightarrow \{\text{city, region}\}$

Beispiel

zipCatalog			
city	region	street	zip
Frankfurt	Hesse	Goethestraße	60313
Frankfurt	Hesse	Galgenstraße	60437
Frankfurt	Brandenburg	Goethestraße	15234

$\pi_{zip,street}$

street	
zip	street
15234	Goethestraße
60313	Goethestraße
60437	Galgenstraße

$\pi_{city,region,zip}$

cities		
city	region	zip
Frankfurt	Hesse	60313
Frankfurt	Hesse	60437
Frankfurt	Brandenburg	15234

Die Zerlegung bewahrt die Abhängigkeiten **nicht**:
{street, city, region} $\rightarrow$ {zip} kann nicht über der Zerlegung überprüft werden und wird nicht durch andere FDs garantiert.

Beispiel 1

Gegeben

- $R = (A, B, C)$
- $F = \{A \rightarrow B, B \rightarrow C\}$
- $R_1 = (A, B),\ R_2 = (B, C)$

Ist das eine verlustfreie Zerlegung?

Beispiel 1

Gegeben

- $R = (A, B, C)$
- $F = \{A \rightarrow B, B \rightarrow C\}$
- $R_1 = (A, B),\ R_2 = (B, C)$

Ist das eine verlustfreie Zerlegung?

Hinweis:

- $(\mathcal{R}_1 \cap \mathcal{R}_2) \rightarrow \mathcal{R}_1 \in F^+_{\mathcal{R}}$
- $(\mathcal{R}_1 \cap \mathcal{R}_2) \rightarrow \mathcal{R}_2 \in F^+_{\mathcal{R}}$

Beispiel 1

Gegeben

- $R = (A, B, C)$
- $F = \{A \to B, B \to C\}$
- $R_1 = (A, B),\ R_2 = (B, C)$

Ist das eine verlustfreie Zerlegung?

Hinweis (gemeinsame Attribute sind Superschlüssel in einer der Relationen):

- $(\mathcal{R}_1 \cap \mathcal{R}_2) \to \mathcal{R}_1 \in F^+_{\mathcal{R}}$
- $(\mathcal{R}_1 \cap \mathcal{R}_2) \to \mathcal{R}_2 \in F^+_{\mathcal{R}}$

Beispiel 1

Gegeben

- $R = (A, B, C)$
- $F = \{A \to B, B \to C\}$
- $R_1 = (A, B),\ R_2 = (B, C)$

Ist das eine verlustfreie Zerlegung?

Hinweis (gemeinsame Attribute sind Superschlüssel in einer der Relationen):

- $(\mathcal{R}_1 \cap \mathcal{R}_2) \to \mathcal{R}_1 \in F^+_{\mathcal{R}}$
- $(\mathcal{R}_1 \cap \mathcal{R}_2) \to \mathcal{R}_2 \in F^+_{\mathcal{R}}$

JA:

- $R_1 \cap R_2 = \{B\}$ und $B \to BC$

Beispiel 1

Gegeben

- $R = (A, B, C)$
- $F = \{A \rightarrow B, B \rightarrow C\}$
- $R_1 = (A, B),\ R_2 = (B, C)$

Bewahrt diese Zerlegung Abhängigkeiten?

Beispiel 1

Gegeben

- $R = (A, B, C)$
- $F = \{A \to B, B \to C\}$
- $R_1 = (A, B),\ R_2 = (B, C)$

Bewahrt diese Zerlegung Abhängigkeiten?

Hinweis:
$(F_1 \cup F_2)^+ = F^+$

Beispiel 1

Gegeben

- $R = (A, B, C)$
- $F = \{A \to B, B \to C\}$
- $R_1 = (A, B),\ R_2 = (B, C)$

Bewahrt diese Zerlegung Abhängigkeiten?
Hinweis (wir können alle Abhängigkeiten lokal überprüfen):
$(F_1 \cup F_2)^+ = F^+$

Beispiel 1

Gegeben

- $R = (A, B, C)$
- $F = \{A \to B, B \to C\}$
- $R_1 = (A, B),\ R_2 = (B, C)$

Bewahrt diese Zerlegung Abhängigkeiten?

Hinweis (wir können alle Abhängigkeiten lokal überprüfen):
$(F_1 \cup F_2)^+ = F^+$

JA:

- $F_1 = \{A \to B, A \to AB\} \cup \{\text{viele triviale Abhängigkeiten}\}$
- $F_2 = \{B \to C, B \to BC\} \cup \{\text{viele triviale Abhängigkeiten}\}$
- $(F_1 \cup F_2)^+ = F^+$

Beispiel 2

Gegeben

- $R = (A, B, C)$
- $F = \{A \rightarrow B, B \rightarrow C\}$
- $R_1 = (A, B),\ R_2 = (\mathbf{A}, C)$

Ist das eine verlustfreie Zerlegung?

Beispiel 2

Gegeben

- $R = (A, B, C)$
- $F = \{A \rightarrow B, B \rightarrow C\}$
- $R_1 = (A, B),\ R_2 = (\mathbf{A}, C)$

Ist das eine verlustfreie Zerlegung?

Hinweis (gemeinsame Attribute sind Superschlüssel in einer der Relationen):

- $(\mathcal{R}_1 \cap \mathcal{R}_2) \rightarrow \mathcal{R}_1 \in F^+_{\mathcal{R}}$
- $(\mathcal{R}_1 \cap \mathcal{R}_2) \rightarrow \mathcal{R}_2 \in F^+_{\mathcal{R}}$

Beispiel 2

Gegeben

- $R = (A, B, C)$
- $F = \{A \to B, B \to C\}$
- $R_1 = (A, B),\ R_2 = (\mathbf{A}, C)$

Ist das eine verlustfreie Zerlegung?

