universität wien

Kapitel 7

Funktionale Abhängigkeiten und Normalformen

Inhalt

7.1 Problemstellung

Ziel, formale Kriterien zu finden, um die "Güte" eines modellierten Relationenschemas beurteilen zu können

Eigenschaften „guter“ Relationenschema:

- Vermeidung von **Redundanzen**
- Vermeidung von **Anomalien**

Beispiel

Lagerbestand

ArtikelNr	Menge	LagerNr	Lageradresse
101	25	1	Waagg. 10
102	5	2	Griesg. 12
102	10	1	Waagg. 10
105	23	4	Heuplatz 12

Probleme

Redundanz: Daten sind mehrfach vorhanden.

- die Adresse eines Lagers ist mehrfach gespeichert (siehe LagerNr 1)

Einfügeanomalie: Bekannte Daten können nicht eingefügt werden, weil andere Daten unbekannt (nicht vorhanden) sind.

- neues Lager 5 am Marktplatz 13 hat noch keine Teile und kann daher nicht eingefügt werden

Entfernungsanomalie: Beim Entfernen einer Information geht weitere Information verloren.

- Wenn der letzte Teil aus einem Lager entfernt wird, gehen auch die Daten über das Lager verloren

Änderungsanomalie: Wenn sich ein Tatbestand ändert, so müssen mehrere Tupel geändert werden.

- Wenn das Lager 1 auf eine neue Adresse verlegt wird, so muss diese Adresse bei jedem Teil, der im Lager vorrätig ist, geändert werden

Lösungsansatz

Vermeiden (bzw. Reduzieren) der Anomalien durch inhaltserhaltende Transformationen in sogenannte Normalformen (Normalisierungstheorie):

➔ **Zerlegung von Relationen**

Beispiel

Zerlegung der Relation “Lagerbestand” in 2 Relationen

Lagerbestand

ArtikelNr	Menge	LagerNr
101	25	1
102	5	2
102	10	1
105	23	4

Lager

LagerNr	Lageradresse
1	Waagg. 10
2	Griesg. 12
4	Heuplatz 12

7.2 Funktionale Abhängigkeiten

Formale Normalisierungstheorie liefert Lösungsansätze

- Stellt uns Mechanismen zur Verfügung, ähnliche Probleme zu lösen, d.h. Schema-Zerlegungen algorithmisch herzuleiten

Ein Werkzeug dazu sind **funktionale Abhängigkeiten**

Funktionale Abhängigkeiten können als **Integritätsbedingungen** gesehen werden

- Integritätsbedingungen sind Einschränkungen im Zustandsraum einer Datenbank (z.B. Fremdschlüssel)
- Funktionale Abhängigkeiten sind besondere Integritätsbedingungen, die für die Normalisierung verwendet werden.

Ist eine funktionale Abhängigkeit als statische Einschränkung auf einem Schema R definiert, so muss jede Relation r über R diese Einschränkung zu jedem Zeitpunkt erfüllen

Funktionale Abhängigkeit

Sei R ein Relationenschema, seien X und Y Teilmengen von R.
Eine Relation r (R) erfüllt die **funktionale Abhängigkeit** (functional dependency, FD) **X → Y**, wenn für je zwei Tupel u, v ∈ r (R) gilt: u (X) = v (X) ⇒ u (Y) = v (Y)

- X heißt linke Seite, Y rechte Seite der FD X → Y
- Die FD X → Y drückt aus: Wenn zwei Tupel gleiche Werte für alle Attribute in X haben, dann müssen auch ihre Y Werte übereinstimmen, d.h. die X-Werte bestimmen funktional abhängig die Y-Werte

Beispiel: Es seien auf dem relationalen Schema *Angestellter (AngNr, AbtNr, Ort)* die folgenden funktionalen Abhängigkeiten definiert: F = {AngNr → AbtNr, AbtNr → Ort}

Angestellter (FDs erfüllt)

AngNr	AbtNr	Ort
110	1	Linz
112	2	Wien
113	1	Linz

Angestellter (FDs nicht erfüllt)

AngNr	AbtNr	Ort
110	1	Linz
112	2	Wien
110	2	Graz

Volle funktionale Abhängigkeit, Schlüssel

Eine funktionale Abhängigkeit $X \rightarrow Y$ auf R heißt **volle funktionale Abhängigkeit** auf R, wenn für keine echte Teilmenge Z von X gilt: $Z \rightarrow Y$

- Die Notation für die volle funktionale Abhängigkeit ist $\mathbf{X \dot{\rightarrow} Y}$.
  - Manchmal wird der Punkt auch über den Pfeil geschrieben.

Das bedeutet, dass kein Attribut aus X entfernt werden kann, ohne dass die FD zerstört wird

- Die volle FD ist für einelementige X (ein Attribut) trivialerweise erfüllt

Diese Definition erlaubt es uns jetzt, die Schlüsseleigenschaft von Attributen mit Hilfe der funktionalen Abhängigkeit zu definieren

- X ist **Oberschlüssel** in R, wenn gilt: $X \rightarrow R$
- K ist **Schlüsselkandidat** in R, wenn gilt: $K \dot{\rightarrow} R$
- Ein Attribut, das in mindestens einem Schlüsselkandidat vorkommt, heißt **prim** (**primes Attribut**) in R oder Schlüsselattribut in R.

7.3 Ableitungsregeln für funktionale Abhängigkeiten

Problemstellung

Gegeben sei eine Menge F von funktionalen Abhängigkeiten

Welche zusätzlichen Abhängigkeiten f werden durch F impliziert (F |= f)?

Beispiel

aus F = { AngNr → AbtNr, AbtNr → Ort }

folgt auch: AngNr → Ort

|= Gültigkeitsbegriff, modelltheoretisch

|- Ableitbarkeitsbegriff, beweistheoretisch

Ziel ist daher ein Axiomensystem, dass F |= f ⇔ F |- f

Das heißt: Alles was gültig ist, ist auch herleitbar und umgekehrt

Axiomensystem von Armstrong

Sei R ein Relationsschema und X, Y, Z, W ⊆ R

(A1) **Reflexivität** (reflexivity)

$Y \subseteq X \quad \vdash X \rightarrow Y$

(A2) **Erweiterung** (augmentation)

$X \rightarrow Y, Z \subseteq W \quad \vdash XW \rightarrow YZ$

(A3) **Transitivität**

$X \rightarrow Y, Y \rightarrow Z \quad \vdash X \rightarrow Z$

Vollständigkeitssatz

Satz: Die Axiome (A1) bis (A3) sind

korrekt (sound),

d.h. sie leiten von F nur wirklich gültige FD ab (F |- f ⇒ F |= f), und

vollständig (complete),

d.h. sie erzeugen bei wiederholter Anwendung alle von F implizierten Abhängigkeiten (F |= f ⇒ F |- f).

Beispiel: Wir wollen ableiten $X \rightarrow Y, Z \subseteq Y \vdash X \rightarrow Z$

(1) $Y \rightarrow Z$ (Reflexivität angewendet auf $Z \subseteq Y$ (gegeben))

(2) $X \rightarrow Z$ (Transitivität von $X \rightarrow Y$ (gegeben) und $Y \rightarrow Z$ (siehe (1))

Siehe nächste Folie
Ableitungsregel (A6)

Weitere Ableitungsregeln

Weitere nützliche Ableitungsregeln, die aus den Armstrong Axiomen hergeleitet werden können:

(A4) **Vereinigung** (additivity)

$$X \rightarrow Y, X \rightarrow Z \quad \vdash X \rightarrow YZ$$

(A5) **Pseudotransitivität** (pseudotransitivity)

$$X \rightarrow Y, WY \rightarrow Z \quad \vdash WX \rightarrow Z$$

(A6) **Zerlegung** (projectivity)

$$X \rightarrow Y, Z \subseteq Y \quad \vdash X \rightarrow Z$$

Hülle von funktionalen Ableitungen

Die Menge aller funktionalen Abhängigkeiten, die von F auf R impliziert werden, heißt die **Hülle** (closure) von F auf R, geschrieben F_R^+

Ist die Attributmenge aus dem Kontext klar, wird nur F^+ geschrieben.

$F^+ = \{ f \mid F \models f \}$

Die Hülle X^+ einer Menge von Attributen $X \subseteq R$ bezüglich F ist die Menge aller Attribute A, sodass $X \rightarrow A$ aus F^+ ist.

$X^+ = \{ A \mid (X \rightarrow A) \in F^+ \}$

Beispiel

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

Wir wollen zeigen, dass AB ein Schlüsselkandidat für R ist, d.h. wir müssen zeigen, dass $\{AB\}^+ = R$

Notwendige, aber nicht hinreichende Bedingung für Schlüsselkandidaten

(1) $A \rightarrow C$	(gegeben)
(2) $AB \rightarrow ABC$	(Erweiterung von (1) durch AB)
(3) $B \rightarrow D$	(gegeben)
(4) $ABC \rightarrow ABCD$	(Erweiterung von (3) mit (ABC))
(5) $AB \rightarrow ABCD$	(Transitivität angewandt auf (2) und (4))

d.h. $\{AB\}^+ = R$

Das **Membership-Problem** für funktionale Abhängigkeiten

Problemstellung

Gegeben sei eine Menge von funktionalen Abhängigkeiten F auf R.