Hinweis (gemeinsame Attribute sind Superschlüssel in einer der Relationen):

- $(\mathcal{R}_1 \cap \mathcal{R}_2) \to \mathcal{R}_1 \in F^+_{\mathcal{R}}$
- $(\mathcal{R}_1 \cap \mathcal{R}_2) \to \mathcal{R}_2 \in F^+_{\mathcal{R}}$

JA:

- $R_1 \cap R_2 = \{A\}$ und $A \to AB$

Beispiel 2

Gegeben

- $R = (A, B, C)$
- $F = \{A \rightarrow B, B \rightarrow C\}$
- $R_1 = (A, B),\ R_2 = (\mathbf{A}, C)$

Bewahrt diese Zerlegung Abhängigkeiten?

Beispiel 2

Gegeben

- $R = (A, B, C)$
- $F = \{A \to B, B \to C\}$
- $R_1 = (A, B),\ R_2 = (\mathbf{A}, C)$

Bewahrt diese Zerlegung Abhängigkeiten?
Hinweis (wir können alle Abhängigkeiten lokal überprüfen):
$(F_1 \cup F_2)^+ = F^+$

Beispiel 2

Gegeben

- $R = (A, B, C)$
- $F = \{A \to B, B \to C\}$
- $R_1 = (A, B),\ R_2 = (\mathbf{A}, C)$

Bewahrt diese Zerlegung Abhängigkeiten?

Hinweis (wir können alle Abhängigkeiten lokal überprüfen):
$(F_1 \cup F_2)^+ = F^+$

NEIN:

- $F_1 = \{A \to B, A \to AB\} \cup \{\text{viele triviale Abhängigkeiten}\}$
- $F_2 = \{\text{viele triviale Abhängigkeiten}\}$
- $(F_1 \cup F_2)^+ \neq F^+$
- $B \to C$ kann nicht überprüft werden ohne $R_1 \bowtie R_2$ zu berechnen

Zusammenfassung funktionaler Abhängigkeiten

Eigenschaften eines guten Datenbankentwurfs

- Keine Redundanz
- Keine Änderungsanomalien
- Alle Informationen können dargestellt werden

Warum Zerlegung?

- Verwandelt einen schlechten Entwurf in einen guten
  - Zerlege große Relationenschemata in mehrere kleinere

Eine gute Zerlegung

- ist verlustlos
- bewahrt Abhängigkeiten

Normalformen

Normalformen

- legen Eigenschaften von Relationenschemata fest
- verbieten bestimmte Kombinationen von funktionalen Abhängigkeiten in Relationen
- sollen Redundanzen und Anomalien vermeiden
- Richtlinie, um gute Zerlegungen zu erhalten

1NF, 2NF, 3NF, BCNF, 4NF, ...

Erste Normalform (1NF)

Eine Relation $\mathcal{R}$ ist in 1NF, wenn alle Attribute atomar sind (keine zusammengesetzten, mengenwertige oder relationenwertige Domänen).

Nicht 1NF:

parents		
father	**mother**	**child**
Johann	Martha	{**Else, Lucie**}
Johann	Maria	{**Theo, Josef**}
Heinz	Martha	{**Cleo**}

1NF:

parents		
father	**mother**	**child**
Johann	Martha	Else
Johann	Martha	Lucie
Johann	Maria	Theo
Johann	Maria	Josef
Heinz	Martha	Cleo

Zweite Normalform (2NF)

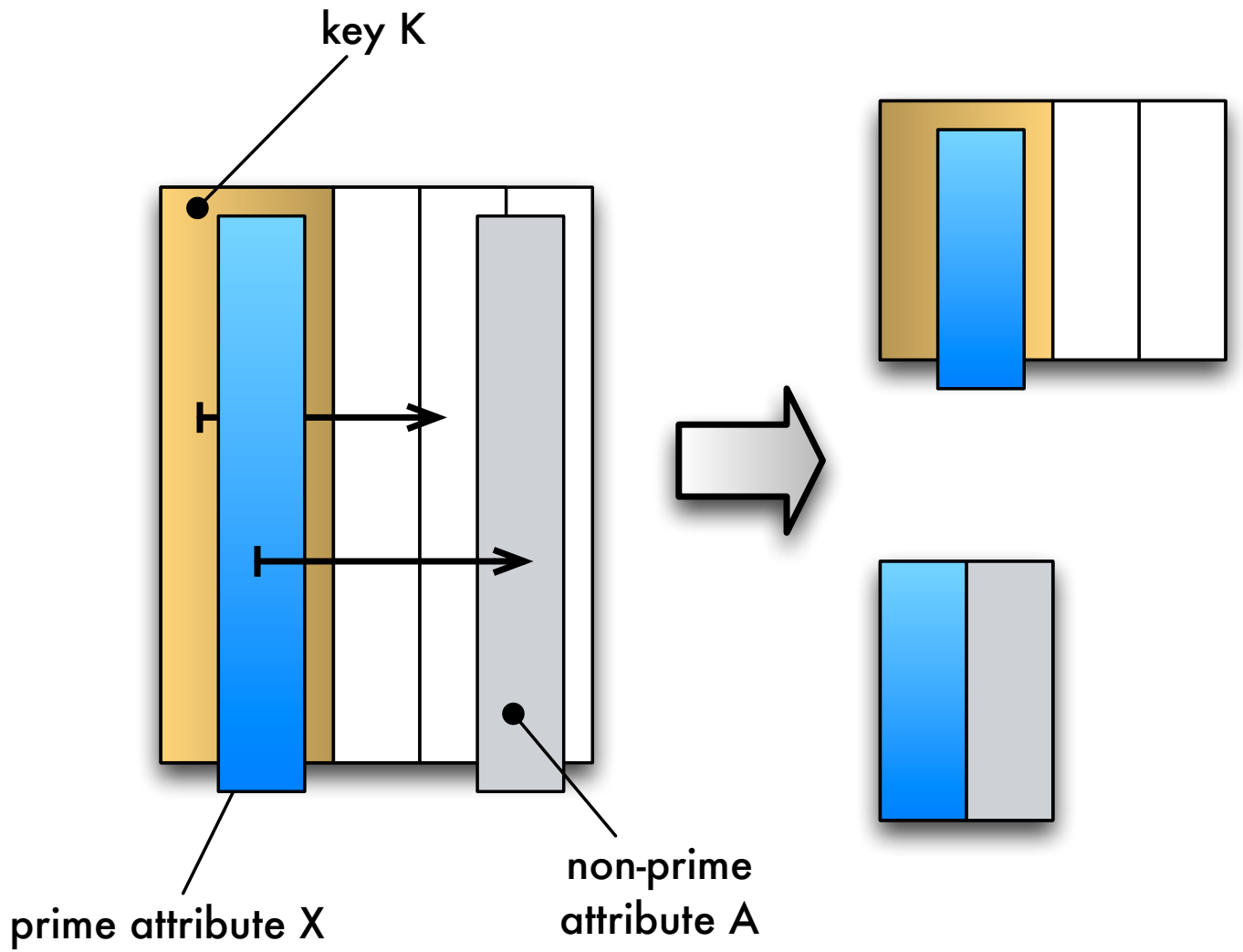
Ein Relationschema $\mathcal{R}$ mit FDs F ist in 2NF, falls jedes Nicht-Primattribut $A \in \mathcal{R}$ voll funktional abhängig von *jedem* Kandidatenschlüssel der Relation ist.