Welche zusätzlichen Abhängigkeiten werden durch F impliziert?

Ist $X \rightarrow Y$ Element von F^+ ?

Membership Algorithmus

CLOSURE (X, F)
//input: X … Attributmenge, $X \subseteq R$
// F … Menge von FDs über R
//output: X^+
$X^+ = X$
while $\exists (Y \rightarrow Z) \in F$ mit $Y \subseteq X^+$ und $Z \notin X^+$ **do**
$X^+ = X^+ \cup Z$
end
return (X^+)

MEMBER ($X \rightarrow Y$, F)
//input: $X \rightarrow Y$ … FD,
// F … Menge von FDs
//output: true … wenn $F \models X \rightarrow Y$
false … sonst

return ($Y \subseteq$ CLOSURE (X, F))

Das **Membership-Problem** für funktionale Abhängigkeiten

Problemstellung

Gegeben sei eine Menge von funktionalen Abhängigkeiten F auf R.

Welche zusätzlichen Abhängigkeiten werden durch F impliziert?

Ist $X \rightarrow Y$ Element von F^+ ?

Membership Algorithmus

CLOSURE (X, F)
//input: X … Attributmenge, $X \subseteq R$
// F … Menge von FDs über R
//output: X^+
$X^+ = X$
while (Änderung an X^+) **do**
foreach FD $Y \rightarrow Z$ **in** F **do**
if $Y \subseteq X^+$ **and** $Z \notin X^+$ **then** $X^+ = X^+ \cup Z$
return (X^+)

MEMBER ($X \rightarrow Y$, F)
//input: $X \rightarrow Y$ … FD,
// F … Menge von FDs
//output: true … wenn $F \models X \rightarrow Y$
false … sonst

return ($Y \subseteq$ CLOSURE (X, F))

Äquivalenz von Mengen funktionaler Abhängigkeiten

Zwei Mengen funktionaler Abhängigkeiten F und G sind **äquivalent** ($F \equiv G$), wenn $F^+ = G^+$ gilt. F heißt dann **Überdeckung** (cover) von G.

Algorithmus: $F \equiv G$?

für alle $g \in G : F \models g$? **und** für alle $f \in F : G \models f$?

Beispiel

$F = \{AB \rightarrow C, A \rightarrow B, A \rightarrow C, B \rightarrow C\}$

$G = \{AB \rightarrow C, A \rightarrow B, B \rightarrow C\}$

Frage: Ist $F \equiv G$?

(1) Prüfe für alle $g \in G$; ob $F \models g$ erfüllt

(2) Prüfe für alle $f \in F$; ob $G \models f$ erfüllt

$(A \rightarrow C) \in G^+$, d.h. $C \in A^+$ bzgl. G

$A^+_G = \{A, B, C\}$

daher: $F \equiv G$.

Definition: kanonische Überdeckung

Eine Menge von funktionalen Abhängigkeiten F_c heißt **kanonische Überdeckung** von F, wenn:

(1) $F_c \equiv F$ (i.e. $F_c^+ = F^+$)

(2) Keine funktionale Abhängigkeit in F_c enthält überflüssige Attribute.
   a) $\forall A \in \alpha, \alpha \to \beta \in F_c : F_c - (\alpha \to \beta) \cup ((\alpha - A) \to \beta)) \not\equiv F_c$
   b) $\forall B \in \beta, \alpha \to \beta \in F_c : F_c - (\alpha \to \beta) \cup (\alpha \to (\beta - B)) \not\equiv F_c$

(3) Jede linke Seite einer funktionalen Abhängigkeit tritt in F_c nur einmal auf.

Motivation: F^+ ist zwar eindeutig, aber für praktische Belange meist zu umfangreich. Die kanonische Überdeckung gibt eine standardisierte, in gewissem Sinne minimale Menge von funktionalen Abhängigkeiten.

Die kanonische Überdeckung ist nicht unbedingt eindeutig.