$(\kappa_j \rightarrow A) \in F^+$ und κ_j ist linksreduziert (**voll** funktional abhängig)

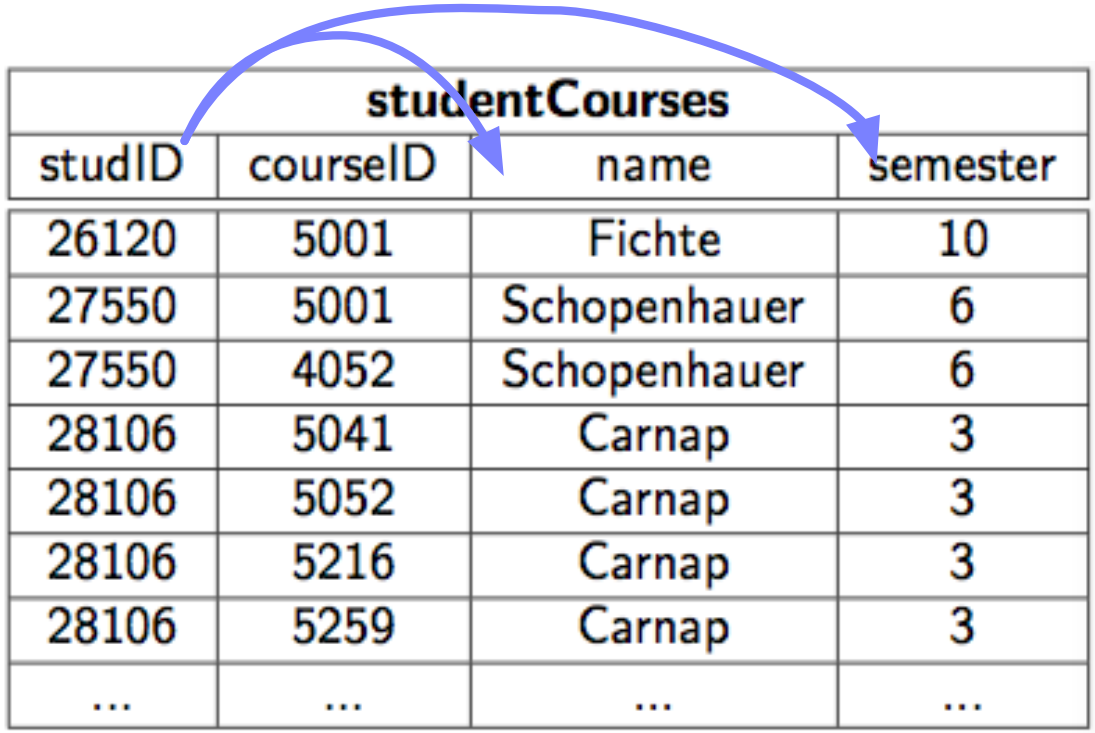
- Ein Attribut A ist **prim**, wenn $A \in (\kappa_1 \cup ... \cup \kappa_i)$
- Ein Attribut A ist **nicht prim**, wenn $A \in \mathcal{R} - (\kappa_1 \cup ... \cup \kappa_i)$

- 2NF verhindert partielle Abhängigkeiten
- Es gibt keine funktionale Abhängigkeit von einem Nicht-Prim-Attribut auf nur eine Teilmenge von Schlüsselattributen.

Eliminierung partieller Abhängigkeiten



Eliminierung partieller Abhängigkeiten



studentCourses			
studID	courseID	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
...	...	...	...

Zerlegung in zwei Relationen:

- takes: {[studID, courseID]}
- student: {[studID, name, semester]}

Beide Relationen sind in 2NF.

Beispiel

studentCourses			
studID	courseID	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
...	...	...	...

- Kandidatenschlüssel: $\{ \{\text{studID}, \text{courseID}\} \}$
- Schlüsselattribute (prim): $\{\text{studID}, \text{courseID}\}$
- Nicht-Schlüsselattribute (nicht prim): $\{\text{name}, \text{semester}\}$
- FDs: $\{\text{studID}\} \rightarrow \{\text{name}\}$ und $\{\text{studID}\} \rightarrow \{\text{semester}\}$

Beispiel

studentCourses			
studID	courseID	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
...	...	...	...

- Kandidatenschlüssel: $\{ \{\text{studID}, \text{courseID}\} \}$
- Schlüsselattribute (prim): $\{\text{studID}, \text{courseID}\}$
- Nicht-Schlüsselattribute (nicht prim): $\{\text{name}, \text{semester}\}$
- FDs: $\{\text{studID}\} \rightarrow \{\text{name}\}$ und $\{\text{studID}\} \rightarrow \{\text{semester}\}$

Ist diese Relation in 2NF?

Beispiel

studentCourses			
studID	courseID	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
...	...	...	...