Existenz kanonischer Überdeckung

Satz: Zu jeder Menge von funktionalen Abhängigkeiten gibt es eine kanonische Überdeckung

Konstruktion:

- Linksreduktion: Für jede funktionale Abhängigkeit $\alpha \rightarrow \beta$ und jedes $A \in \alpha$: Ersetze $\alpha \rightarrow \beta$ durch $(\alpha - A) \rightarrow \beta$, falls $\beta \epsilon\ (\alpha - A)^+$
- Rechtsreduktion: Für jede funktionale Abhängigkeit $\alpha \rightarrow \beta$ und jedes $B \in \beta$: Ersetze $\alpha \rightarrow \beta$ durch $\alpha \rightarrow (\beta - B)$, falls $B \epsilon\ \alpha^+$ bezüglich $F_c - (\alpha \rightarrow \beta) \cup (\alpha \rightarrow (\beta - B))$
- Entferne „sinnlose“ funktionale Abhängigkeiten der Form $\alpha \rightarrow \emptyset$
- Fasse Abhängigkeiten mit gleicher linker Seite zusammen (aus $\alpha \rightarrow \beta$ und $\alpha \rightarrow \gamma$ wird $\alpha \rightarrow \beta\gamma$)

Beispiel kanonische Überdeckung

Finde kanonische Überdeckung für G = {A → BC, B → C, AB → D}

(1) Linksreduktion:

Ist A in AB → D überflüssig?
B^+ bezüglich {A → BC, B → C, AB → D} ist gleich {BC} enthält D nicht nein
Ist B in AB → D überflüssig?
A^+ bezüglich {A → BC, B → C, AB → D} ist gleich {ABCD} enthält D ja
G‘ = {A → BC, B → C, A → D}

(2) Rechtsreduktion:

Ist B in A → BC überflüssig?
A^+ bezüglich {A → C, B → C, A → D} ist gleich {ACD} enthält B nicht nein
Ist C in A → BC überflüssig?
A^+ bezüglich {A → B, B → C, A → D} ist gleich {ABCD} enthält C ja
G‘‘={A → B, B → C, A → D}
Ist C in B → C überflüssig?
B^+ bezüglich {A → B, B → ∅, A → D} ist gleich {B} enthält C nicht nein
Ist D in A → D überflüssig?
A^+ bezüglich {A → B, B →C, A → ∅} ist gleich {ABC} enthält D nicht nein

(3) Entferne Abhängigkeiten der Form $\alpha \rightarrow \emptyset$:

keine vorhanden

(4) Fasse linke Seiten zusammen: G_c= {A → BD, B → C}

Systematische Ermittlung aller Schlüsselkandidaten

Gegeben: R = (S, F)

Gesucht: alle Schlüsselkandidaten für R

Methode

(1) Ermittle folgende Teilmengen von S

N = alle Attribute von S, die in FDs aus F nicht vorkommen

R = alle Attribute von S, die in FDs aus F nur auf der rechten Seite vorkommen

L = alle Attribute von S, die in FDs aus F nur auf der linken Seite vorkommen

M = S - N - R - L

Für alle Schlüsselkandidaten muss gelten: $K = NLH$ mit $H \subseteq M$

Systematische Ermittlung aller Schlüsselkandidaten (2)

(2) Schicht 0: S' = NLM

(3) Schicht 1:

Alle gültigen Teilmengen von S' mit n - 1 Attributen

Ueberprüfe für jedes $X \in M : S' - X \rightarrow X$

Wenn ja: Oberschlüssel

Wenn nein: Nichtschlüssel

(4) Schicht i:

Enthält alle gültigen Teilmengen von S' mit n - i Attributen, die Oberschlüssel sind

(5) Generiere Schichten, solange Oberschlüssel vorhanden.

(6) Schlüsselkandidaten sind alle Attributmengen K, die selbst Oberschlüssel sind und für die gilt, dass keine Teilmenge Oberschlüssel ist

Beispiel

R(S,F) = (
{A, B, C, D, E, F, G},
{AB → EG, DE → C, AC → BD}
)

N	=	{F}
R	=	{G}
L	=	{A}
M	=	{B, C, D, E}

0) A F und einige von B C D E

1) AFCDE → B ja
AFBDE → C ja
AFBCE → D ja
AFBCD → E ja

Schlüsselkandidaten: AFC, AFBD, AFDE

2) AFDE → BC ja
AFCE → BD ja
AFCD → BE ja
AFBE → CD nein
AFBD → CE ja
AFBC → DE ja

3) AFD → BCE nein
AFE nicht nötig
AFC → BDE ja
AFB nicht nötig

4) AF → BCDE nein

Schlüsselkandidaten suchen von unten nach oben (darauf achten, dass nicht bereits ein Teil schon ein Schlüsselkandidat ist, z.B. wenn ACF ein Schlüsselkandidat, dann kann AFBC kein Schlüsselkandidat sein)