- Kandidatenschlüssel: $\{ \{\text{studID}, \text{courseID}\} \}$
- Schlüsselattribute (prim): $\{\text{studID}, \text{courseID}\}$
- Nicht-Schlüsselattribute (nicht prim): $\{\text{name}, \text{semester}\}$
- FDs: $\{\text{studID}\} \rightarrow \{\text{name}\}$ und $\{\text{studID}\} \rightarrow \{\text{semester}\}$

Partielle Abhängigkeit: studentCourses ist nicht in 2NF

Dritte Normalform (3NF)

Ein Relationenschema $\mathcal{R}$ ist in dritter Normalform, wenn für jede für $\mathcal{R}$ geltende FD der Form $\alpha \rightarrow B$ mit Attribut $B \in \mathcal{R}$ mindestens eine von drei Bedingungen gilt:

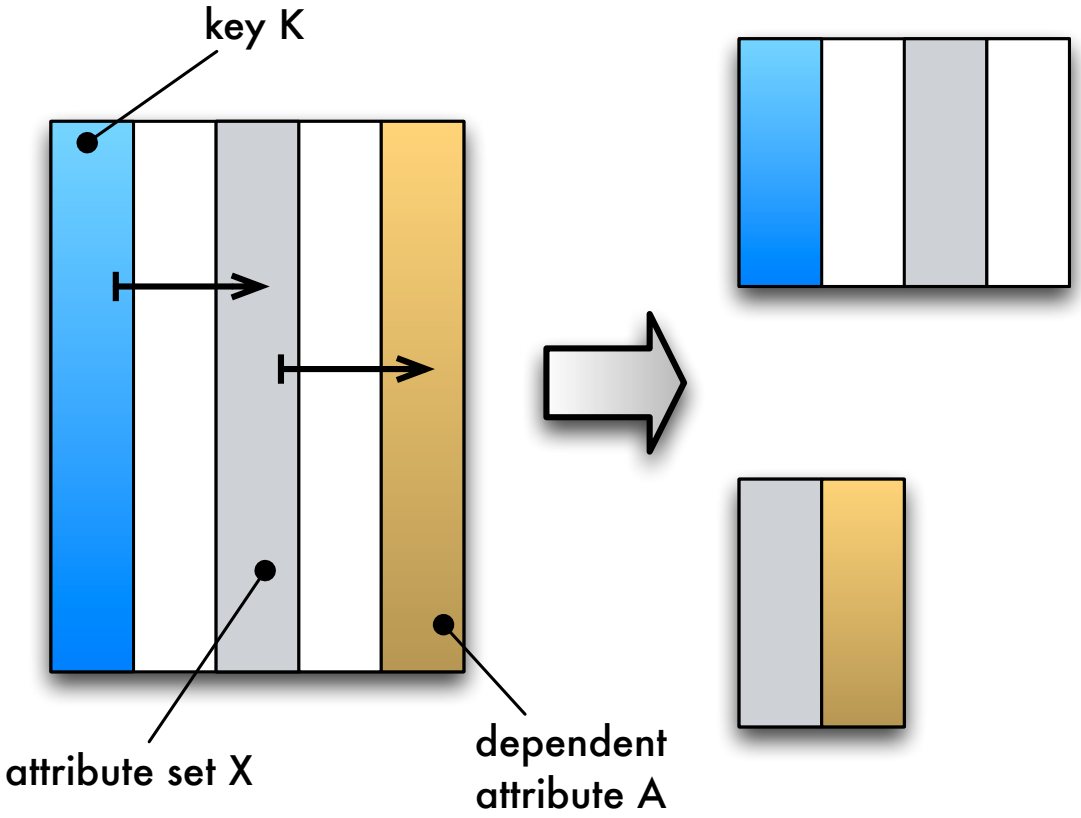
1. $B \in \alpha$, d.h. die FD ist **trivial**
2. α is ein **Superschlüssel** von R
3. B ist **prim** (Schlüsselattribut)

Eigenschaften

- Nicht-Prim-Attribute dürfen keine anderen Attribut bestimmen.
- 3NF verhindert **partielle und transitive** Abhängigkeiten.
- Ausnahme (Fall 3): transitive Abhängigkeit mit Prim-Attributen als Endpunkte sind okay!
- 3NF „beinhaltet“ 2NF

Dritte Normalform (3NF)

Die 3NF verhindert **transitive Abhängigkeiten** und beinhaltet 2NF.



Nur Prim-Attribute können Endpunkte von transitiven Abhängigkeiten sein!

Beispiel

studentCourses			
studID	courseID	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
...	...	...	...

- Kandidatenschlüssel: $\{ \{\text{studID}, \text{courseID}\} \}$
- FDs: $\{\text{studID}\} \rightarrow \{\text{name}\}$, $\{\text{studID}\} \rightarrow \{\text{semester}\}$

Ist die Relation in 3NF?

Beispiel

studentCourses			
studID	courseID	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
...	...	...	...

- Kandidatenschlüssel: { {studID, courseID} }
- FDs: {studID} $\rightarrow$ {name}, {studID} $\rightarrow$ {semester}

Hinweise (FD $\alpha \rightarrow B$)

1. $B \in \alpha$, i.e., die FD ist trivial
2. α ist ein Superschlüssel von R
3. B ist prim

Beispiel

studentCourses			
studID	courseID	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
...	...	...	...

- Kandidatenschlüssel: { {studID, courseID} }
- FDs: {studID} → {name}, {studID} → {semester}

Hinweise (FD $\alpha \rightarrow B$)

1. $B \in \alpha$, i.e., die FD ist trivial – beide FDs sind nicht trivial
2. α ist ein Superschlüssel von R
3. B ist prim

Beispiel

studentCourses			
studID	courseID	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
...	...	...	...

- Kandidatenschlüssel: $\{ \{\text{studID}, \text{courseID}\} \}$
- FDs: $\{\text{studID}\} \rightarrow \{\text{name}\}$, $\{\text{studID}\} \rightarrow \{\text{semester}\}$

Hinweise (FD $\alpha \rightarrow B$)

1. $B \in \alpha$, i.e., die FD ist trivial – beide FDs sind nicht trivial
2. α ist ein Superschlüssel von R – $\{\text{studID}\}$ ist kein Superschlüssel
3. B ist prim

Beispiel

studentCourses			
studID	courseID	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
...	...	...	...