1. Normalform (1NF)

Ein Relationenschema R ist in 1. Normalform, wenn die Wertebereiche aller Attribute von R atomar sind, d.h. die Attributwerte dürfen nicht strukturiert (komplexe Werte) sein

- Die 1. Normalform ist in reinen relationalen Datenbanksystemen immer erfüllt
- Dies gilt aber nicht für objekt-relationale oder objekt-orientierte Datenbanksysteme

2. Normalform (2NF)

Ein Relationenschema R ist in 2. Normalform, wenn es in 1NF ist und wenn jedes nicht prime Attribut von jedem Schlüsselkandidaten von R voll funktional abhängig ist.

Intuition 2NF

Intuitiv ist folgendes klar: wenn ein nichtprimes Attribut von einem Teil eines Schlüssels abhängig ist, kann dies zu Anomalien führt

- Siehe unser erstes Beispiel *Lagerbestand(ArtikelNr, Menge, LagerNr, Lageradresse)*
- Auf diesem Schema sind folgende zwei funktionale Abhängigkeiten definiert:
  - FD1: ArtikelNr, LagerNr → Menge, d.h. *ArtikelNr* und *LagerNr* bestimmen gemeinsam *Menge*
  - FD2: LagerNr → Lageradresse, d.h. das Attribut *LagerNr* bestimmt die *Lageradresse*
- *ArtikelNr* und *LagerNr* sind gemeinsam Schlüssel
- Aufgrund der FD2 ist das Schema aber nicht in 2. Normalform, da das nicht prime Attribut *Lageradresse* nur von einem Teil des Schlüssels, nämlich *LagerNr* allein, abhängig ist, d.h. *Lageradresse* ist nicht voll funktional abhängig vom Schlüssel

Das Schema *Lagerbestand* kann aber durch Zerlegung einfach in die 2. Normalform gebracht werden, in dem man zwei Schemata *Lagerbestand2* und *Lager* erzeugt:

- *Lagerbestand2(ArtikelNr, Menge, LagerNr)* , auf dem die FD1: ArtikelNr, LagerNr → Menge gilt und
- *Lager(LagerNr, Lageradresse),* auf dem die FD2: LagerNr → Lageradresse gilt

3. Normalform

Eine Relation R ist in **3. Normalform (3NF),** wenn sie in der 2. Normalform ist und kein nicht primes Attribut transitiv vom Schlüssel in R abhängt

Die 3. Normalform ist also dann verletzt, wenn ein nicht primes Attribut über einen Umweg (eine Weiterleitung) über andere Attribute vom Schlüssel abhängig ist

Formal lässt sich die **transitive Abhängigkeit** so definieren:

Gegeben sei eine Menge funktionaler Abhängigkeiten F auf R und $X \subseteq R$, $A \in R$

A ist **transitiv abhängig** von X, wenn es eine Attributmenge $Y \subseteq R$ gibt, sodass $(X \rightarrow Y) \in F^+$ und $(Y \rightarrow A) \in F^+$ sind und $(Y \rightarrow X) \notin F^+$, $A \notin XY$ gilt

Beispiel 3NF

Gegeben sei das Schema *Angestellter (AngNr, AbtNr, Ort)*

Darauf sind zwei funktionale Abhängigkeiten definiert:

FD1: AngNr → AbtNr

FD2: AbtNr → Ort

Dieses Schema ist nicht in 3. Normalform, da folgendes gilt

AngNr → AbtNr → Ort

Ort ist nur transitiv (über *AbtNr*) vom Schlüssel *AngNr* abhängig

Die sich dadurch ergebenden Anomalien kann man einfach erkennen

Das Schema *Angestellter* kann aber durch Zerlegung in die 3. Normalform gebracht werden:

Angestellter2(AngNr, AbtNr), mit der FD1: AngNr → AbtNr und

Abteilung(AbtNr, Ort) mit der FD2: AbtNr → Ort

Datenbankschema in xNF

Gegeben sei ein Relationenschema R mit Attributmenge S und funktionalen Abhängigkeiten F, d.h. R = (S, F)

Ein **relationales Datenbankschema** ist eine Menge von Relationenschemata: $R = \{R_1, \ldots, R_n\}$

Ein Datenbankschema R ist in **xNF** (d.h. 1NF, 2NF, 3NF, …) wenn jedes Relationenschema in R in xNF (d.h. 1NF bzw. 2NF bzw. 3NF …) ist