- Kandidatenschlüssel: $\{ \{\text{studID}, \text{courseID}\} \}$
- FDs: $\{\text{studID}\} \rightarrow \{\text{name}\}$, $\{\text{studID}\} \rightarrow \{\text{semester}\}$

Hinweise (FD $\alpha \rightarrow B$)

1. $B \in \alpha$, i.e., die FD ist trivial – beide FDs sind nicht trivial
2. α ist ein Superschlüssel von R – $\{\text{studID}\}$ ist kein Superschlüssel
3. B ist prim – name und semester sind nicht prim

Beispiel

studentCourses			
studID	courseID	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
...	...	...	...

- Kandidatenschlüssel: $\{ \{\text{studID}, \text{courseID}\} \}$
- FDs: $\{\text{studID}\} \rightarrow \{\text{name}\}$, $\{\text{studID}\} \rightarrow \{\text{semester}\}$

Die Relation ist nicht in 3NF.

Boyce-Codd-Normalform (BCNF)

Ein Relationenschema $\mathcal{R}$ ist in BCNF, wenn für jede für $\mathcal{R}$ geltende FD der Form $\alpha \rightarrow B$ mit Attribut $B \in \mathcal{R}$ mindestens eine von zwei Bedingungen gilt:

1. $B \in \alpha$, d.h. die FD ist **trivial**
2. α ist **Superschlüssel** von $\mathcal{R}$

Eigenschaften

- Unterschied zu 3NF: Die dritte Bedingung fällt weg (B ist prim)
- Die Boyce-Codd-Normalform ist eine weitere Verschärfung der 3NF („beinhaltet“ 3NF)
- Eliminierung **transitiver Abhängigkeiten** auch zu Primattributen.

Beispiel 3NF vs. BCNF

cities: {[city, state, govenor, inhabitants]}

FDs

1. {city, state} $\rightarrow$ {inhabitants}
2. {state} $\rightarrow$ {governor}
3. {governor} $\rightarrow$ {state}

Kandidatenschlüssel:
{city, state} and {city, governor}

Beispiel 3NF vs. BCNF

cities: {[city, state, govenor, inhabitants]}

FDs

1. {city, state} → {inhabitants}
2. {state} → {governor}
3. {governor} → {state}

Kandidatenschlüssel:
{city, state} and {city, governor}

Ist die Relation in 3NF?

- Trivialität? (Bedingung 1)
- Ist die linke Seite ein Superschlüssel? (Bedingung 2)
- Ist die rechte Seite prim? (Bedingung 3)

Beispiel 3NF vs. BCNF

cities: {[city, state, govenor, inhabitants]}

FDs

1. {city, state} → {inhabitants}
2. {state} → {governor}
3. {governor} → {state}

Kandidatenschlüssel:
{city, state} and {city, governor}

Ist die Relation in 3NF?

- Trivialität? (Bedingung 1) – Keine der FDs ist trivial
- Ist die linke Seite ein Superschlüssel? (Bedingung 2)
- Ist die rechte Seite prim? (Bedingung 3)

Beispiel 3NF vs. BCNF

cities: {[city, state, govenor, inhabitants]}

FDs

1. ✓ {city, state} → {inhabitants}
2. {state} → {governor}
3. {governor} → {state}

Kandidatenschlüssel:
{city, state} and {city, governor}

Ist die Relation in 3NF?

- Trivialität? (Bedingung 1) – Keine der FDs ist trivial
- Ist die linke Seite ein Superschlüssel? (Bedingung 2) – FD1
- Ist die rechte Seite prim? (Bedingung 3)

Beispiel 3NF vs. BCNF

cities: {[city, state, govenor, inhabitants]}

FDs

- ✓ {city, state} → {inhabitants}
- ✓ {state} → {governor}
- ✓ {governor} → {state}

Kandidatenschlüssel:
{city, state} and {city, governor}

Ist die Relation in 3NF?

- Trivialität? (Bedingung 1) – Keine der FDs ist trivial
- Ist die linke Seite ein Superschlüssel? (Bedingung 2) – FD1
- Ist die rechte Seite prim? (Bedingung 3) – FD2 und FD3

Beispiel 3NF vs. BCNF

cities: {[city, state, govenor, inhabitants]}

FDs

- ✓ {city, state} → {inhabitants}
- ✓ {state} → {governor}
- ✓ {governor} → {state}

Kandidatenschlüssel:
{city, state} and {city, governor}

Ist die Relation in 3NF? JA

- Trivialität? (Bedingung 1) – Keine der FDs ist trivial
- Ist die linke Seite ein Superschlüssel? (Bedingung 2) – FD1
- Ist die rechte Seite prim? (Bedingung 3) – FD2 und FD3

Beispiel 3NF vs. BCNF

cities: {[city, state, govenor, inhabitants]}

FDs

1. ✓ {city, state} → {inhabitants}
2. {state} → {governor}
3. {governor} → {state}

Kandidatenschlüssel:
{city, state} and {city, governor}

Ist die Relation in BCNF? NEIN

- Trivialität? (Bedingung 1) – Keine der FDs ist trivial
- Ist die linke Seite ein Superschlüssel? (Bedingung 2) – FD1
- ~~Ist die rechte Seite prim? (Bedingung 3) – FD2 und FD3~~

BCNF Zerlegung

Es ist immer möglich ein Relationenschema $\mathcal{R}$ mit FDs F in

- 3NF Relationenschemata $\mathcal{R}_1, \ldots, \mathcal{R}_n$ zu zerlegen, so dass die Zerlegung
  - verlustfrei ist
  - Abhängigkeiten bewahrt
- BCNF Relationenschemata $\mathcal{R}_1, \ldots, \mathcal{R}_n$ zu zerlegen, so dass die Zerlegung
  - verlustfrei ist

Es ist nicht immer möglich eine BCNF Zerlegung $\mathcal{R}_1, \ldots, \mathcal{R}_n$ of $\mathcal{R}$ zu erzeugen, die Abhängigkeiten bewahrt.

Zerlegungsalgorithmus für BCNF

- $result = \{\mathcal{R}\}$, berechne F^+

Zerlegungsalgorithmus für BCNF

- $result = \{\mathcal{R}\}$, berechne F^+
- So lange es ein Relationenschema $\mathcal{R}_i \in result$ gibt, das nicht in BCNF ist, mache Folgendes:

Zerlegungsalgorithmus für BCNF

- $result = \{\mathcal{R}\}$, berechne F^+
- So lange es ein Relationenschema $\mathcal{R}_i \in result$ gibt, das nicht in BCNF ist, mache Folgendes:
  - Es gibt also eine für $\mathcal{R}_i$ geltende nicht-triviale FD $\alpha \rightarrow \beta$ mit:
    - $\alpha \cap \beta = \emptyset$
    - $\{\alpha \rightarrow \mathcal{R}_i\} \notin F^+$ (α ist kein Superschlüssel von $\mathcal{R}_i$)

Zerlegungsalgorithmus für BCNF

- $result = \{\mathcal{R}\}$, berechne F^+
- So lange es ein Relationenschema $\mathcal{R}_i \in result$ gibt, das nicht in BCNF ist, mache Folgendes:
  - Es gibt also eine für $\mathcal{R}_i$ geltende nicht-triviale FD $\alpha \rightarrow \beta$ mit:
    - $\alpha \cap \beta = \emptyset$
    - $\{\alpha \rightarrow \mathcal{R}_i\} \notin F^+$ (α ist kein Superschlüssel von $\mathcal{R}_i$)
  - Zerlege $\mathcal{R}_i$ in $\mathcal{R}_{i1} := \alpha \cup \beta$ und $\mathcal{R}_{i2} := \mathcal{R}_i - \beta$
    $result := (result - \mathcal{R}_i) \cup \mathcal{R}_{i1} \cup \mathcal{R}_{i2}$

Zerlegungsalgorithmus für BCNF

- $result = \{\mathcal{R}\}$, berechne F^+
- So lange es ein Relationenschema $\mathcal{R}_i \in result$ gibt, das nicht in BCNF ist, mache Folgendes:
  - Es gibt also eine für $\mathcal{R}_i$ geltende nicht-triviale FD $\alpha \rightarrow \beta$ mit:
    - $\alpha \cap \beta = \emptyset$
    - $\{\alpha \rightarrow \mathcal{R}_i\} \notin F^+$ (α ist kein Superschlüssel von $\mathcal{R}_i$)
  - Zerlege $\mathcal{R}_i$ in $\mathcal{R}_{i1} := \alpha \cup \beta$ und $\mathcal{R}_{i2} := \mathcal{R}_i - \beta$
    $result := (result - \mathcal{R}_i) \cup \mathcal{R}_{i1} \cup \mathcal{R}_{i2}$

Um zu überprüfen, ob α ein Superschlüssel ist, können wir auch α^+ berechnen statt F^+.

Beispiel

cities: {[city, state, governor, inhabitants]}

Kandidatenschlüssel: {city, state} und {city, governor}

FDs:

1. $\{\textbf{state}\} \rightarrow \{\textbf{governor}\}$
2. $\{\text{city, state}\} \rightarrow \{\text{inhabitants}\}$
3. $\{\text{governor}\} \rightarrow \{\text{state}\}$

Beispiel

cities: {[city, state, governor, inhabitants]}

Kandidatenschlüssel: {city, state} und {city, governor}

FDs:

1. $\{\textbf{state}\} \rightarrow \{\textbf{governor}\}$
2. $\{\text{city, state}\} \rightarrow \{\text{inhabitants}\}$
3. $\{\text{governor}\} \rightarrow \{\text{state}\}$

$\mathcal{R}_{i1}$ governments: {[state, governor]}

$\mathcal{R}_{i2}$ cities: {[city, state, inhabitants]}

Diese Zerlegung ist verlustfrei und bewahrt Abhängigkeiten.

- FD1 und FD3 können governments zugewiesen werden.
- FD2 kann cities zugewiesen werden.

Zerlegung in BCNF

zipCatalog: {[street, city, region, zip]}

Kandidatenschlüssel: {street, city, region}

FDs

1. $\{\text{street, city, region}\} \rightarrow \{\text{zip}\}$ OK
2. $\{\text{zip}\} \rightarrow \{\text{city, region}\}$ verletzt BCNF

Zerlegung in BCNF

zipCatalog: {[street, city, region, zip]}

Kandidatenschlüssel: {street, city, region}

FDs

1. {street, city, region} → {zip} OK
2. {zip} → {city, region} verletzt BCNF

Zerlegung anhand von FD2:

- cities: {[zip, city, region]}
- streets: {[zip, street]}

Zerlegung in BCNF

zipCatalog: {[street, city, region, zip]}

Kandidatenschlüssel: {street, city, region}

FDs

1. {street, city, region} → {zip} OK
2. {zip} → {city, region} verletzt BCNF

Zerlegung anhand von FD2:

- cities: {[zip, city, region]}
- streets: {[zip, street]}

Diese Zerlegung

- ist verlustfrei
- bewahrt aber nicht Abhängigkeiten! FD1 kann nicht überprüft werden, ohne den Join zu berechnen

Übersicht bisher diskutierter Normalformen

Normalform	**Kurzcharakteristik**
1NF	nur atomare Attribute
2NF	keine partielle Abhängigkeit
3NF	keine transitive Abhängigkeit (Ausnahme: Primattribute)
BCNF	keine transitive Abhängigkeit

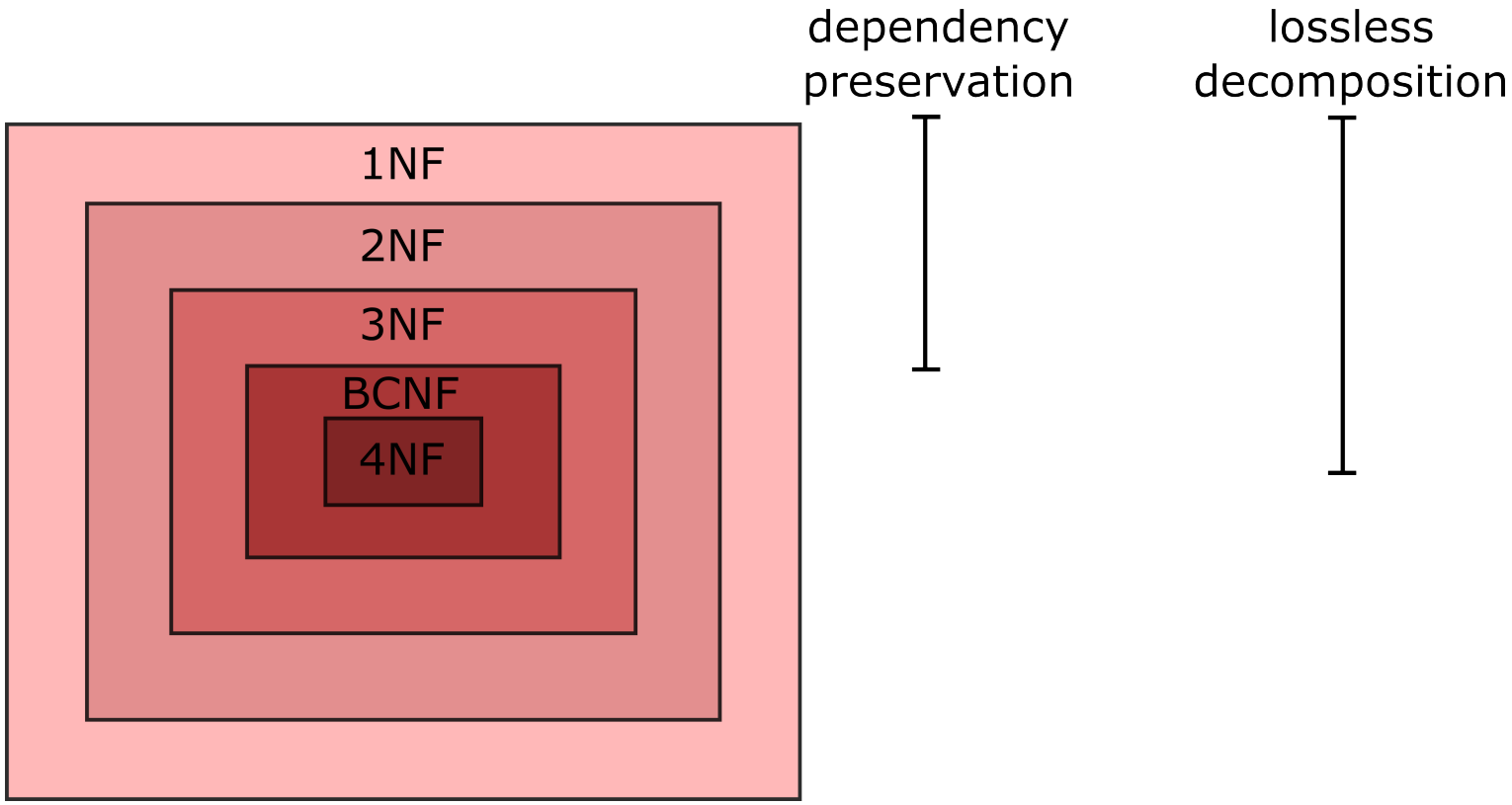
Übersicht bisher diskutierter Normalformen

Normalform	**Kurzcharakteristik**
1NF	nur atomare Attribute
2NF	keine partielle Abhängigkeit
3NF	keine transitive Abhängigkeit (Ausnahme: Primattribute)
BCNF	keine transitive Abhängigkeit

In der Praxis gibt man sich bei einer BCNF-Zerlegung, die zu einem Abhängigkeitsverlust führen würde, mit der dritten Normalform zufrieden.

Normalformen

- Die Verlustlosigkeit ist für alle Zerlegungsalgorithmen in alle Normalformen garantiert.
- Die Abhängigkeitsbewahrung kann nur bis zur dritten Normalform garantiert werden.



Zusammenfassung

Eigenschaften eines „guten“ relationalen Entwurfs

- Vermeidung von Redundanz, Anomalien und Null-Werten
- Informationen können dargestellt werden

Funktionale Abhängigkeiten (FDs)

- Das wichtigste Konzept beim relationalen Datenbankentwurf
- Armstrong-Axiome, um die Hülle eine Menge von FDs zu bestimmen
- Kanonische/minimale Überdeckung
- FDs sind ein Werkzeug, um ein „guten“ relationalen Entwurf zu garantieren

Zusammenfassung

Zerlegung

- Hilft Relationenschemata zu verbessern
- Versucht, funktionale Abhängigkeiten in Schlüsselabhängigkeiten zu verwandeln
- Ein gutes ER-Modell und Abbildung auf Relationen führt oft direkt zu Relationen in 3NF (oder höhere NF).

3NF

- Verlustfrei und abhängigkeitsbewahrend
- Kann immer erreicht werden

BCNF

- Verlustfrei aber nicht immer abhängigkeitsbewahrend

Zusammenfassung

Zerlegung

- Hilft Relationenschemata zu verbessern
- Versucht, funktionale Abhängigkeiten in Schlüsselabhängigkeiten zu verwandeln
- Ein gutes ER-Modell und Abbildung auf Relationen führt oft direkt zu Relationen in 3NF (oder höhere NF).

3NF

- Verlustfrei und abhängigkeitsbewahrend
- Kann immer erreicht werden

BCNF

- Verlustfrei aber nicht immer abhängigkeitsbewahrend

Denormalisierung:
Prozess bei dem die Normalisierung „zurückgenommen“ wird, um Anfragen schneller bearbeiten zu können

Anhang

5. Funktionale Abhängigkeiten
   - Funktionale Abhängigkeiten überprüfen

6. Normalisierung durch Zerlegung von Relationen
   - Verlustfreie Zerlegung
   - Abhängigkeitsbewahrung

Funktionale Abhängigkeiten überprüfen

Wir überprüft man, ob eine konkrete FD gilt?

Funktionale Abhängigkeiten überprüfen

Alternative Formulierung

Eine funktionale Abhängigkeit (FD) $\alpha \rightarrow \beta$ gilt für $\mathcal{R}$, wenn Folgendes für alle Ausprägungen von $\alpha = \{A_1, \ldots, A_i\}$ gilt:

$$|\pi_\beta(\sigma_{A_1=c_1 \wedge \cdots \wedge A_i=c_i}(R))| \leq 1,$$

wobei $c_1 \ldots c_i$ die möglichen Werte für α sind.

Funktionale Abhängigkeiten überprüfen

Simpler Algorithmus, um eine funktionale Abhängigkeit zu überprüfen

boolean testFD(relation R, FD $\alpha \rightarrow \beta$){
    erstelle Gruppen G_i von Tupeln mit den gleichen Werten für α;
    für jede Gruppe $G_i \subseteq R$:
        if nicht alle β gleich, then:
            return false;
        else:
            return true;
}

Beispielzerlegung

parents		
father	mother	child
Johann	Martha	Else
Johann	Maria	Theo
Heinz	Martha	Cleo

Zerlegung

$\pi_{father,child}$

fathers	
father	child
Johann	Else
Johann	Theo
Heinz	Cleo

$\pi_{mother,child}$

mothers	
mother	child
Martha	Else
Maria	Theo
Martha	Cleo

Annahme: Namen sind einzigartig.

Beispielzerlegung

parents		
father	mother	child
Johann	Martha	Else
Johann	Maria	Theo
Heinz	Martha	Cleo

Zerlegung

$\pi_{father,child}$

fathers	
father	child
Johann	Else
Johann	Theo
Heinz	Cleo

$\pi_{mother,child}$

mothers	
mother	child
Martha	Else
Maria	Theo
Martha	Cleo

Annahme: Namen sind einzigartig.

Verlustfrei oder nicht?

Beispielzerlegung

parents		
father	mother	child
Johann	Martha	Else
Johann	Maria	Theo
Heinz	Martha	Cleo

Zerlegung

$\pi_{father,child}$

fathers	
father	child
Johann	Else
Johann	Theo
Heinz	Cleo

$\pi_{mother,child}$

mothers	
mother	child
Martha	Else
Maria	Theo
Martha	Cleo

Annahme: Namen sind einzigartig.

Es ist verlustfrei: das gemeinsame Attribut “child„ ist ein Superschlüssel in beiden neuen Relationen.

Beispiel einer verlustfreien Zerlegung

Erklärungen

- $\mathcal{R} = \{father, mother, child\}$
- $F_{\mathcal{R}} = \{\{child\} \rightarrow \{mother\}, \{child\} \rightarrow \{father\}\}$
- $\mathcal{R}_1 = \{father, child\}$, $\mathcal{R}_2 = \{mother, child\}, \mathcal{R}_1 \cap \mathcal{R}_2 = \{child\}$
- Dann muss eine der folgenden Bedingungen erfüllt sein:
  - $\{father\} \subseteq \mathit{AttrClosure}(F_{\mathcal{R}}, \{child\})$
    oder
  - $\{mother\} \subseteq \mathit{AttrClosure}(F_{\mathcal{R}}, \{child\})$

Beispiel einer verlustfreien Zerlegung

Erklärungen

- $\mathcal{R} = \{father, mother, child\}$
- $F_{\mathcal{R}} = \{\{child\} \rightarrow \{mother\}, \{child\} \rightarrow \{father\}\}$
- $\mathcal{R}_1 = \{father, child\}$, $\mathcal{R}_2 = \{mother, child\}, \mathcal{R}_1 \cap \mathcal{R}_2 = \{child\}$
- Dann muss eine der folgenden Bedingungen erfüllt sein:
  - $\{father\} \subseteq \mathit{AttrClosure}(F_{\mathcal{R}}, \{child\})$
    oder
  - $\{mother\} \subseteq \mathit{AttrClosure}(F_{\mathcal{R}}, \{child\})$
- Beides gilt in diesem Beispiel!

Die Zerlegung ist verlustfrei.

Beispiel

zipCatalog: $\{[\text{ street, city, region, zip }]\}$

FDs

- $\{\text{zip}\} \rightarrow \{\text{city, region}\}$
- $\{\text{street, city, region}\} \rightarrow \{\text{zip}\}$

Zerlegung

- streets: $\{[\text{ zip, street }]\}$
- cities: $\{[\text{ zip, city, region }]\}$

Beispiel

zipCatalog			
city	region	street	zip
Frankfurt	Hesse	Goethestraße	60313
Frankfurt	Hesse	Galgenstraße	60437
Frankfurt	Brandenburg	Goethestraße	15234

$\pi_{zip,street}$

street	
zip	street
15234	Goethestraße
60313	Goethestraße
60437	Galgenstraße

$\pi_{city,region,zip}$

cities		
city	region	zip
Frankfurt	Hesse	60313
Frankfurt	Hesse	60437
Frankfurt	Brandenburg	15234

Verlustfrei, da $\{\text{zip}\} \rightarrow \{\text{city, region}\}$

Beispiel

zipCatalog			
city	region	street	zip
Frankfurt	Hesse	Goethestraße	60313
Frankfurt	Hesse	Galgenstraße	60437
Frankfurt	Brandenburg	Goethestraße	15234

$\pi_{zip,street}$

street	
zip	street
15234	Goethestraße
60313	Goethestraße
60437	Galgenstraße

$\pi_{city,region,zip}$

cities		
city	region	zip
Frankfurt	Hesse	60313
Frankfurt	Hesse	60437
Frankfurt	Brandenburg	15234

Die Zerlegung ist **nicht** abhängigkeitsbewahrend :
{street, city, region} $\rightarrow$ {zip} kann in der Zerlegung nicht überprüft werden